# PROCEEDINGS OF
# COMMON JOINT EASTERN
# MIDWESTERN REGION

OCTOBER 6, 7, 8, 1965

AT AMERICANA HOTEL, NEW YORK, NEW YORK

NORMAN GOLDMAN
REGIONAL SECRETARY

# TABLE OF CONTENTS

# TABLE OF CONTENTS

CONTINUED

PAGES

# TABLE OF CONTENTS

## CONTINUED

PAGES

# NOTES FROM ADVANCED MONITOR WORKSHOP

## DIM & EQUIVALANCE

Dim always starts at sector 4800. Equi immediately after.

Dim entry 20 positions
- Disk address
- sector count
- loading address
- entry address
- record mark or group mark

Load add = $\overline{9}$9999→non-core image, flag on units position→loaded by subroutine supervisor.

Last position may be flagged to indicate file protection and perm. assignment.

## SEQUENTIAL PROGRAM LIST

Cylinder 99, indicates availability of disk storage. Must correspond with Dim table. System table editor may be used to check correspondance.

## WORK AREA

1) Used for assemblies and compilations

2) Used for FORTRAN disk I/O logical record 1 starts at sector 219.

3) Sectors 0-199 used to store F I/O and arith for overlays.

4) Sectors 200-218 used to store DK I/O.

5) Going back from end of work area--local tables and locals for mainline if necessary.

## GENERAL USE OF DUP

1) Whenever disk sector address required drive code must be specified. (1, 3, 5, or 7)

2) First job card after multi-drive definition must contain module change codes.

3) All packs used by system must have splist-dlabl--beware of 4800 in Dim entry 3.

## FORTRAN SUBPROGRAMS IN SPS

Linkage

```
     BTM    Name,*+11
     DSA    A,B,...
```

Indicator Record

```
   S  DS       ,*+101
      DC      6,987898,5-S
      DAC     6,NAMEbb,7-S
      DVLC    22-S,5LAST,2,ff
              2,kk,5,ENTRY-6,
              5,0,30,0
      DSC    17,0,0
      DORG    S-100
        .
        .
        .
      DC      5,0
        .
ENTRY   .     :
        .
        .
        .
LAST  DC      1,@   (even)
```

## DETERMINATION OF DISK LOAD ADDRESSES

1) Check for break in address assignment.

2) Look for series of constants to define disk control field.

3) Look for TRA--TCD

```
E.G. #1      Page 7 super-iort monitor I

             DSC  1,1
             DSA  DSA04
             DC   3,20
             DC   6,402@

E.G. #2      Page 9 FII phase 1-A

             K    PHADDA,701
             WN   PHADDA,702
             TRA
             TCD  LDPHA
```

## SEGMENTS OF FORTRAN

**Phase I-A**

1) Move blocks of I-B to work area for fast access.

2) Calculate memory size.

3) Initialize symbol table starting at 16000 with ‡̄‡‡‡‡00000.

4) Store subroutine names in symbol table.

5) Initialize input area.

6) Read control records and set indicators.

7) Read communications sector.

8) Initialize FORTRAN communication area.

Phase I-B

1) Read statement into chi.

2) Place record mark at end of statement.

3) Create symbol and name tables.

4) Create strings and store in work area.

Phase I-C

Storage Allocation

1) Store constants in work area (cyl zero) in system output format.

2) Replace pointer with address of constant.

3) Bring statement numbers into sym table.

4) Place ‡ at end of entry as storage is allocated.

5) Check for undefined statement numbers.

6) Allocate non-constant storage.

Phase II

Generation of object code

1) In core (2218-10000), string & symbol, manipulation routines, housekeeping, statement number routine.

2) Secondary blocks (100000-14100), arithmetic translator, goto, if, I/O, function intialization.

3) Tertiary blocks (14100-16000), variable subscripting, literal subscripting, do.

<u>EXAMPLE</u>

A = -B1**C123

1) Symbol table initialization

```
16000  ‡̅‡‡‡‡00000
16010  ‡̅‡‡‡‡00000
                 .
                 .
                 .
19990  ‡̅‡‡‡‡00000
```

Symbol table entries start at 19999 and are 10 digits. First 5 are address of variable in name table. Last 5 are codes indicating what is in name table.

Name table entries start at 16000 and are variable in length.

2) Create symbol and name table entries and generate string.

```
16000     4̅1 4̅2 71 4̅3 71
16010     72 73 00 00 00
  .             .
  .             .
  .             .
19970     16 01 32 00 00    C123 Entry
19980     16 00 52 00 00    B1   Entry
19990     16 00 12 00 00    A    Entry
```

1̅999 0̅133 0̅129 1̅998 0̅115 1̅997 0̅132

0̅133 =
0̅129 unary -
0̅115 **
0̅132 ;

3) Storage allocation

```
 16000 4̅1 4̅2 71 4̅3 71
 16010 72 73 00 00 00
   .          .
   .          .
   .          .
 19970 0̅002920000
 19980 0̲̅001920000
 19990 0000920000
```

4) Generate object code through forcing table

| oper | LV | RV |
|---|---|---|
| = | 60 | 59 |
| unary - | 5 | 0 |
| ** | 5 | 4 |
| ; | 0 | 60 |

1̅9990̅133 0̅129 1̅998 0̅115 1̅997 0̅132

| Scan | LV of | RV of |
|---|---|---|
| 1 | 1̅999 | 0̅129 |
| 2 | 0̅133 | 1̅998 |
| 3 | 0̅115 | 0̅132 |

RV≥LV⇒ Generate and collapse string

LV of 0̅115 = 2          RV of 0̅132 = 60

Generate

```
          BTM    TOFAC, 19
          BTM    FLEXP, 29
```

Collapse

A= -FAC

$\overline{1}999$ $\overline{0}133$ $\overline{0}129$ $\overline{0}101$ $\overline{0}132$

LV of $\overline{0}129$ = 5    RV of $\overline{0}132$ = 60

Generate

```
        BTM    RSGN, 0
```

Collapse

A = FAC

LV of $\overline{0}133$ = 60    RV of $\overline{0}132$ = 60

Generate

```
        BTM-   FRFAC, 9
```

## FORTRAN LOADER
### Six Blocks

1) Initialization, read local cards, and build local tables in work area.

2) Save common ( to 21 sectors) and load mainline.

3) Load incore subroutines.

4) Load library routines and flipper if necessary.

5) Load locals if necessary.

6) Restore common, check N1 & N2, move I/O and arith into work area if necessary, and call in arith and I/O.

## SEGMENTATION OF I/O

Monitor I

9 overlays
1 arith 18 I/O

Monitor II

Variable length same as monitor I.
1 arith, 1 read, 1 write

## SUPERVISOR

1) Handles all reading of supervisor phases sets up read, write, or control function and executes it. Checks for error before operation is executed.

2) Error routine 0 brought in. If indicator 19 on. Updates error counters on disk. In core with error routines.

3) Error routine 0 plus determines if error is disk or non-disk error.

4) Error routine 1 disk error or cylinder overflow. If cylinder overflow, DDA adjusted. Up to 9 retries if disk error.

5) Error routine 2 determines other I/O errors and gives retry if possible (card punch).

6) Error routine 4 trap to here if read caused input of ** control card. Transfer made to monitor.

7) Bring in SPS supervisor or checks for loader.

8) Brings in reloc loader.

9) Brought in by loader and examined for return-indicates what source brought loader in.

10) Check if loader called by dup or monitor reread caller if necessary.

11) Handles reading and prolessing of all monitor control records.

12) Relocating loader.

Electric Utilities Programs Team
of
COMMON
(nee IBM 1620 Users Group)

Newsletter #26 October 20, 1965

The following were present at our October 7 meeting at the Americana Hotel in New York City.

Larry J. Dupre, Central Louisiana Electric Company
Barry J. Deliduka, Central Vermont Public Service Corporation
L. E. Cox, Jr., Memphis Light, Gas and Water Division
Thomas H. Farrow, Jr., Tampa Electric Company
Paul D. Folse, Tampa Electric Company
Richard W. Page, New York State Electric and Gas Corporation
Alvin L. Lipson, Virginia Electric and Power
George S. Haralampu, New England Electric System
Jene Y. Louis, Long Island Lighting Company
Stanley A. Clark, Public Service Company of New Hampshire
LeRoy Sluder, Jr., Long Island Lighting Company
Phillip R. Shire, Commonwealth Associates, Inc.
David C. Hopper, East Kentucky RECC
Robert F. Steinhart, IBM, New York
W. H. Morrow, Jr., IBM, New York
Carol Ziegler, Orange and Rockland Utilities, Inc.
R. A. Smails, Stone and Webster Service Corporation
O. B. Anderson, Jr., Southern Services, Inc.
J. E. Hernandez Betancourt, Puerto Rico Water Resources Authority
Herb Blaicher, Jersey Central Power and Light
Lutz P. Mueller, Jersey Central Power and Light
D. D. Williams, Baltimore Gas and Electric Company
Frank J. Wells, Long Island Lighting Company
Henry Mahlmann, Long Island Lighting Company
E. J. Orth, Jr., Southern Services, Inc.

Frank Wells, our Chairman, opened the meeting with a short business session. The FIRST ITEM OF BUSINESS discussed was organization of the Team. The 1620 Users Group now goes under the name of COMMON and includes users of the 1130, 1800, and System/360 Models 30 and 40. Those interested in utility applications who have this hardware on order are automatically included in the membership of our Team.

After considerable discussion, the group decided the best course would be to keep one Team, and not fragment ourselves into groups interested in one particular computer. The main reason for this decision is that our Team is a problem oriented group. Just as the wide range of 1620 models (from Bikini to late Victorian) has not interfered with discussion of our various problems, so also should the various computers not interfere with our information exchange.

The small computers will always be around. While many of our members will have access to a large computer in the accounting department, a small machine such as the 1130 will be most useful to give immediate answers on small to medium size jobs, and on jobs which require an immediate answer. There are cases where a smaller computer such as the 1800 would be used as a terminal for a larger centralized on-line computer.

Dick Page of New York State Electric and Gas Corporation raised the question, "What does belonging to COMMON do for us?". That is, why meet when COMMON meets? Al Lipson of Virginia Electric and Power Company explained the value of the hardware sessions at the general meetings. Other advantages of the general meetings are the sessions on operations research and statistical techniques, and the in-depth discussions of software.

We might mention at this point as an item of interest for newer members that the Utilities Team will not always meet in conjunction with the Eastern Region of COMMON. There have been instances where we did not meet with the Users Group at all. For instance, it has been our practice to meet at the biennial PICA Conference and skip that Users Group meeting. Since our membership is nationwide, we try to hold meetings away from the Eastern Region on a regular basis so that more folks from the West might attend. We meet twice a year.

While we are on the subject, our NEXT TEAM MEETING will be held at the Mid-Western Region meeting in St. Louis, February 9, 10, 11 at the Chase Park Plaza. A quick consensus of our members gave a thumbs down on meeting in Toronto at the joint Canadian-Eastern meeting on March 20. We hear via the grapevine that tutorial sessions on the 1130 and 1800 are to be held at the St. Louis meeting.

As a SECOND ITEM OF BUSINESS, Frank Wells informed the Team that due to changing responsibilities he has left the sphere of computer applications and is resigning Chairmanship of the Team. Frank appointed Ed Orth to act as interim Chairman. Ed Cox volunteered to act as interim Secretary.

Frank appointed a three-man nominating committee consisting of Don Williams, George Haralampu, and Al Lipson.

After the intermission, the nominating committee reported the nomination of the temporary Chairman and Secretary as candidates for permanent Chairman and Secretary. For your purposes, a ballot is attached to this Newsletter. Please mark your choice, fold it as indicated, add a stamp, and drop it in the mail by November 5.

Bob Steinhart of IBM distributed the LIST OF MODIFICATIONS TO THE 1620 ELECTRIC LOAD FLOW. A copy of this list is attached for those who did not attend the New York meeting. Bob also commented on the Electric Load Flow for the 1130. This program is under test at the present time, and will be available during the first or second quarter of 1966. Minimum configuration is 8K core with disk, card, and typewriter.

Concerning his modifications for the 1620 load flow, Ed Cox of the Memphis Light, Gas and Water Division would like to emphasize that they will prevent erroneous generator table overflows only when the number of generators in the system plus the number of generator changes made on one case do not exceed the table limits.

In order to clear up some rampant confusion on expected speed of the 1130 LOAD FLOW, Bill Morrow of IBM contacted Arno Glimn for us. Arno estimates that the program will operate at 60 buses per second per iteration. That will make it roughly thirty times faster than the 1620 Model I load flow. Please note that this is strictly an index of the solution time. Output medium (card, typewriter, printer) should be taken into consideration in arriving at a final ratio.

In answer to a request from George Haralampu of New England Electric Service on IBM COMMITTMENTS FOR THE 1130, Bob Steinhart sends us the following information.

> "With respect to the 1130, the following are being prepared: MATHPAK, COGO, Numerical Surface Techniques and Contour Map Plotting, and Statistical System. MATHPAK is a set of FORTRAN subprograms for function evaluation, matrix manipulation, etc. COGO is announced for availability during the Third Quarter of 1966, the others during the Second Quarter. We are aware of the need for an 1130 critical path scheduling program, but can say nothing more on this subject at present. Both the 1130 and 1800 are supported with FORTRAN, an assembly language, a monitor system for the disk-oriented configurations, etc. The 1800, in addition, is supported with the Time Sharing Executive System.
>
> "System/360 now consists of Models 20, 30, 40, 44, 50, 65, 67, and 75. All except Model 20 are exceptionally well suited for engineering and are fully supported with FORTRAN and other programming systems. Announced application program support includes the Scientific Subroutine Package (like MATHPAK for the 1130, but more extensive) which will be available this year, Project Management System (includes PERT), Mathematical Programming System (includes linear programming), and General Purpose Simulation System. The availability dates for the last three items have not yet been announced."

Concerning the 1800 LOAD FLOW, we hear that the 1130 source deck may be assembled on the 1800. The 1130 load flow under test is also apparently identical to the 360 load flow. Specifications of the 1130 load flow were summarized in Newsletter #23, dated June 18, 1965.

Bill Morrow had some interesting comments to make concerning IBM's EDUCATION EFFORTS FOR THE 1130 AND 1800. He indicated that a "teacher's teacher" course has been designed and distributed so that competence should now exist at the District level for 1130 and 1800 courses. Concerning this matter of courses, please carefully check background material assumed by the course. If the course is not as promised, or if you have suggestions for improving the course of material, please contact Bill Morrow. His full address follows:

W. H. Morrow, Jr.
Program Administrator
Public Utilities-Engineering and Operations
112 East Post Road
White Plains, New York 10601

Bill will sincerely appreciate your comments. It is only through our feedback that IBM can design the best possible course for their customers.

Quite a bit of conflicting information has been going around concerning CORE REQUIREMENTS FOR THE ENGINEERING OPERATING SYSTEM. The latest word is that the EOS requires the full operating system. Without communications capability, minimum core requirements are 128K. With communications, core requirements are 256K. Bill Morrow is sending us some material concerning the EOS. We will forward it to the membership when it comes in. You also may wish to check back Newsletters.

Phil Shire of Commonwealth Associates discussed some experimentation they have done to hasten CONVERGENCE OF THE 1620 LOAD FLOW. He has not had much success with a technique for random choice of acceleration factor. However, much better results were obtained by increasing alpha by 0.2 on the two buses with the highest mismatch. Phil discussed three programs which he hopes to get in the Library before too long. They are: 1) A steam distribution program for steam networks, programmed in FORTRAN and SPS; 2) A column design program using the AISC formulas, programmed in UTO FORTRAN; 3) A program for circuit routing in power plants using dynamic programming techniques, coded in SPS.

Don Williams of Baltimore Gas and Electric Company gave a very interesting discussion on MANAGEMENT INFORMATION SYSTEMS, telling us about the system being worked up at his company. Of particular interest is their transformer load management system. Don stressed the need for looking at the whole picture ahead of time. He who jumps in and starts work without considering how all the details fit together runs a great risk of going in all directions at once.

Don mentioned that Detroit Edison has mechanized the ordering of materials and feeder design. That is, input data of so many miles of such and such construction causes output of a complete bill of materials including costs.

There is a very definite need for critical path techniques in a management information system. When CPM plus the proper reporting techniques are built in, management can tell almost on a day to day basis exactly what has been done, what remains to be done, the items holding up the works, and money spent. That is, management will not have to wait for this information to filter down through the various channels. In its final form, such an information system will place an inquiry station at the finger tips of top management.

Many believe that engineers are best suited to design such a total information system due to their experience with the electrical system, and due to the nature of their education. However, as Herb Blaicher of Jersey Central emphasized, "Such a system assumes a high degree of sophistication at top management. It is up to us to sell them through a proper education program".

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Thus, our meeting.

With the changes wrought by COMMON, our Team is now the largest and most active in our industry. The ferment generated by the various new generation computers will produce many new ideas and methods. We are fortunate in our industry that competitive interests do not hinder information and program exchange. If our past success in the limited area of the 1620 is to be magnified in our new and larger scope, we must continue in our efforts to communicate with one another.

This is our last Newsletter. Ed Cox will be taking over the Newsletter along with the other duties of Team Secretary. We have enjoyed serving you all. Let us all continue in our support of the Newsletter through our new Secretary.

In his moving from our ranks to fields of even greater achievement, Frank Wells leaves us with a well-knit organization. The magnitude of his contribution to our Team can be seen in the new applications we have discussed, attacked, and conquered, and in the continuing cohesive spirit of the Team. Frank was one of the original members of the Team, and has seen it grow from a fledgling group to the present membership of 55 under his guidance. We wish Frank continued success in his new responsibilities.

Sincerely,

Ed Orth Jr

Ed Orth
Team Secretary
Southern Services, Inc.
PO Box 2641
Birmingham, Alabama 35202

# EDUCATIONAL APPLICATIONS OF MARK SENSING

by RICHARD D. ROSS AND TONY A. ROSS
UNIVERSITY OF MISSISSIPPI
COMPUTER CENTER

# EDUCATIONAL APPLICATIONS OF MARK SENSING

by Richard D. Ross and Tony A. Ross
University of Mississippi
Computer Center

The mark sense attachment to the IBM 519 Reproducer has rapidly changed many aspects of the function of the Computer Center at the University of Mississippi. This attachment allows data to be mark sensed on IBM cards with a special graphite pencil and then in turn punched into the cards via the 519 Reproducer mark sense attachment.

Presently the University of Mississippi uses mark sense cards for:

(1) Scoring of objective exams
(2) Scoring all Student Counseling test given to incoming freshmen
(3) Used for completely automating test scoring and final grade output for Army R.O.T.C.
(4) Used for student attendance record keeping
(5) Used for Athletic Association ticket information

The first two items listed above will be discussed in detail in this paper and a brief mention of the last three will be given.

## University of Mississippi Test Scoring Program

The University of Mississippi Computer Center began scoring objective tests on the computer in 1963. The test scoring program that has been used was written by Robert M. O'Brien of Northeastern University, Massachusetts. Mr. O'Brien's program has been used extensively for the past three years although it was found that his program has certain limitations. These limitations are:

(1) Only 150 questions per test could be graded
(2) Only one correct answer per question was accepted
(3) Weighting of questions was not permissable
(4) Batch test grading was not permissable

All of these restrictions have been removed from the University of Mississippi Test Scoring Program (UMTS), and some additional features have been added.

Given here is the University of Mississippi's Test Scoring Program abstract as it will be sent to the 1620 Program Library in the immediate future.

## PROGRAM ABSTRACT

TITLE: University of Mississippi Test Scoring Program (UMTS)

AUTHOR: Richard D. Ross

DIRECT INQUIRIES TO: Richard D. Ross, Director
Computer Center, Carrier 103
University of Mississippi
University, Mississippi
phone: area code 601-232-8368

DESCRIPTION: UMTS is a flexible means of scoring objective exams taken on mark sense cards. It features a card output patterned after the Northeastern University Test Scoring Program by Robert M. O'Brien, Northeastern University, Mass. A numerical grade for each student is published along with a grade distribution (with mean and standard deviation) and an exam analysis--indicating how many choices per question were made and the percentage of correct answers per question. UMTS has a maximum range of 500 5-choice questions (10 cards) per exam with multiplicity of correct answers permitted. In addition, each question may be weighted with a weight value from 1 to 5.

UMTS allows for identification to be punched in columns 76-80 of each students grade card. This identification is taken from columns 14-18 of the control card. One of the most important features of UMTS is the speed of grading each students exam. Given below is speed of grading different tests:

| No. of Tests | No. of Questions | Time in Seconds |
|---|---|---|
| 100 | 50 | 93 |
| 100 | 100 | 120 |
| 100 | 150 | 155 |
| 100 | 200 | 190 |
| 100 | 300 | 260 |
| 100 | 400 | 330 |
| 100 | 500 | 400 |

As you can see from the table, to grade 150 questions takes approximately 1 1/2 seconds.

RESTRICTIONS/RANGE: No special instructions are required, although TNF and/or Direct Divide can be used on computers that have these capabilities. The maximum number of questions that may be graded is 150 questions for 20K computers and 500 questions for 40K computers.

## INPUT

1. Program Deck.

2. Control Card.

| Card Columns | Data |
|---|---|
| 1-2 | Number of test cards per student |
| 3-5 | Number of questions on the exam. |
| 6-8 | Number of questions not to be graded (this includes only those questions properly left blank). |
| 9 | "1" if the grade distribution and exam analysis by section is desired. Otherwise, a "0" or blank. |
| 10 | "1" if the grade distribution and exam analysis by all sections totaled together is desired. Otherwise, a "0" or blank. |
| 11 | "1" if grade distribution is desired on last card indicator. Otherwise, a "0" or blank. |
| 12 | "1" if name is to be omitted from output, otherwise a "0" or blank. |
| 14-18 | Any data in columns 14-18 of header card will be punched in columns 76-80 of each student's output card. This could be used to give the percent of the final grade that this test will be and the test number or any other identification that is needed. Another possible use for this output is to put the instructor's initials or in some four-letter cases, their last name. If left blank, nothing will be punched. |

3. Keys For The Exam.

The key cards for the exam are the same as the student answer cards. They are of three types: major keys, secondary keys, and weight cards.

A. MAJOR KEYS - Required
Contain the instructor's first choice of correct answers. It must contain an answer for each question to be graded. Questions not to be graded must be left blank.
Columns 1-5 have a 99999.

B. SECONDARY KEYS - Optional
Contain alternate answers to those given on the major keys. If a question on a secondary key card is left blank, no alternate answer is assumed. There can be 4 or less secondary key cards for each major key.
Columns 1-5 have a 99998 for first alternate key, 99997 for second, 99996 for third, and 99995 for the fourth alternate key.

C. WEIGHT KEYS - Optional
If used, the weight key will have a weight for each question answered on the major key. An answer A on the weight key assigns that question a weight of 1; a B, a weight of 2; a C, a weight of 3; a D, a weight of 4; an E, a weight of 5. If a question is left blank, the weight is assumed to be 1.
Columns 1-5 contain 99994.

Column 30 of ALL the key cards contains:
1, if the card pertains to the first 50 questions
2, if the card pertains to the second 50 questions
3, if the card pertains to the third 50 questions
and so on, until
9, if the card pertains to the ninth 50 questions
0, if the card pertains to the tenth 50 questions

Only one answer per question is allowed, but by using the alternate key cards, if the student answers any one of the correct answers he will get credit for that question. Let it be stressed that one and only one answer is to be marked per question.

If any of the alternate key cards or weight cards are not marked, they do not have to be read in, but if they are read in they are ignored.

The order by which the key cards are read in after the control card is of no consequence.

4. Student Answer Cards.

| Card Columns | Data |
|---|---|
| 1-5 | Student number |
| 6-23 | Student name |
| 24-25 | Section number |
| 26-29 | Course number |
| 30 | Card number |
| 31-80 | Student's answers |

The student's answer cards do not have to be in any particular order. The only requirement is that all cards for one student by read in together.

OUTPUT

1. Student s grade card

| Card Column | Data |
|---|---|
| 2-3 | Section number |
| 6-9 | Course number |
| 15-32 | Student's name |
| 39-43 | Student number |
| 49-51 | Number of correct answers |
| 57-59 | Number of incorrect answers |
| 65-67 | Number of questions omitted |
| 72-74 | Score |
| 76-80 | Any data in columns 14-18 of the control card |

2. Grade distribution cards.

| Card Columns | Data |
|---|---|
| 2-3 | Section number |
| 6-9 | Course number |
| 35-37 | Score |
| 48-50 | Frequency |
| 61-63 | Cumulative frequency |
| 74-76 | Percentile |

3. Exam Analysis Cards.

| Card Columns | Data |
|---|---|
| 2-3 | Section number |
| 6-9 | Course number |
| 14-16 | Question number |
| 23-26 | Number of A answers |
| 32-35 | Number of B answers |
| 41-44 | Number of C answers |
| 50-53 | Number of D answers |
| 59-62 | Number of E answers |
| 68-71 | Number of omissions |
| 78-80 | Percent of correct answers to this question |

## STUDENT COUNSELING CENTER TEST GRADING

The Student Counseling Center has converted all of their test grading to mark sense IBM cards. This has saved them much time and effort in giving pre-college entrance tests and getting the results as soon as possible after the last test is given. There were three programs written for the Student Counseling Center to produce the desired results. These programs are named:

(1) ACT AND MASTER CARD PROGRAM
(2) TEST CARD PREPARATION PROGRAM
(3) TEST SCORING STUDENT COUNSELING PROGRAM

The Student Counseling Center gives a battery of tests and they are:

(1) Diagnostic Reading Test, form A, F, or H.
(2) Nelson-Denny Reading Test
(3) Abstract Reasoning
(4) Edwards
(5) Math Test
(6) Strong Entrance Test

The above tests range from 50 question tests to 225 question tests for the Edwards and to 400 question tests for the Strong Entrance Test.

Given here is the outline of the procedure for preparing the tests to be given and a sample of the results.

Step 1. A master card should be made for every student using the following format:

| | | |
|---|---|---|
| Col. | 01-05 | Alpha Number |
| Col. | 06-23 | Name |
| Col. | 26 | Sex |
| Col. | 31-32 | Age |
| Col. | 34-35 | Classification |
| Col. | 38-50 | Street Address |
| Col. | 52-64 | Home Town |
| Col. | 66-78 | State |
| Col. | 80 | Asterisk |

The two digit classification in Cols. 34-35 will be as follows:

| | |
|---|---|
| 01 | Pre-College |
| 02 | Freshman |
| 03 | Sophmore |
| 04 | Junior |
| 05 | Senior |
| 06 | Transfer (Year 1) |
| 07 | Transfer (Year 2) |
| 08 | Transfer (Year 3) |
| 09 | Transfer (Year 4) |
| 10 | Graduate (Year 1) |
| 11 | Graduate (Year 2) |
| 12 | Graduate (Year 3) |
| 13 | Graduate (Year 4) |
| 14 | Liberal Arts |
| 15 | Business and Government |
| 16 | Engineering |
| 17 | Pre-Medicine |
| 18 | Pre-Pharmacy |
| 19 | Education |

If an ACT card is available for the students, a computer program labeled ACT AND MASTER CARD PROGRAM is available to prepare the master card and the ACT card that will be used in the test grading program later. The master card will have all of the information in the correct Cols. with the exception of age which is in Cols. 31-32 and this will have to be punched in by hand. The classification is assumed to be 01 for pre-college students. After the master cards and the ACT cards are prepared from the original ACT cards, the output is then sorted on Col. 80. The ACT cards will fall in the first pocket of the sorter and the master cards will fall in pocket eight of the sorter.

Step 2. After the master card has been prepared either by the computer program or manually, the program labeled TEST CARD PREPARATION is now loaded into the computer to prepare the mark-sense cards for the DRT, Nelson-Denny, Abstract, Edwards, and the Math Test. After the program has been loaded into the computer, it will type the message "READ IN MASTER CARDS" and at this time, read in the master cards that have been prepared in Step 1. After the master cards have been read in, the computer will then type out the message "ENTER NO. 01 CARDS" and at this time you will place in the punch hopper of the 1622 No. 1 mark-sense cards. Press start on the 1620 Console and punch start on the 1622. After it has punched all of the No. 01 cards necessary, it will then type out the message "ENTER NO. 02 CARDS" and at this time you will clear the punch hopper and continue this procedure until you have completed punching the No. 05 mark-sense cards. If an error occurs and you want to begin again, press RESET-INSERT-RELEASE and START on the 1620 and the computer will type "ENTER NO. 01 CARDS."

If an error occurs while punching a particular set of cards and you only want to begin on this set again, turn switch-4 on and press RESET-INSERT-RELEASE- and START, then turn switch-4 off.

Step 3. These cards are now interpreted on the 548 Interpreter.

Step 4. After all cards have been interpreted, they are sorted on the Alpha Number, Cols. 5, 4, 3, 2, 1, and the test number which is in Col. 25. In summary, you will sort on Cols. 5, 4, 3, 2, 1, and 25.

Step 5. The cards are now in order according to the tests that are given and the number of the test will be interpreted on the mark-sense card and will appear in the block labeled Section Number. For the DRT test, there will be an alphabetic letter A, F, or H for the form of test given. Be sure that this letter agrees with the form of test that is being given.

Step 6. After all tests have been given, they will now be mark sensed using the 519 Reproducer. After the cards have been mark-sensed, take all of the DRT tests on which the student should have marked the line number that he was on during the reading part of the DRT test. This number will now be punched manually as a three digit number in Cols. 27, 28, and 29 of the No. 1 and No. 2 cards of each person. The same number should appear on both cards, if not, an error message will be typed later. Some number has to appear in these Cols. even if it is 0; hence, if the student did not give the line number, then you should enter 000 in Cols. 27, 28, and 29.

Step 7. After the cards have been mark-sensed, the master cards followed by the ACT cards that were previously punched are placed in front of all of the tests that have been given. All of these cards are now sorted on Cols. 30, 25, 5, 4, 3, 2, and 1. These cards are now ready for grading.

Step 8. Load in the program labeled TEST SCORING STUDENT COUNSELING CENTER followed by all description headers, key cards, and weight decks. After all of these have been read in, the message "READY TO GRADE TESTS" will be typed on the typewriter. Place the sorted deck in the read hopper and read in the tests and in turn, the correct answers will be punched. If a check stop occurs, this may indicate that a card has invalid characters punched on it or it may mean that sorting was incorrect. If any error messages are typed out, this will indicate that some of the test cards were prepared incorrectly.

Step 9. The answers punched are now printed on the 407 using the standard board with switch-1 on. This board is wired to skip to a new page if there is a nine (9) in column "1" of a card providing switch-1 is on. Two copies of everything will be printed--one for the student and one for the Student Counseling Center.

Step 10. The format for all cards is given on the supplimentary page. If there is a student who appeared at the test late and there are no master cards or test cards made for this student, the student will be given the correct card numbers for each of his tests, and he will sign his name across the top of the card under the line marked signature. These cards will then have to have the correct information punched in them according to the format on the supplimentary sheet. This information is punched in the card before the cards are mark-sensed. Also a master card will have to be punched for each of these and an Alpha Number given to them. This number can not be the same as any other Alpha Number given for this test. The same Alpha Number has to appear on the person's master card, his ACT card, and all tests that he has taken.

To be able to grade all of these tests on the computer, a set of answer cards and percentile cards have to be read in for each test. In case of the Diagnostic Reading Test answer cards and percentile cards have to be read in for the three different forms-A, H, and F. For some of the tests the men and women have different percentiles and these also have to be read in as tables. Shown on the next five pages are the description headers, key cards, weight cards, and percentile cards that are read in as part of the input data to the TEST SCORING STUDENT COUNSELING PROGRAM. This data is now followed by the student's answer cards and the results are produced.

Shown also is a sample input for the Strong Entrance Test and a sample output.

```
          DESCRIPTIONS HEADERS - MASTER KEY CARDS - PERCENTILES
 01 PRE-COLLEGE
 02 FRESHMAN
 03 SOPHMORE
 04 JUNIOR
 05 SENIOR
 06 TRANSFER (YEAR 1)
 07 TRANSFER (YEAR 2)
 08 TRANSFER (YEAR 3)
 09 TRANSFER (YEAR 4)
 10 GRADUATE (YEAR 1)
 11 GRADUATE (YEAR 2)
 12 GRADUATE (YEAR 3)
 13 GRADUATE (YEAR 4)
 14 LIBERAL ARTS
 15 BUSINESS AND GOVERNMENT
 16 ENGINEERING
 17 PRE-MEDICINE
 18 PRE-PHARMACY
-19 EDUCATION
 01       NORMS     ACT
 02       U.S.       ENGLISH
 03       COLLEGE    MATHEMATICS
 04       BOUND      SOCIAL STUDIES
 05                  NATURAL SCIENCE
 06                  COMPOSITE
 07                                   R  P
 08                                    S  R
 09       UNIV OF   DRT
 10       MISS       RATE
 11       1959-1962  VOCABULARY
 12       FORM ( )   COMPREHENSION
 13
 14       NATL 1960 NELSON-DENNY
 15       FRESHMAN   VOCABULARY
 16
 17
 18       U.S. 12TH DAT
 19       GRADE      ABSTRACT R.
 20       M-F
 21
 22
 23       M-F       EDWARDS
 24                   1
 25                   2
 26                   3
 27                   4
 28                   5
 29                   6
 30                   7
 31                   8
 32                   9
 33                  10
 34                  11
 35                  12
 36                  13
 37                  14
 38                    15
 39                    16
 40
 41        1959      UNIV. OF MISS.
 42                   MATH TEST
 43
 44        U.S. 12TH DAT
 45        GRADE M-F  SPACE RELATIONS
 46
 47        UNIV OF   DRT
 48        MISS       RATE
 49        1959-1962  VOCABULARY
-50        FORM ( )   COMPREHENSION
0135 0237 0339 0440 0541 0642 0743 0844 0944 1045          PERCENTILES       01
1146 1246 1346 1447 1547 1648 1748 1848 1949 2049          PERCENTILES       02
2149 2250 2350 2451 2551 2652 2752 2852 2953 3053          PERCENTILES       03
3154 3253 3354 3454 3554 3654 3755 3855 3955 4055          PERCENTILES       04
4156 4256 4356 4456 4556 4656 4757 4857 4957 5057          PERCENTILES       05
5158 5258 5358 5458 5559 5659 5759 5859 5960 6060          PERCENTILES       06
6160 6260 6361 6461 6561 6661 6761 6861 6962 7062          PERCENTILES       07
7163 7263 7363 7464 7564 7665 7765 7866 7966 8066          PERCENTILES       08
8167 8267 8367 8468 8568 8668 8769 8869 8970 9070          PERCENTILES       09
9171 9271 9373 9474 9574 9675 9777 9879 9980 0035          PERCENTILES       10
 01 01 02
 01 02 03
 01 03 04
 01 04 05
 01 05 06
 02 01 10
 02 02 11
 02 03 12
 03 01 15
 04 01 19
 05 01 24
 05 02 25
 05 03 26
 05 04 27
 05 05 28
 05 06 29
 05 07 30
 05 08 31
 05 09 32
 05 10 33
 05 11 34
 05 12 35
 05 13 36
 05 14 37
 05 15 38
 05 16 39
-06 01 42
```

```
          DESCRIPTIONS HEADERS - MASTER KEY CARDS - PERCENTILES
99999DIAGNOSTIC KEY F-AA2    186896897678796869878889763104231224444021324322303
99999DIAGNOSTIC KEY F-AA2    285655859795678897766957654400024141124343421241234
99999DIAGNOSTIC KEY F-FF2    187867876667886676778987873010321101101341331114444
99999DIAGNOSTIC KEY F-FF2    268887795569757685686875992040013131311442421223311
99999DIAGNOSTIC KEY F-HH2    178698867767786789799686883332130101414300412112024
99999DIAGNOSTIC KEY F-HH2    297767656585796895656756793213024141314132132443314
00004 00009 00013 00017 00022 00026 00030 00035 00039 00043                R2 01
00048 00052 00056 00061 00065 00069 00074 00078 00082 00087                R2 02
00091 00095 00100 00104 00108 00113 00117 00121 00126 00130                R2 03
00134 00139 00143 00147 00152 01156 01160 02165 02169 03173                R2 04
04178 05182 06186 07191 09195 11199 12204 14208 16212 20217                R2 05
22221 24225 26230 30234 33238 36243 39247 41251 44256 47260                R2 06
49264 51269 55273 57277 61282 63286 64290 67295 69299 72303                R2 07
74308 76312 77316 79321 81325 83329 85334 86338 87342 88347                R2 08
90351 90355 91360 92364 93368 93373 93377 94381 94386 95390                R2 09
95394 96399 96403 96407 97412 97416 97420 97425 98429 98433                R2 10
98438 98442 98446 98451 98455 99459 99464 99468 99472 99477                R2 11
99481 99485 99490 99494 99498 99503 99507 99511 99516 99520                R2 12
99524 99529 99533 99537 99542 99546 99550 99555 99559 99563                R2 13
00 00 00 0-                                               1                V2 01
01 02 02 02 03 03 04 06 07 09 11 13 15 18 21 24 27 30 34 38                V2 02
42 45 50 55 59 63 67 71 76 80 85 89 93 96 97 99 99 99 99 99                V2 03
00 00 00 0                         1 02 02 03 03 05 07 09 11                C2 01
14 18 22 27 33 38 44 50 57 63 70 78 84 89 94 97 99 99 99 99                C2 02
99999NELSON-DENNY KEY   3    187686588578765559965985594324100310442423111312102
99999NELSON-DENNY KEY   3    279886569785855796567655982314323242422413031421334
00 00 00 00  *     1 02 03 03 03 04 05 06 07 09 11 12 14 16                V3 01
18 20 22 24 26 28 31 34 37 39 42 44 47 50 52 55 57 59 61 63                V3 02
65 67 69 71 73 75 77 79 80 82 83 86 87 88 89 90 91 92 92 93                V3 03
94 94 95 95 96 96 97 97 97 98 98 98 99 99 99 99 99 99 99 99                V3 04
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                V3 05
99999ABSTRACT KEY       4    159868565987789767989876561023204211313414244310441
0103 0303 0305 0305 0305 0305 0305 0305 0510 0510                          P4 01
0510 0510 0510 0510 1015 1015 1015 1015 1020 1520                          P4 02
1520 1525 2025 2030 2530 2535 3035 3040 3545 3545                          P4 03
4050 4555 5060 5560 6065 6570 7075 7580 8085 8585                          P4 04
8590 9095 9095 9597 9799 9999 9999 9999 9999 9999                          P4 05
006A 011A 016A 021A 026A 031A 036A 041A 046A 051A 056A 061A 066A 071A      05 01
002B 003B 004B 005B 151B 152B 153B 154B 155B 076B 077B 078B 079B 080B      05 02
002A 012A 017A 022A 027A 032A 037A 042A 047A 052A 057A 062A 067A 072A      05 03
006B 008B 009B 010B 156B 157B 158B 159B 160B 081B 082B 083B 084B 085B      05 04
003A 008A 018A 023A 028A 033A 038A 043A 048A 053A 058A 063A 068A 073A      05 05
011B 012B 014B 015B 161B 162B 163B 164B 165B 086B 087B 088B 089B 090B      05 06
004A 009A 014A 024A 029A 034A 039A 044A 049A 054A 059A 064A 069A 074A      05 07
016B 017B 018B 020B 166B 167B 168B 169B 170B 091B 092B 093B 094B 095B      05 08
005A 010A 015A 020A 030A 035A 040A 045A 050A 055A 060A 065A 070A 075A      05 09
021B 022B 023B 024B 171B 172B 173B 174B 175B 096B 097B 098B 099B 100B      05 10
076A 081A 086A 091A 096A 106A 111A 116A 121A 126A 131A 136A 141A 146A      05 11
026B 027B 028B 029B 030B 176B 177B 178B 179B 180B 102B 103B 104B 105B      05 12
077A 082A 087A 092A 097A 102A 112A 117A 122A 127A 132A 137A 142A 147A      05 13
031B 032B 033B 034B 035B 181B 182B 183B 184B 185B 106B 108B 109B 110B      05 14
078A 083A 088A 093A 098A 103A 108A 118A 123A 128A 133A 138A 143A 148A      05 15
036B 037B 038B 039B 040B 186B 187B 188B 189B 190B 111B 112B 114B 115B      05 16
079A 084A 089A 094A 099A 104A 109A 114A 124A 129A 134A 139A 144A 149A      05 17
041B 042B 043B 044B 045B 191B 192B 193B 194B 195B 116B 117B 117B 120B      05 18
```

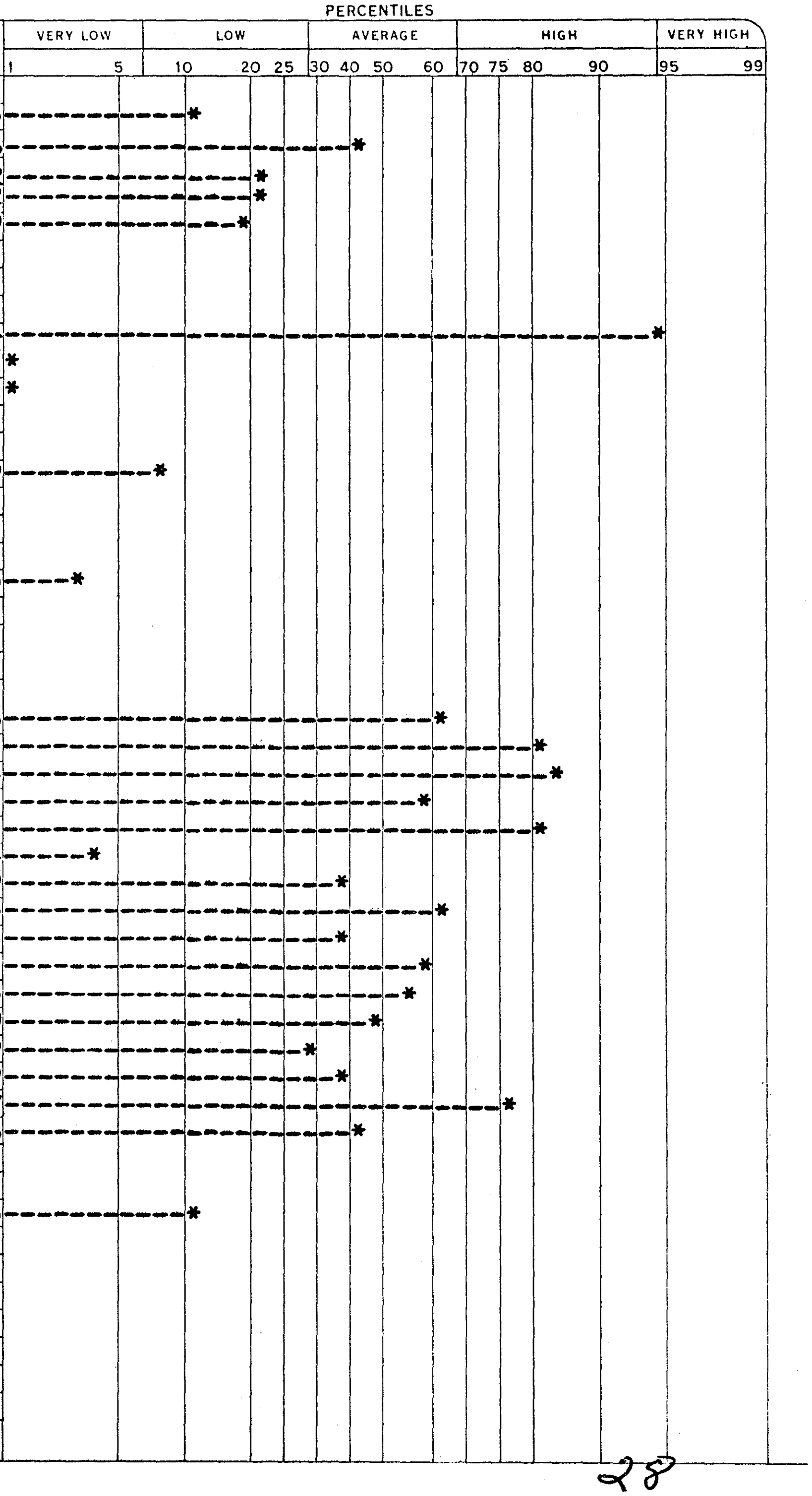
```
080A 085A 090A 095A 100A 105A 110A 115A 120A 130A 135A 140A 145A 150A      05 19
046B 047B 048B 049B 050B 196B 197B 198B 199B 200B 121B 122B 123B 124B      05 20
151A 156A 161A 166A 171A 176A 181A 186A 191A 196A 206A 211A 216A 221A      05 21
051B 052B 053B 054B 055B 202B 203B 204B 205B 126B 127B 128B 129B 130B      05 22
152A 157A 162A 167A 172A 177A 182A 187A 192A 197A 202A 212A 217A 222A      05 23
056B 057B 058B 059B 060B 206B 208B 209B 210B 131B 132B 133B 134B 135B      05 24
153A 158A 163A 168A 173A 178A 183A 188A 193A 198A 203A 208A 218A 223A      05 25
061B 062B 063B 064B 065B 211B 212B 214B 215B 136B 137B 138B 139B 140B      05 26
154A 159A 164A 169A 174A 179A 184A 189A 194A 199A 204A 209A 214A 224A      05 27
066B 067B 068B 069B 070B 216B 217B 218B 220B 141B 142B 143B 144B 145B      05 28
155A 160A 165A 170A 175A 180A 185A 190A 195A 200A 205A 210A 215A 220A      05 29
071B 072B 073B 074B 075B 221B 222B 223B 224B 146B 147B 148B 149B 150B      05 30
101B 026B 107B 032B 113B 038B 119B 044B 125B 050B 051A 201A 057A 207A 063A 05 31
213A 069A 219A 075A 225A 001A 151A 007A 157A 013A 163A 019A 169A 025A 175A 05 32
00 00 00 0                                                                 M5 01
00 00 01 00 00 00 00 01 00 01 00 00 00 00 0                                M5 02
00 00 03 00 00 00 00 02 00 02 00 00 01 00 01 00                            M5 03
00 01 06 00 00 00 00 05 00 03 01 00 03 00 02 00                            M5 04
00 03 09 00 01 01 00 09 01 05 02 01 06 01 03 00                            M5 05
00 06 14 00 02 02 01 13 02 09 03 02 10 02 06 00                            M5 06
01 10 19 01 03 03 04 20 02 12 06 03 13 03 09 01                            M5 07
02 16 27 03 06 04 05 27 04 18 10 05 18 05 13 03                            M5 08
04 23 36 06 11 07 09 34 05 24 13 07 24 06 17 06                            M5 09
07 34 46 09 15 11 13 42 08 31 18 10 29 09 24 15                            M5 10
10 43 54 15 22 16 17 50 09 38 24 15 37 12 32 27                            M5 11
16 52 63 21 28 21 19 58 13 45 31 20 44 14 40 46                            M5 12
22 63 71 28 34 28 25 65 17 53 37 28 51 19 47 68                            M5 13
30 73 78 37 43 36 32 72 21 61 46 33 57 23 57 86                            M5 14
40 81 84 49 52 45 39 78 27 67 55 41 64 27 65 96                            M5 15
50 88 88 62 61 54 45 83 32 75 63 50 69 32 72 99                            M5 16
58 93 92 72 68 64 51 87 38 79 70 57 75 39 77 99                            M5 17
66 96 95 81 76 72 57 90 45 84 75 64 79 45 84 99                            M5 18
74 98 96 89 82 79 64 93 54 88 82 71 85 53 88 66                            M5 19
83 99 97 93 86 84 72 95 63 92 86 78 89 59 92 99                            M5 20
86 99 99 97 91 89 78 97 73 94 90 84 92 67 95 66                            M5 21
91 99 99 98 94 93 84 99 79 97 94 89 94 73 98 99                            M5 22
95 99 99 99 96 96 89 99 85 99 96 93 96 81 98 99                            M5 23
98 99 99 99 98 98 93 99 91 99 98 97 98 89 99 99                            M5 24
99 99 99 99 99 99 95 99 94 99 99 98 99 90 99 99                            M5 25
99 99 99 99 99 99 97 99 98 99 99 99 99 93 99 99                            M5 26
99 99 99 99 99 99 99 99 99 99 99 99 99 96 99 99                            M5 27
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                            M5 28
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                            M5 29
00 00 00 0                                                                 F5 30
00 00 01 00 00 00 0                        1 00                            F5 31
00 00 03 00 00 00 0*                 1 01 02 00                            F5 32
01 01 05 00 01 00 00 01 01 00 00 00 03 02 05 00                            F5 33
02 02 09 00 03 00 00 03 02 01 00 00 05 03 09 00                            F5 34
03 03 14 00 06 00 00 06 03 02 01 01 09 05 14 00                            F5 35
05 06 20 01 09 00 00 09 05 04 02 02 13 08 21 01                            F5 36
08 09 28 03 13 01 01 13 09 07 02 03 18 11 27 02                            F5 37
13 14 37 06 20 02 03 18 11 11 03 04 24 15 36 04                            F5 38
19 21 46 10 28 03 05 25 17 14 06 07 30 19 44 44                            F5 39
```

```
27 29 55 15 36 04 09 33 23 19 09 10 36 24 51 24                        F5 40
36 41 64 21 45 07 12 44 28 25 14 13 44 31 60 43                        F5 41
47 50 72 30 53 12 16 51 36 31 20 17 50 38 66 63                        F5 42
58 62 77 42 62 17 22 59 42 37 26 21 57 45 74 83                        F5 43
64 73 83 52 70 24 28 67 51 45 32 28 63 52 79 96                        F5 44
72 80 88 63 77 32 35 76 59 51 41 35 69 57 84 99                        F5 45
79 85 91 73 82 41 43 81 67 58 50 42 76 63 89 99                        F5 46
84 91 94 82 87 50 49 86 74 66 58 52 80 69 92 99                        F5 47
90 95 96 88 92 58 58 90 83 74 66 59 85 76 95 99                        F5 48
93 98 97 92 96 68 66 93 87 81 74 68 90 81 97 99                        F5 49
96 99 98 96 97 77 73 96 92 86 82 73 93 87 98 99                        F5 50
97 99 99 97 98 84 80 98 95 90 87 81 95 91 98 99                        F5 51
98 99 99 99 99 90 85 99 98 93 91 85 97 94 99 99                        F5 52
99 99 99 99 99 94 90 99 99 96 95 90 99 96 99 99                        F5 53
99 99 99 99 99 96 94 99 99 98 97 94 99 97 99 99                        F5 54
99 99 99 99 99 98 97 99 99 99 99 96 99 98 99 99                        F5 55
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                        F5 56
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                        F5 57
99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99                        F5 58
00001MATH KEY          06    199998888889575856665966750124242100230034001432024
00001MATH KEY          06     29659668596
00 00 00 0                  2 04 06 08 11 13 16 19 23 27 30            M   01
34 37 41 44 49 52 56 60 63 66 68 71 73 76 78 80 82 83 85 86            M   02
87 88 90 92 93 93 94 95 96 96 97 97 98 99 99 99 99 99 99 99            M   03
```

# UNIVERSITY OF MISSISSIPPI
## STUDENT COUNSELING CENTER

| NAME | AGE | SEX | HOME TOWN | STATE | CLASSIFICATION |
|---|---|---|---|---|---|
| ROSS TONY | 26 | M | UNIVERSITY | MISSISSIPPI | PRE-COLLEGE |

| NORMS | DESCRIPTION | S S | P R |
|---|---|---|---|
| NORMS | ACT | | |
| U.S. | ENGLISH | 14 | 13 |
| COLLEGE | MATHEMATICS | 19 | 46 |
| BOUND | SOCIAL STUDIES | 16 | 22 |
| | NATURAL SCIENCE | 16 | 22 |
| | COMPOSITE | 16 | 19 |
| | | R | P |
| | | S | R |
| UNIV OF | DRT | | |
| MISS | RATE | 386 | 94 |
| 1959-1962 | VOCABULARY | 12 | 1 |
| FORM ( ) | COMPREHENSION | 12 | 1 |
| | | | |
| NATL 1960 | NELSON-DENNY | | |
| FRESHMAN | VOCABULARY | 16 | 9 |
| | | | |
| | | | |
| U.S. 12TH | DAT | | |
| GRADE | ABSTRACT R. | 2 | 3 |
| M-F | | | |
| | | | |
| | | | |
| M-F | EDWARDS | | |
| | 1 | 17 | 66 |
| | 2 | 14 | 81 |
| | 3 | 14 | 84 |
| | 4 | 15 | 62 |
| | 5 | 18 | 82 |
| | 6 | 7 | 4 |
| | 7 | 14 | 39 |
| | 8 | 12 | 65 |
| | 9 | 16 | 38 |
| | 10 | 13 | 61 |
| | 11 | 14 | 55 |
| | 12 | 15 | 50 |
| | 13 | 9 | 29 |
| | 14 | 16 | 39 |
| | 15 | 16 | 77 |
| | 16 | 11 | 46 |
| | | | |
| 1959 | UNIV. OF MISS. | | |
| | MATH TEST | 15 | 13 |
| | | | |
| U.S. 12TH | DAT | | |
| GRADE M-F | SPACE RELATIONS | | |
| | | | |
| UNIV OF | DRT | | |
| MISS | RATE | | |
| 1959-1962 | VOCABULARY | | |
| FORM ( ) | COMPREHENSION | | |



28

# UNIVERSITY OF MISSISSIPPI
## STUDENT COUNSELING CENTER

| NAME | AGE | SEX | HOME TOWN | STATE | CLASSIFICATION |
|---|---|---|---|---|---|
| ROSS RICHARD | 23 | M | UNIVERSITY | MISSISSIPPI | GRADUATE (YEAR 1) |

| NORMS | DESCRIPTION | S<br>S | P<br>R | PERCENTILES<br>VERY LOW · LOW · AVERAGE · HIGH · VERY HIGH<br>1 5 10 20 25 30 40 50 60 70 75 80 90 95 99 |
|---|---|---|---|---|
| NORMS | ACT | | | |
| U.S. | ENGLISH | 17 | 27 | ----------------* |
| COLLEGE | MATHEMATICS | 27 | 85 | ----------------------------* |
| BOUND | SOCIAL STUDIES | 17 | 28 | ----------------* |
| | NATURAL SCIENCE | 17 | 27 | ----------------* |
| | COMPOSITE | 20 | 46 | -------------------* |
| | | R | P | |
| | | S | R | |
| UNIV OF | DRT | | | |
| MISS | RATE | 334 | 85 | ----------------------------* |
| 1959-1962 | VOCABULARY | 11 | 1 | * |
| FORM ( ) | COMPREHENSION | 10 | 1 | * |
| | | | | |
| NATL 1960 | NELSON-DENNY | | | |
| FRESHMAN | VOCABULARY | 18 | 12 | -----------* |
| | | | | |
| | | | | |
| U.S. 12TH | DAT | | | |
| GRADE | ABSTRACT R. | 3 | 3 | ----* |
| M-F | | | | |
| | | | | |
| | | | | |
| M-F | EDWARDS | | | |
| | 1 | 15 | 50 | --------------------* |
| | 2 | 17 | 96 | ---------------------------------* |
| | 3 | 16 | 92 | ------------------------------* |
| | 4 | 7 | 3 | ----* |
| | 5 | 19 | 86 | ----------------------------* |
| | 6 | 15 | 54 | ---------------------* |
| | 7 | 13 | 32 | -----------------* |
| | 8 | 14 | 78 | --------------------------* |
| | 9 | 15 | 32 | -----------------* |
| | 10 | 13 | 61 | -----------------------* |
| | 11 | 15 | 63 | ------------------------* |
| | 12 | 16 | 57 | ----------------------* |
| | 13 | 11 | 44 | -------------------* |
| | 14 | 14 | 27 | ----------------* |
| | 15 | 10 | 32 | -----------------* |
| | 16 | 8 | 6 | -------* |
| | | | | |
| 1959 | UNIV. OF MISS. | | | |
| | MATH TEST | 12 | 6 | -------* |
| | | | | |
| U.S. 12TH | DAT | | | |
| GRADE M-F | SPACE RELATIONS | | | |
| | | | | |
| UNIV OF | DRT | | | |
| MISS | RATE | | | |
| 1959-1962 | VOCABULARY | | | |
| FORM ( ) | COMPREHENSION | | | |

# UNIVERSITY OF MISSISSIPPI
## STUDENT COUNSELING CENTER

| NAME | AGE | SEX | HOME TOWN | STATE | CLASSIFICATION |
|---|---|---|---|---|---|
| ROSS RICHARD D | 26 | M | BATESVILLE | MISSISSIPPI | GRADUATE (YEAR 1) |

### STRONG VOCATIONAL INTEREST TEST PERCENTILES

| GROUP | OCCUPATION | S S | G R | C (0–30) B– (30–35) B (35–40) B+ (40–45) A (45–65)<br>0 5 10 15 20 25 30 35 40 45 50 55 60 65 |
|---|---|---|---|---|
| GENERAL | ARTIST | 13 | C | C     ***** |
| PROFESS | PSYCHOLOGIST | 18 | C | C****** |
| | ARCHITECT | 10 | C | C  ******* |
| | PHYSICIAN | 27 | C+ | *******C+ |
| | PSYCHIATRIST | 22 | C+ | **C+*** |
| | OSTEOPATH | 35 | B– | ******--B– |
| | DENTIST | 28 | C+ | ******C+ |
| | VETERINARIAN | 45 | B+ | ******---------------B+ |
| TECHNICAL | BIOLOGICAL SCIENCE | 12 | C | C |
| PROFESS | EXPERIMENTAL PSCH | 38 | B | B |
| | CHEMIST | 24 | C+ | ***C+*** |
| | PHYSICIST | 16 | C | ***C*** |
| | MATHEMATICIAN | 8 | C– | C–  ****** |
| | ENGINEER | 31 | B– | ********B– |
| APPLIED | FARMER | 46 | A– | *****----A– |
| TECHNICAL | FORESTER | 35 | B– | *******-----------B– |
| | MATH SCI TEACHER | 31 | B– | ***B– |
| | | | | |
| SERVICE | HS SOC SCI TEACHER | 21 | C+ | C+  ****** |
| | SOCIAL WORKER | 17 | C | C |
| | CLINICAL PSYCH | 31 | B– | B– |
| | VOC COUNSELOR | 23 | C+ | C+****** |
| | YMCA PHYS DIRECTOR | 25 | C+ | *****C+ |
| | PERSONNEL MANAGER | 15 | C | C ******** |
| | IND RELATIONS | 55 | A | A |
| | PUBLIC ADMN | 24 | C+ | C+****** |
| | YMCA SECRETARY | 17 | C | C****** |
| | CITY SCHOOL SUPT | 11 | C | C******* |
| DETAILED | CPA PARTNER | 12 | C | C  ******* |
| BUSINESS | SENIOR CPA | 26 | C+ | **C+** |
| | ACCOUNTANT | 18 | C | C******* |
| | OFFICE WORKER | 27 | C+ | C+ ****** |
| | PURCHASING AGENT | 41 | B+ | *******------B+ |
| | BANKER | 33 | B– | **B–*** |
| | PHARMACIST | 40 | B+ | *********---B+ |
| | | | | |
| SALES | SALES MANAGER | 39 | B | ******------B |
| | LIFE INS SALESMAN | 39 | B | *****--B |
| | REALTOR | 49 | A– | ***-----A– |
| VERBAL | ADVERTISING | 21 | C+ | C+     ***** |
| | LAWYER | 25 | C+ | C+******* |
| | AUTHOR JOURNALIST | 20 | C+ | C+     *** |
| MUSIC | MUSIC PERFORMER | 19 | C | C |
| | MUSIC TEACHER | 9 | C– | C– |
| | | | | |
| MISCELL | PRODUCTION MANAGER | 41 | B+ | *********----B+ |
| | ARMY OFFICER | 26 | C+ | ******----C+ |
| | PRESIDENT MFG CO | 41 | B+ | ******---B+ |
| | INTEREST MATURITY | 49 | A– | A– |
| | | | | |

30

# UNIVERSITY OF MISSISSIPPI
## STUDENT COUNSELING CENTER

| NAME | AGE | SEX | HOME TOWN | STATE | CLASSIFICATION |
|---|---|---|---|---|---|
| ROSS TONY A | 20 | M | KAPPA ALPHA H | MISSISSIPPI | PRE-COLLEGE |

### STRONG VOCATIONAL INTEREST TEST PERCENTILES

| GROUP | OCCUPATION | S S | G R | C (0 5 10 15 20 25 30) B− (30 35) B (35 40) B+ (40 45) A (45 50 55 60 65) |
|---|---|---|---|---|
| GENERAL | ARTIST | 23 | C+ | C+***** |
| PROFESS | PSYCHOLOGIST | 20 | C+ | C+****** |
| | ARCHITECT | 12 | C | C  ******* |
| | PHYSICIAN | 23 | C+ | ****C+* |
| | PSYCHIATRIST | 27 | C+ | ******C+ |
| | OSTEOPATH | 31 | B- | *****B- |
| | DENTIST | 15 | C | C ******** |
| | VETERINARIAN | 20 | C+ | *****C+ |
| TECHNICAL | BIOLOGICAL SCIENCE | 2 | C- | C- |
| PROFESS | EXPERIMENTAL PSCH | 26 | C+ | C+ |
| | CHEMIST | 6 | C- | C-    ******** |
| | PHYSICIST | 16 | C | ***C*** |
| | MATHEMATICIAN | 8 | C- | C-   ****** |
| | ENGINEER | 10 | C | C    ******** |
| APPLIED | FARMER | 16 | C | C       ***** |
| TECHNICAL | FORESTER | 8 | C- | C-****** |
| | MATH SCI TEACHER | 13 | C | C     ***** |
| | | | | |
| SERVICE | HS SOC SCI TEACHER | 38 | B | ******----B |
| | SOCIAL WORKER | 37 | B | B |
| | CLINICAL PSYCH | 47 | A- | A- |
| | VOC COUNSELOR | 34 | B- | *****B- |
| | YMCA PHYS DIRECTOR | 28 | C+ | *******-C+ |
| | PERSONNEL MANAGER | 28 | C+ | *****C+* |
| | IND RELATIONS | 45 | B+ | B+ |
| | PUBLIC ADMN | 40 | B+ | ********----B+ |
| | YMCA SECRETARY | 25 | C+ | ****C+* |
| | CITY SCHOOL SUPT | 27 | C+ | *********---C+ |
| DETAILED | CPA PARTNER | 32 | B- | *******--B- |
| BUSINESS | SENIOR CPA | 26 | C+ | **C+** |
| | ACCOUNTANT | 17 | C | C ******* |
| | OFFICE WORKER | 30 | B- | B-***** |
| | PURCHASING AGENT | 25 | C+ | *C+**** |
| | BANKER | 29 | C+ | C+****** |
| | PHARMACIST | 28 | C+ | **C+**** |
| | | | | |
| SALES | SALES MANAGER | 51 | A | ******---------------A |
| | LIFE INS SALESMAN | 59 | A | *****------------------A |
| | REALTOR | 56 | A | ***------------A |
| VERBAL | ADVERTISING | 44 | B+ | *****---B+ |
| | LAWYER | 57 | A | ******------------------A |
| | AUTHOR JOURNALIST | 39 | B | ***-B |
| MUSIC | MUSIC PERFORMER | 29 | C+ | C+ |
| | MUSIC TEACHER | 27 | C+ | C+ |
| | | | | |
| MISCELL | PRODUCTION MANAGER | 25 | C+ | *C+***** |
| | ARMY OFFICER | 26 | C+ | ******----C+ |
| | PRESIDENT MFG CO | 32 | B- | ***B-* |
| | INTEREST MATURITY | 50 | A | A |
| | | | | |

31

## ARMY ROTC RECORD KEEPING

The Army R.O.T.C. is now keeping all class records such as merits, demerits, absences, excused absences, and essay scores on mark-sense cards. They also have a unit card on which information about each person is mark sensed. They have four mark sense cards and are labeled:

(1) Unit Card (U-Card)
(2) Essay Card (E-Card)
(3) Absentee Card (A-Card)
(4) Merit and Demerit Card (M-Card)

Shown on the preceeding page is a picture of the four cards that are used. Careful examination of these cards will show that one is able to keep all significant data on these IBM cards and at the end of each semester these cards are used to determine the final grade of each student. For the four different levels of Military Science-1, 2, 3, and 4- the approximate time to furnish final grades is approximately one hour.

### Student Attendance Record Keeping And Athletic Association Ticket Information

Also shown on the following page is a picture of the attendance record keeping mark-sense card and the athletic association ticket information card that is used at the University of Mississippi for attendance record keeping for high schools and for mailing ticket information to football ticket buyers for the coming year. Due to the lenght of the paper already and the time allotted for giving the paper, these two items can not be discussed. Further information about these topics will have to be directed to the author.

## UNIVERSITY OF MISSISSIPPI ATHLETIC ASSOCIATION

NAME
ADDRESS

25000

⊂A⊃⊂B⊃⊂C⊃⊂D⊃⊂E⊃⊂F⊃

NUMBER ▢

LIMIT OF FOUR SEASON TICKETS PER FAMILY UNTIL JUNE 1

SAME NUMBER OF TICKETS MUST BE ORDERED. FOR EACH GAME

| | SEASON TICKETS | DATE | PRICE | AMOUNT | SEC | ROW | SEATS | |
|---|---|---|---|---|---|---|---|---|
| SEASON TICKETS | | | | | | | | ⊂1⊃ |
| | | | | | | | | ⊂2⊃ |
| | | | | | | | | ⊂3⊃ |
| | | | | | | | | ⊂4⊃ |
| | | | | | | | | ⊂5⊃ |
| NUMBER ▢ ADDITIONAL GAME | | | | | | | | ⊂6⊃ |
| ▢ | | | | | | | | ⊂7⊃ |
| ▢ | | | | | | | | ⊂8⊃ |
| ▢ | | | | | | | | ⊂9⊃ |
| ▢ | | | | | | | | |

U M A A

REBEL GUIDE @ $ PER COPY

POSTAGE AND INSURANCE

TOTAL AMOUNT ENCLOSED $

✱ BOX SEATS @ $

✱✱ BOX SEATS @ $

⊂10⊃

IBM M43783

---

NAME
ADDRESS

25000

⊂A⊃⊂B⊃⊂C⊃⊂D⊃⊂E⊃⊂F⊃

| NUMBER | NON-SEASON TICKETS | DATE | PRICE | AMOUNT | SECTION | ROW | SEATS | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | ⊂1⊃ |
| | | | | | | | | ⊂2⊃ |
| ▢ | | | | | | | | ⊂3⊃ |
| ▢ | | | | | | | | ⊂4⊃ |
| ▢ | | | | | | | | ⊂5⊃ |
| ▢ | | | | | | | | ⊂6⊃ |
| ▢ | | | | | | | | ⊂7⊃ |
| ▢ | | | | | | | | ⊂8⊃ |
| ▢ | | | | | | | | ⊂9⊃ |
| ▢ | | | | | | | | |
| ▢ | | | | | | | | |
| ▢ | | | | | | | | |

U M A A

REBEL GUIDE @ $ PER COPY

POSTAGE AND INSURANCE

TOTAL AMOUNT ENCLOSED $

X - LIMIT OF FOUR UNTIL JULY 1

⊂10⊃

---

ALPHA NUMBER OR ID NUMBER | NAME | SOCIAL SECURITY NO. | SEX | INSTR. | SCHOOL | COUNTY | STATE

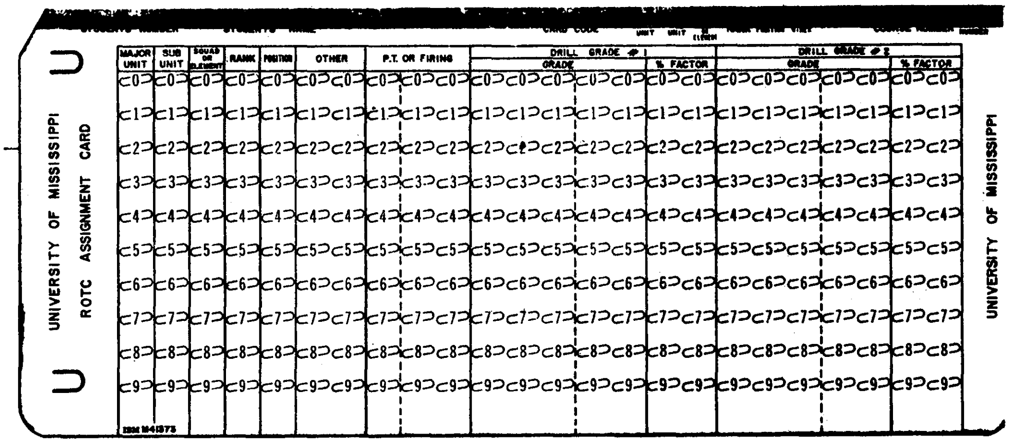
ATTENDANCE ACCOUNTING FOR 4-WEEK PERIOD

UNIVERSITY OF MISSISSIPPI

| 1st WEEK | | | | | 2nd WEEK | | | | | 3rd WEEK | | | | | 4th WEEK | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ | ⊂1⊃ |
| ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ | ⊂2⊃ |
| ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ | ⊂3⊃ |
| ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ | ⊂4⊃ |
| ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ | ⊂5⊃ |
| ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ | ⊂6⊃ |
| ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ | ⊂7⊃ |
| ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ | ⊂8⊃ |
| ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ | ⊂9⊃ |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

IBMM45761



STUDENTS NUMBER STUDENTS NAME CARD CODE UNIT UNIT COURSE NUMBER

U

UNIVERSITY OF MISSISSIPPI

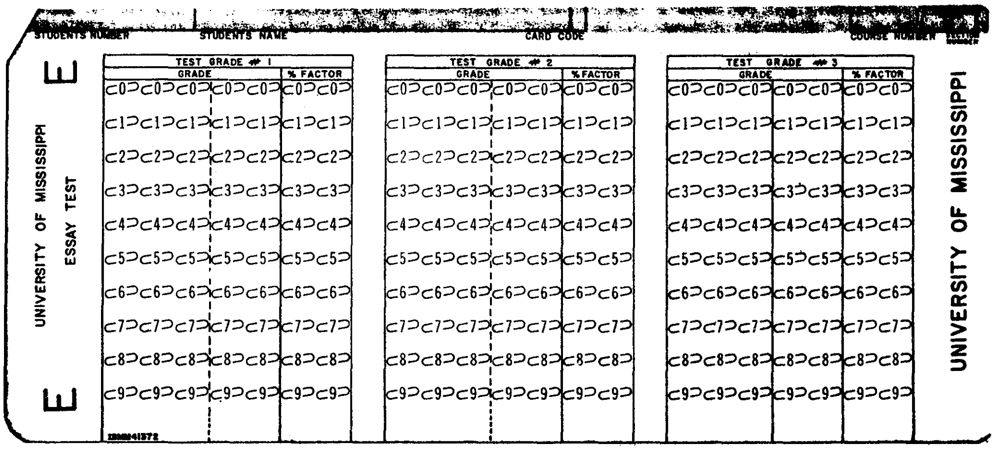
ROTC ASSIGNMENT CARD

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAJOR UNIT | SUB UNIT | SQUAD OR ELEMENT | RANK | POSITION | OTHER | | P.T. OR FIRING | | | DRILL GRADE # 1 | | | | | | | DRILL GRADE # 2 | | | | | | |
| | | | | | | | | | | GRADE | | | | | % FACTOR | | GRADE | | | | | % FACTOR | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

U

IBM M41373

UNIVERSITY OF MISSISSIPPI



STUDENTS NUMBER STUDENTS NAME CARD CODE COURSE NUMBER SECTION NUMBER

E

UNIVERSITY OF MISSISSIPPI

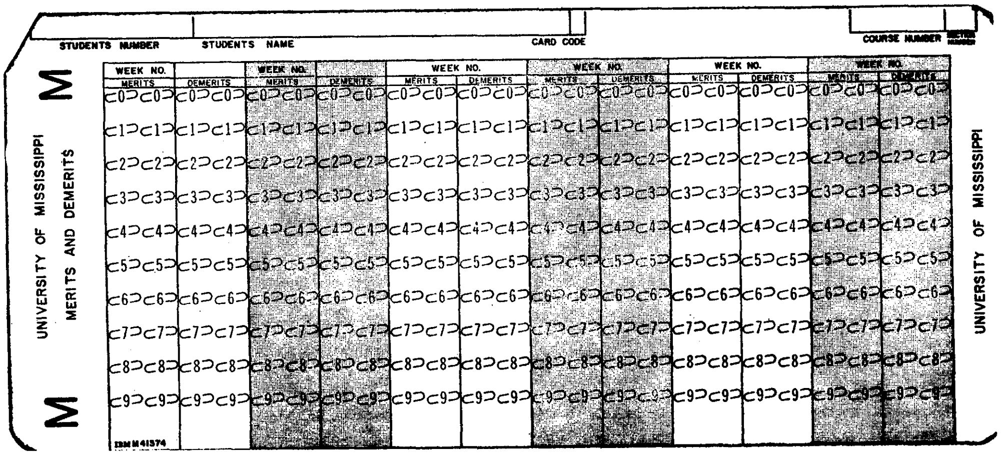
ESSAY TEST

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TEST GRADE # 1 | | | | | | | TEST GRADE # 2 | | | | | | | TEST GRADE # 3 | | | | | | |
| GRADE | | | | | % FACTOR | | GRADE | | | | | % FACTOR | | GRADE | | | | | % FACTOR | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

E

IBM M41372

UNIVERSITY OF MISSISSIPPI



STUDENTS NUMBER STUDENTS NAME CARD CODE COURSE NUMBER SECTION NUMBER

M

UNIVERSITY OF MISSISSIPPI

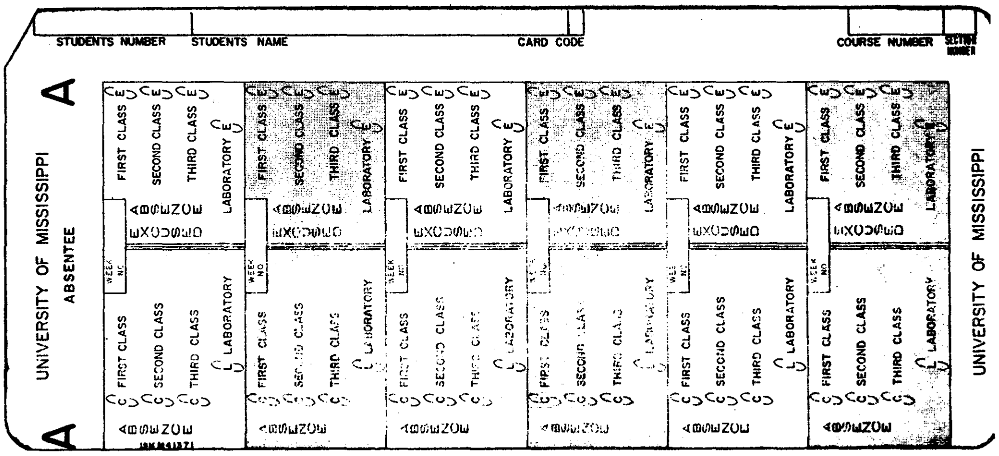
MERITS AND DEMERITS

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WEEK NO. | | | | WEEK NO. | | | | WEEK NO. | | | | WEEK NO. | | | | WEEK NO. | | | | WEEK NO. | | | |
| MERITS | | DEMERITS | | MERITS | | DEMERITS | | MERITS | | DEMERITS | | MERITS | | DEMERITS | | MERITS | | DEMERITS | | MERITS | | DEMERITS | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

M

IBM M41374

UNIVERSITY OF MISSISSIPPI



STUDENTS NUMBER STUDENTS NAME CARD CODE COURSE NUMBER SECTION NUMBER

A

UNIVERSITY OF MISSISSIPPI

ABSENTEE

| | | | | | |
|---|---|---|---|---|---|
| E E E | E E E | E E E | E E E | E E E | E E E |
| FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY E |
| ABSENCE<br>EXCUSED | ABSENCE<br>EXCUSED | ABSENCE<br>EXCUSED | ABSENCE<br>EXCUSED | ABSENCE<br>EXCUSED | ABSENCE<br>EXCUSED |
| WEEK NO | WEEK NO | WEEK NO | WEEK NO | WEEK NO | WEEK NO |
| FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U | FIRST CLASS<br>SECOND CLASS<br>THIRD CLASS<br>LABORATORY U |
| U U U | U U U | U U U | U U U | U U U | U U U |
| ABSENCE | ABSENCE | ABSENCE | ABSENCE | ABSENCE | ABSENCE |

A

IBM M41371

UNIVERSITY OF MISSISSIPPI

SAMPLE INPUT FOR

```
01   ACT TEST
02   DRT TEST  (FORM-F)
03   NELSON-DENNY READING TEST
04   ABSTRACT REASONING
05   EDWARDS TEST
06   MATH TEST
```

```
00001ROSS RICHARD     M                      UNIVERSITY  MISSISSIPPI  *
00001ROSS RICHARD    01M   117 27  27 85  17 28  17 27  20 46
00001ROSS RICHARD    F2M077196869897866789889677897793231424133241424211243314
00001ROSS RICHARD    F2M077266879879996897769879668794421331244113332441312144
00001ROSS RICHARD    03M   19668796568596775659877 984021434201133013422030423
00001ROSS RICHARD    03M   296758856787957799865566894124120344443022133300243
00001ROSS RICHARD    04M   157697876787959987968766852333001314411343011343234
00001ROSS RICHARD    05M   188989998889989899899998844433334344334434344443334
00001ROSS RICHARD    05M   289989998888998989998989894333434443343434443433343
00001ROSS RICHARD    05M   388989998899889988899999844443343434343434343433334
00001ROSS RICHARD    05M   489898989889899898899988883334344434333334434344433
00001ROSS RICHARD    05M   588998998889999889989998 98
00001ROSS RICHARD    06M   18867589966787675998599877323 22 34 110323443351 34
00001ROSS RICHARD    06M   29887657899

00002ROSS TONY        M                      UNIVERSITY  MISSISSIPPI  *
00002ROSS TONY       01M   114 13  19 46  16 22  16 22  16 19
00002ROSS TONY       F2M089198666977697876987896867693214123241214323342213341
00002ROSS TONY       F2M089299768968768796997768977861124341341242143423143421
00002ROSS TONY       03M   1777975688699598696877689500244202433223002334 1304
00002ROSS TONY       03M   2697568=975678865598977869421311232141220022423301 2
00002ROSS TONY       04M   166786599978856897 868957511442210244442231213100 44
00002ROSS TONY       05M   188899989899989989898998894344434433434443443333434
00002ROSS TONY       05M   289998889899998999899999893334333444344433333443 3334
00002ROSS TONY       05M   388998898989988989989898983344343434444433434333443
00002ROSS TONY       05M   488899999889989898898989894444443443444344343 33434
00002ROSS TONY       05M   589899888999898988899898 89
00002ROSS TONY       06M   18766788 65995775669989886312233444434322110112 3334
00002ROSS TONY       06M   29999786796
```

SAMPLE STRONG ENTRANCE TEST INPUT

```
00001ROSS RICHARD D      M    26 10                BATESVILLE    MISSISSIPPI   *
00001                    01870     78888889778799779998     11332232111323213113
00001                    01871     89779788879878987898     21112131311311333121
00001                    01872     88977998887877977789     33322323221133231133
00001                    01873     88779798978987799998     22133313111233322322
00001                    01874     89898877779799888889     23323333133211232321
00001                    01875     97899899879987899797     13131133123123231113
00001                    01876     79988887997977897987     21231111111122222213
00001                    01877     89989787877798889897     21331322121133123222
00001                    01878     77797977997977799799     33123311131111122321
00001                    01879     97999999999999997997     13313333212122223223
00002ROSS TONY A         M    20                   KAPPA ALPHA H MISSISSIPPI   *
00002                    01900     98797799978979878777     12112333133122112113
00002                    01901     97799899779977779777     13111111311322132221
00002                    01902     77787997877997778777     11121121331232333331
00002                    01903     77798999979778779998     23132313332122233333
00002                    01904     87998787799987898989     23232131213312332331
00002                    01905     98999999899799799777     33122333321133332133
00002                    01906     79989988997989788997     22222311213231132123
00002                    01907     78788798999897889877     11221333223221121233
00002                    01908     79988997777798787998     32311311132123132211
00002                    01909     87999798899778997977     21233333113332223223
```

# FORT 5

## A Monitor Control Record

In response to the need for a method to handle a large volume of FORTRAN programs from students with a limited number of key punches available, a system was developed whereby the students could use MARK SENSE cards to write their FORTRAN programs. This eliminated the need of the key punch for a FORTRAN program. The developed program was made available to 1620 USERS earlier this year through the library as the FORTRAN DECODER PROGRAM. This program, written by H. B. Kerr was designed for a 20K 1620, later being made adaptable to the 1311 MONITOR SYSTEM FORTRAN COMPILER, so the additional statements available in Monitor could be used with the MARK SENSE system.

The program, as designed, accepted as input the MARK SENSE PROGRAM and translated that coded program to regual FORTRAN SOURCE STATEMENTS and gave an output of each statement on a card, allowing, as I said earlier, The elimination of the key punch completely. These coded MARK SENSE programs could be batch-processed through the translator, thus saving a little more time in the processing of student programs.

Upon installation of the 1311 MONITOR SYSTEM at our computer center, it was felt that some means was needed to speed up this process and reduce the work load of the operator. I felt that if the TRANSLATOR program could be stored on disk and called when needed to translate a program, the process could be partially accelerated. Upon consideration of this idea, the thought that the incorporation of the translator with the FORTRAN COMPILER, so as to eliminate the need of forming a translated source deck, would accomplish our purpose even more.

Two basic methods of doing this were considered. The idea of translating the statements, and storing them on disk and then calling them back one by one to be compiled by the FORTRAN COMPILER was first in mind. This process however brought about complications in modifying the monitor system, such as: reserving disk storage area for the translated statements, not knowing what maxium size to save for programs. Then the compiler would have to accept its input statements from disk rather than card. And lastly, no means by which to bring in the FORTRAN COMPILER after the translation and storage had been completed--still allowing stacked input processing--was available.

After weighing these complications, the preceding idea was abandoned and the present method was developed. The Control Record Analyzer of the Monitor System was studied along with the Supervisor and the FORTRAN COMPILER? Another decision now had to be made: whether to expand the analyzer, or to eliminate something already present and use the area made available. Looking over available control records, it was found that the TYPE AND PAUS statements were not used enough to be considered necessary, so work was started with the area that checks for these control records.

By elimination of the TYPE and PAUS control records, an area about 100 digits in size was made available for any modification that was necessary. First, the PAUS statement was changed to FORT allowing the use of the ‡‡FORT CONTROL RECORD.

Considerable effort was made to call the translator from disk and have it, in turn, call the FORTRAN COMPILER, but that idea did not work originally because of lack of knowledge of the overlay routine of the MONITOR. other methods were considered by trial and error but none worked satisfactorily.

Then it was back to the idea of calling the translator first. To be able to do this and eliminate an overlay, it was necessary first to change the bottom limit of the symbol table area from 15999, as defined in the MONITOR SYSTEM, to 25999 allowing an area for the translator to reside in core at the same time the compiler was there. This was easily accomplished by simply changing the compare position of the symbol table overlay routine. For this reason alone, the program operates on a 40K or 60K system because of the symbol table.

Once this was done, work was begun on the instructions necessary to call both the translator and the compiler into memory at the same time. The only way to permanently place the translator on disk was to store it with a DIM Number and file-protect it. When this was done the 100 digits of the TYPE and PAUS area were used to compare on the FORT control record card and to seek and read the translator from disk into core at location 17000, which is the lowest area not used by FORTRAN IID. This worked and things were going fine, but now the linkage between the translator and the compiler had to be worked out.

The non-disk I/O section had to be modified to eliminate the compiler read statement. This could not be a permanent change because of the constant use of this statement. But since both the translator and the compiler were in memory, I felt that I could, by programing in the translator, change the instructions in the I/O causing a branch to the translator rather than the reading of a card, then use the input area of the translator to bring in the card. This created problems in the translator and a drastic modification of the translator was begun. The origional translator used the BRANCH AND TRANSMIT instruction frequently which could no longer be used because of the MONITOR SYSTEM'S use of it. Also the output of the translator had been on card and now the information in the output area of the translator had to be moved to the input area of the monitor system. The system's input area has flags in all even positions which were not affected by the normal Alpha reading of a card. When transmitting from the translator to the monitor input area I was destroying these flags which are necessary to the operation of the compiler. A routine for clearing and setting flags in the translator output area had to be added so it would be compatible with the monitor. As these changes were made new problems were created; These were worked out as they were encountered.

I then found it was necessary to bring in the translator and branch to it, doing the necessary modifications to the I/O in the SUPERVISOR then transferring control to the point in the Monitor where the FORTRAN COMPILER is normally called from disk and the proper indicators are set.

This gave me the necessary set of instructions to fill the 100 digit area reserved and I proceeded to give a trial run. The first statement was translated and compiled with no problem, but the system then check-stopped. Back to work, where more checking revealed that the card image area in the MONITOR SYSTEM changes to different locations when compiling a program; I had to work this change into my translator so that the output record would be placed at the correct location each time.

These corrections made, I once again made a run. It worked on the simple program that I had provided for a test. I then tried a more complicated program, with continuation cards in it, and developed additional problems.

These changes were continued until the program worked for all possible types of statements and then a sense switch setting was added to allow for the normal source statement typing, as called for by turning on switch 1 when compiling, or for punching of the source statement when switch 1 was off. The output could then be listed on the 407 for return to the programmer. The card-output option is normally used because of the speed increase realized.

After the completion of all programming, the system was loaded to disk and an extensive test began, using this method to process our daily work load of about 50 programs.

By using the control records for FORT, FOR, FORX, SPS, SPSX, we could stack input, mixing the MARK SENSE programs with others as desired. Our purpose was accomplished. We had cut drastically the time needed to process student programs, eliminating the heavy work load of the operator and by option, saving on the number of cards used in a normal program output.

The largest factor was the saving of time. When the programs were being translated, compiled, and executed as seperate steps using the standard FORTRAN COMPILER, we spent an average of 20 minutes on each program. After the disk method was perfected, the average program time was reduced to about 4 minutes - without the necessity of an operators presence constantly.

As I said, the program was given extensive tests by using daily for about 8 months. No additional bugs have developed in the overall system since it was placed in operation.

After I had submitted the abstract for this talk, I began work on my ideas and started reviewing my previous work. I was no longer satisfied with the system and the way the program worked and immediately started modification, only this time with more ease because the basic work had been completed.

The thought that when someone might wish to translate and compile a program without execution, it would be difficult under the system. I created another control record, this one being called FONT, for non-execution purposes. I was limited for space in which to make changes, so found in the SUPERVISOR in lower memory, an overlay read routine which I could use to call the translator from disk.

Using the additional space made available by this change, the instructions for both FORT and FONT were incorporated. Since I was now allowing for a non-execution run which could be used for program checking, I felt that a sense switch setting to allow for a non-output option would be convenient. Switch 4 was my choice since it is only used when typing in source statements for a FORTRAN program.

All these changes were made and the system again tested. It is now working, and up until now we have had no problems with it.

This complete program and procedure is being readied for the Library now and will be submitted soon. The system will be easy to incorporate by using the 5 change cards that will be provided and by placing the systems output deck of the translator that will be furnished at the back of the monitor system before initally loading.

# PREPARATION AND SCORING OF FORTRAN PROGRAMS BY COMPUTERS

By James H. Hayes, Jr. - Computer Center Director

Siena College - Loudonville, New York

## INTRODUCTION

As a College of Liberal Arts, Siena is primarily committed to the intellectual advancement of the student by the training of his mind through the arts of critical thought and correct expression. While it provides pre-professional training in many fields, its ideal is not to foster an extreme, premature specialization, but rather to provide that liberal education which is the comprehensive, cultural background necessary for the professions. In fulfillment of this aim, every student who enters the College is required to follow a prescribed program of core courses in the Liberal Arts. This requirement is composed of training in Languages and Literature, History and Social Science, Mathematics and Natural Science, and Philosophy and Theology, and includes an opportunity for concentrated study in special areas of interest in the Arts, Science, or Business.

## THE COMPUTER AND EDUCATION

The Computer Laboratory is used for teaching and faculty and student research, as well as an administrative arm and part of the Office of the Registrar. In line with the basic philosophy of computer operations, programs are intergrated into the curriculum of different courses. These range from course classes which come to the computer laboratory for class sessions in addition, introductory courses in programing and systems analysis are taught in conjunction with the laboratory. Started about two years ago, the laboratory contains an IBM 1620 computer with a 1622 card reader/punch, 026 card punch, 514 reproducer, 085 collator, 082 sorter, and a 407 accounting printer.

## OBJECTIVES

At Siena we visualize the student using the computer in the same manner as they would use the library. It should be a tool that is available for their use in solving assigned class problems, with no delay or waiting time. In this manner, several hundred students instead of only a selected few could learn to use the computer as a management tool.

## PROBLEM

The one unsolved problem we had in handling a large number of student programs in an efficient manner was the time that it took to keypunch the written instructions into cards for processing. This becomes more of a problem as you attempt to serve more students. We solved this problem of keypunching student programs by scoring of Fortran programs by the computer. This now allows us to process many student programs in a short period of time and eliminate this intermediate step.

## MARK-SENSE CARD DESIGN

Computer scoring of Fortran programs involves the use of "mark-sense" cards, specially designed punched cards on which students indicate their program logic, by writing in the provided spaces with special pencils (Exhibit 1).

## CARD PREPARATION

After the student has completed writing his program, the cards are returned to the computer center where they are fed through a 514 mark-sense reproducer and the marks that the student has made are read and put into the cards as punched holes.

## PROGRAM LANGUAGE

This program is written in the IBM Symbolic Programming System. It is designed to be compatible with the 1620-SP-020 version.

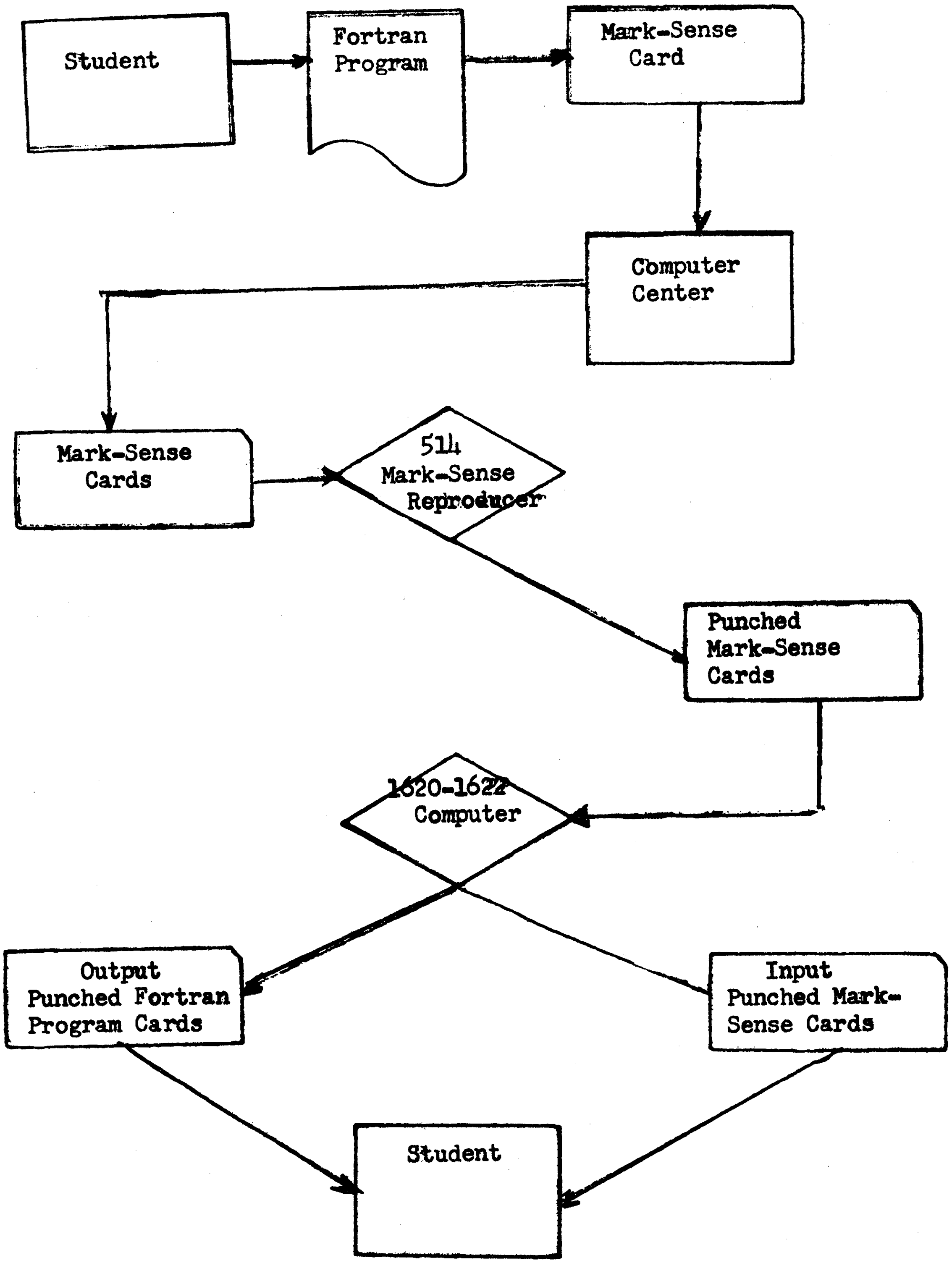
## MACHINE CONFIGURATION

The published version of the program is dimensioned to fit into a basic 1620 with 20K memory with card input and output.

## OPERATING PROCEDURES

1. Clear computer .... press RESET, RELEASE and INSERT. Type 16 00010 00000 and press RELEASE-START key. Next press INSTANT STOP and RESET.
2. Fill punch hopper of 1622 with blank Fortran cards.
3. Set object deck into read hopper of 1622 and press LOAD key.
4. After the object program has been loaded, set data cards in read hopper of the 1622. Press START on the console; program identification will now be typed out.
5. When READER NO FEED light on the console goes on, press READER START key on the 1622.
6. When PUNCH NO FEED light on the console goes on, press PUNCH START key on the 1622.
7. When the output is completely punched, press READER START on the 1622 to complete processing.

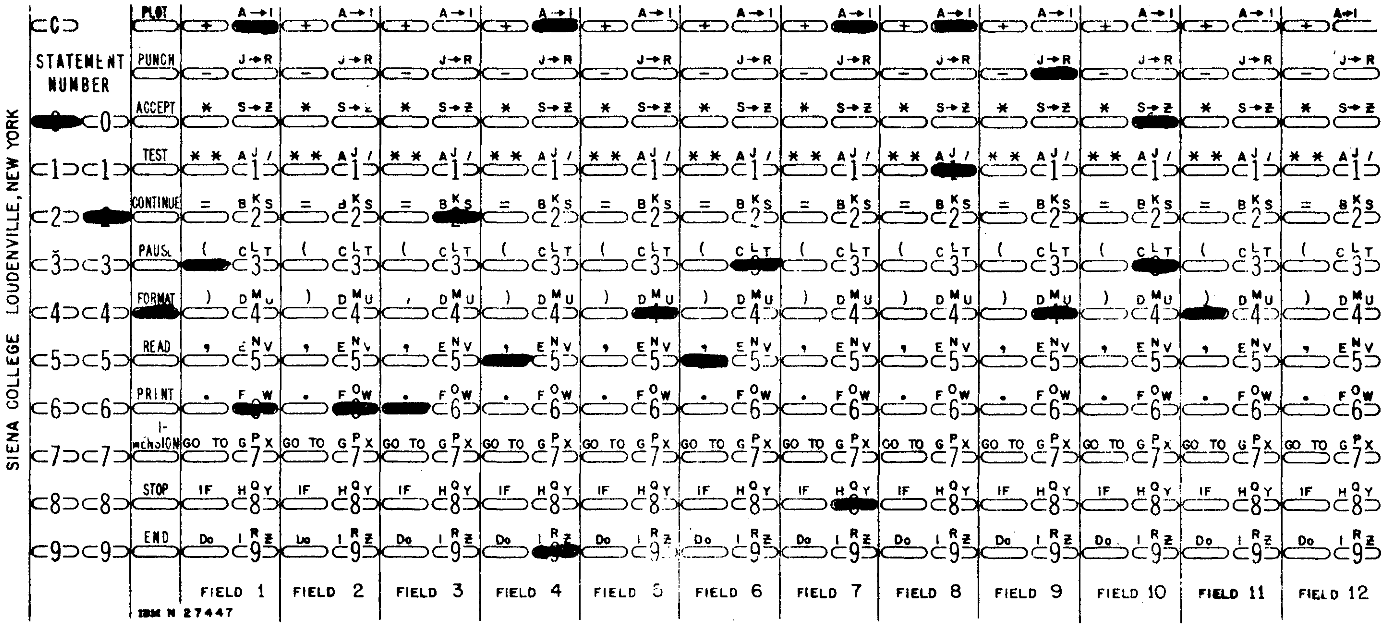
## FLOW CHART


Student
Fortran Program
Mark-Sense Card
Computer Center
Mark-Sense Cards
514 Mark-Sense Reproducer
Punched Mark-Sense Cards
1620-1622 Computer
Output Punched Fortran Program Cards
Input Punched Mark-Sense Cards
Student

(Exhbit 1)

## MARK-SENSE FORTRAN PROGRAM CARD

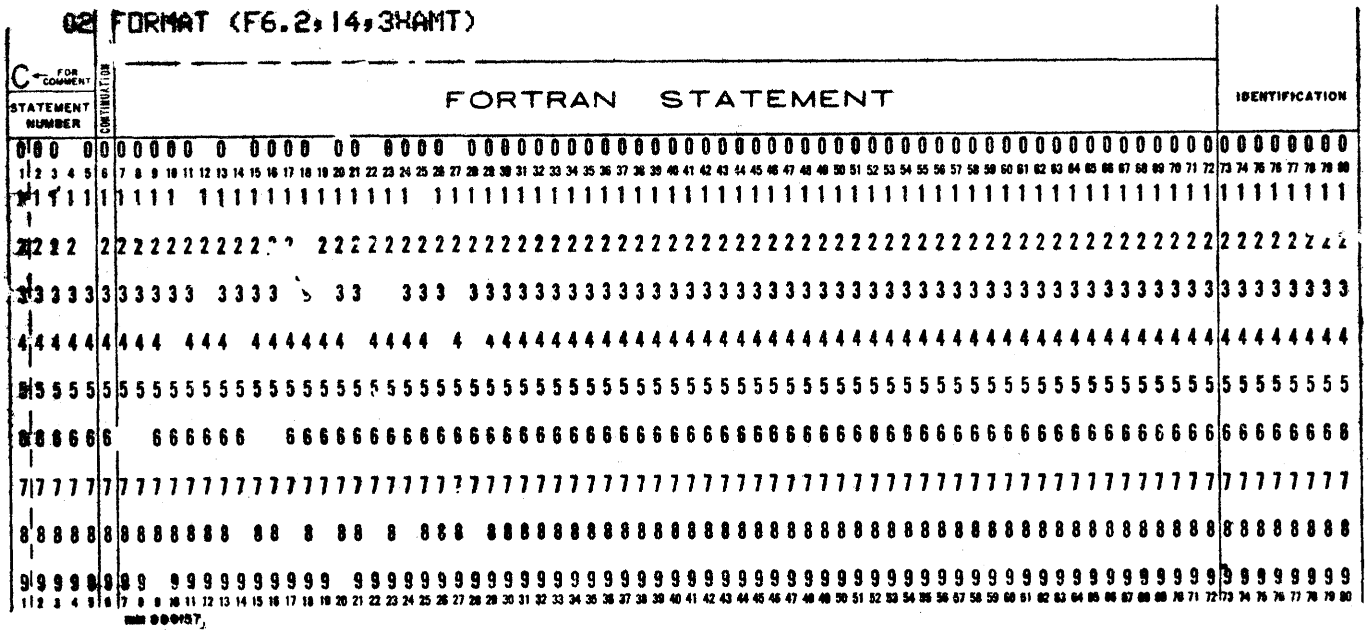
### INPUT


SIENA COLLEGE LOUDENVILLE, NEW YORK
C
STATEMENT NUMBER
PLOT
PUNCH
ACCEPT
TEST
CONTINUE
PAUSE
FORMAT
READ
PRINT
DIMENSION
STOP
END
GO TO
IF
DO
FIELD 1
FIELD 2
FIELD 3
FIELD 4
FIELD 5
FIELD 6
FIELD 7
FIELD 8
FIELD 9
FIELD 10
FIELD 11
FIELD 12
IBM N 27447

(Exhbit 2)

## FORTRAN PROGRAM CARD PUNCHED

### OUTPUT



```
02 FORMAT (F6.2,I4,3HAMT)
```

C FOR COMMENT
STATEMENT NUMBER
CONTINUATION
FORTRAN STATEMENT
IDENTIFICATION
IBM 888157

(Exhibit 3) MARK-SENSE S. P. S. SOURCE PROGRAM 1620

```
1050SETFLGSF   INPUT-1,,2
1060         AM   *-6,2
1070         CM   *-18,INPUT+157
1080         BNZ  *-36
1010START RCTY
1020         WATYMESS1
1030         H
1089BEGIN  RACDINPUT
1040         TFM KONT1,OUTPUT
1090         TF   OUTPUT+6,INPUT
1100         TF   OUTPUT+8,INPUT‸2
5001         TFM KONT2,INPUT,,TO USE INDIRECT ADDRESSING
5002GOB1     TFM KONT3,70,10
5102         CM   KONT2,0,610
5202         BZ   GO3
5003GO       C    KONT2,KONT3,6
5004         BZ   GO3
5016         C    KONT2,KONT3,6
5015         BN   GO2
5005GO1      AM   KONT3,1,10
5006         CM   KONT3,80,10
5008         BZ   GO2
5007         B    GO
5009GO2      RCTY
5010         WATYMES14
5011GO3      AM   KONT2,2,10
5012         CM   KONT2,INPUT+4
5013         BNZ  GOB1
1110         AM   KONT1,11,9,COUNTER STARTS AT 12
3001         CM   INPUT+4,0,10,NO PUNCH
3002         BZ   GOGO1
3003         CM   INPUT+4,10,10,PLOT
3004         BZ   MES02+9
3005         CM   INPUT+4,20,10,PUNCH
3006         BZ   MES03+11
3007         CM   INPUT+4,70,10,ACCEPT
3008         BZ   MES01+13
3009         CM   INPUT+4,71,10,TEST
3010         BZ   MES04+9
3011         CM   INPUT+4,72,10,CONTINUE
3012         BZ   MES05+17
3013         CM   INPUT+4,73,10,PAUSE
3014         BZ   MES06+11
3015         CM   INPUT+4,74,10,FORMAT
3016         BZ   MES07+13
3017         CM   INPUT+4,75,10,READ
3018         BZ   MES08+9
3019         CM   INPUT+4,76,10,PRINT
3020         BZ   MES09+11
3021         CM   INPUT+4,77,10,DIMENSION
```

```
3022         BZ   MES10+19
3023         CM   INPUT+4,78,10,STOP
3024         BZ   MES11+9
3025         CM   INPUT+4,79,10,END
3024         BZ   MES12+7
3026         CM   INPUT+4,0,10,ERR 1
3027         BNZ  MES13+11
4031MES01    DAC  7,ACCEPT'
4032         TR   KONT1,MES01-1,6
4033         AM   KONT1,14,9
4034         B    GOLOOP
4036MES02    DAC  5,PLOT'
4037         TR   KONT1,MES02-1,6
4038         AM   KONT1,10,9
4039         B    GOLOOP
4041MES03    DAC  6,PUNCH'
4042         TR   KONT1,MES03-1,6
4043         AM   KONT1,12,9
4044         B    GOLOOP
4046MES04    DAC  5,TEST'
4047         TR   KONT1,MES04-1,6
4048         AM   KONT1,10,9
4049         B    GOLOOP
4051MES05    DAC  9,CONTINUE'
4052         TR   KONT1,MES05-1,6
4053         AM   KONT1,18,9
4054         B    GOLOOP
4065MES06    DAC  6,PAUSE'
4057         TR   KONT1,MES06-1,6
4058         AM   KONT1,12,9
4059         B    GOLOOP
4001MES07    DAC  7,FORMAT'
4002         TR   KONT1,MES07-1,6
4003         AM   KONT1,14,9
4004         B    GOLOOP
4006MES08    DAC  5,READ'
4007         TR   KONT1,MES08-1,6
4008         AM   KONT1,10,9
4009         B    GOLOOP
4011MES09    DAC  6,PRINT'
4012         TR   KONT1,MES09-1,6
4013         AM   KONT1,12,9
4014         B    GOLOOP
4016MES10    DAC  10,DIMENSION'
4017         TR   KONT1,MES10-1,6
4018         AM   KONT1,20,9
4019         B    GOLOOP
4021MES11    DAC  5,STOP'
4022         TR   KONT1,MES11-1,6
4023         AM   KONT1,10,9
4024         B    GOLOOP
4026MES12    DAC  4,END'
```

```
 4027      TR  KONT1,MES12-1,6
 4028      AM  KONT1,8,9
 4029      TFM KONT1,0,69
 4030      B   END
 4060MES13 DAC 6,ERR 1'
 4061      TR  KONT1,MES13-1,6
 4062      AM  KONT1,12,9
 4065      RCTY
 4063      WATYMES13
 4064      B   GOLOOP
 2120MESS1 DAC 37,ENTER MARK SENSE PROGRAM, PUSH START'
 2130MESS2 DAC 20,PROCESSING COMPLETE'
 2140INPUT DAS 80
 2150OUTPUTDAS 82
 2170KONT1 DC  5,0
 2171CLEAR DAS 81
 2172RCMK  DAC 1,'
 2173KONT2 DC  5,0
 2173KONT3 DC  2,0
 2180KONT4 DC  2,0
 2174MES14 DAC 6,ERR 2'
 2177MES15 DAC 6,GO TO'
 2178MES16 DAC 3,IF'
 2179MES17 DAC 3,DO'
55555GOLOOPTFM KONT1,0,69
 5444GOGO1 AM  KONT1,1,10
 5445      TFM KONT4,0,10
 5446GOGO2 AM  KONT2,2,10
 6999GOGO3 CM  KONT2,0,610
 7000      BZ  GOGO4
 7001      CM  KONT2,10,610,PLUS SIGN
 7002      BNZ *+48
           AM  KONT1,2,10
 7003      TFM KONT1,10,610
           AM  KONT1,2,10
 7004      CM  KONT2,20,610,MINUS SIGN
 7005      BNZ *+48
           AM  KONT1,2,10
 7006      TFM KONT1,20,610
           AM  KONT1,2,10
 7007      CM  KONT2,70,610,MULTIPLY
 7008      BNZ *+24
 7008      TFM KONT1,14,610
 7010      CM  KONT2,71,610,SQUARE
 7011      BNZ *+48
 7012      TFM KONT1,14,610
 7013      AM  KONT1,2,10
 7014      TFM KONT1,14,610
 7015      CM  KONT2,72,610,EQUAL SIGN
 7016      BNZ *+48
           AM  KONT1,2,10
```

```
7017        TFM KONT1,33,610
            AM  KONT1,2,10
7018        CM  KONT2,73,610,OPEN PARENTHESIS
7019        BNZ *+24
7020        TFM KONT1,24,610
7021        CM  KONT2,74,610,CLOSE PARENTHESIS
7022        BNZ *+24
7023        TFM KONT1,4,610
7024        CM  KONT2,75,610,COMMA
7025        BNZ *+24
7026        TFM KONT1,23,610
7027        CM  KONT2,76,610,PERIOD
7028        BNZ *+24
7029        TFM KONT1,3,610
7030        CM  KONT2,77,610,GO TO
7031        BNZ *+72
            SM  KONT1,1,10
7032        TR  KONT1,MFS15-1,6,GO TO
7033        AM  KONT1,11,10
7034        TFM KONT1,0,610,CLFAR RFCORD MARK
            SM  KONT1,2,10
7035        CM  KONT2,78,610,IF
7036        BNZ *+72
            SM  KONT1,1,10
7037        TR  KONT1,MES16-1,6,IF
7038        AM  KONT1,5,10
7039        TFM KONT1,0,610,CLEAR RFCORD MARK
            SM  KONT1,2,10
7040        CM  KONT2,79,610,DO
7041        BNZ *+72
            SM  KONT1,1,10
7042        TR  KONT1,MFS17-1,6,DO
7043        AM  KONT1,5,10
7044        TFM KONT1,0,610,CLEAR RECORD MARK
            SM  KONT1,2,10
7045        AM  KONT1,2,10
7050GOGO4   AM  KONT2,2,10
            CM  KONT1,OUTPUT+143
            BZ  WACD
            CM  KONT1,OUTPUT+143
            BP  WACD
7051        CM  KONT2,0,610
7052        BZ  GOGO5
7053        TF  KONT1,KONT2,611
7053        AM  KONT1,2,10
7054GOGO5   AM  KONT4,1,10
            CM  KONT1,OUTPUT+143
            BZ  WACD
            CM  KONT1,OUTPUT+143
            BP  WACD
7055        CM  KONT4,12,10
7056        BNZ GOGO2
```

```
1040WACD   WACDOUTPUT
           TR  OUTPUT,CLEAR
1041       B   BEGIN
1042END    WACDOUTPUT
1043       RCTY
1044       WATYMESS2
1045       TR  OUTPUT,CLEAR
1046       BNLCBEGIN
1047       H
1048       B   START
1000       DENDSETFLG
```

# THE STATISTICAL VALIDITY OF APPLYING NUMERICAL SURFACE TECHNIQUES AND CONTOUR MAP PLOTTING TO CORRELATION PROBLEMS

L. D. Y. Ong

Health and Safety Laboratory
U. S. Atomic Energy Commission
New York, New York

Presentation at 1620 Users Group Meeting
in New York, New York, October 7, 1965

5/

## INTRODUCTION

Correlation and regression analysis is concerned with the study of relationships between variables. When three or more variables are involved, it is standard practice to initially try to apply the linear relationship $y = a + b_1 x_1 + b_2 x_2$ to describe the data, usually using logarithmic or other transformations to deal with nonlinear cases. If the description of a multidimensional curvilinear regression surface cannot be reduced to this multiple linear regression form, the applied statistician is faced with the formidable task of determining joint causation quantitatively with the model $y = f(x_1, x_2)$. Those who have had this experience can fully appreciate the difficulty that arises in selecting not only a regression equation that fits the data within tolerance but also one whose terms may be deduced logically.

This talk attempts to make the audience aware of a simple alternate method of analysis: plot the variables as three-dimensional coordinate values and infer their regression surface graphically in the form of a contour map. This visual inference of the surface can be used as a model of the process, and can also greatly simplify choosing an appropriate equation for approximating the expected relation algebraically. If very high correlation exists between the surface variables, the analyst, given a sufficient number of points distributed rather uniformly over the ranges of interest and using his fundamental knowledge of the subject, could probably freehand-draw contour lines approximating the "true" surface reasonably well. However, besides the fact that these optimum conditions are seldom met, so-called "eyeball" curve fits introduce an individual bias that, of course, will vary from analyst to analyst.

Given data with not-so-high correlation, a marginally sufficient sample size, and a far-from-uniform distribution of points with large gaps representing unobserved areas, the uniqueness of the contours and consequently the validity of their statistical inferences might be significantly affected and therefore, the true pattern obscured.

In order to minimize this individual bias, a more exact and standardized calculation procedure, perhaps necessitating the use of a computer, for providing the first approximations of the contour lines is desirable. This more formidable statistical approach would provide the best detailed picture of the sample distribution, including values interpolated between the collected observations. It is the speaker's opinion that the best estimate of the "true" or rather population distribution would be a <u>final</u> smoothing of the calculated sample surface. These "ultimate" contour lines, based on the calculated sample surface, should be freehand-drawn by the analyst utilizing his important logical intuition acquired from extensive experience with the subject field. His contribution would be especially valuable in areas sparsely sampled, such as the perimeters of the sampling areas.

## COMPUTER CONTOURING METHODS

A convenient calculation scheme would be the application of IBM 1620/1311 Numerical Surface Techniques and Contour Map Plotting (1620-CS-05X), which is a set of programs that primarily processes three-dimensional coordinate values into a surface which may be expressed graphically in the form of a contour map. This program package recently made available and now coming into general use for providing a visual description of two-parameter distributions, was written by the IBM Corporation in 1620 SPS II-D, is stored on disk and is executed under the control of IBM 1620 Monitor I or II with maps drawn by the Calcomp 560 series (or IBM 1627) plotter. Its 119-page application program reference manual[1] competently describes the two general surface fitting techniques employed by the package:

1. Numerical approximation over a uniform grid with or without smoothing.
2. Orthogonal polynomial curve-fitting.

Besides providing a standard, general computational method, this computer approach eliminates the tedium of calculation and facilitates the handling of unwieldy amounts of data.

## COMPARISON OF COMPUTER METHODS

From our experience, the orthogonal polynomial approach is recommended over the uniform grid technique for solving multiple correlation problems where we are trying to predict the value of a dependent variable y for any given values of two or more independent variables $x_j$. This is because the polynomial method for fitting a surface emphasizes the determination of general trends without possibly misleading localized patterns. In addition, the resulting contours are relatively unique since their computation is sensitive to only two easy-to-converge-on assumptions: the order of the polynomial and the number of significant terms in the equation. The fit is always smoothed with the automatic elimination of insignificant peaks and valleys. Results appear to be fairly insenstive to a third required assumption - the grid interval setting.

An example of such a fit is the isopleth diagram shown in Figure 1[2] which relates strontium-90/calcium ratios (measured from samples of human vertebrae collected in New York City) with age and time of death. Here the dependent variable in this three-dimensional time series is highly correlated to two independent variables and general trends are made obvious. Very high correlation between the dependent variable and age was evident with the contours forming a definite family of curves.

In contrast, the uniform grid approach is recommended primarily for engineering-type contouring where it is reasonable to assume that the local peaks and valleys have significant meaning, e.g. in the construction of elevation contour maps. Here the analyst is trying to create a more detailed picture of his sample and is essentially concerned with interpolating values between the observed points rather than trend determination.

However, the technique can be useful to the correlation analyst who is trying to gain a better insight as to how the true surface is influenced by variables besides those represented by the graph's ordinate and abscissa scales. An example of this is the isopleth diagram shown in Figure 2[3] where strontium-90 fallout deposition is related to latitude and time. It is evident that other variables in addition to latitude and time are affecting the dependent variable. The contour lines were actually sketched freehand with uniform gri d computer results employed as a guiding first approximation.

## MANUAL CONTOURING METHODS

The disadvantage of the uniform grid method is that the calculated surface contours are prone to variation, being highly sensitive to three assumptions: grid interval setting, number of points for smoothing and smoothing technique used.

The effect of the sensitive settings is underlined by observing the results from processing data, with both the machine techniques discussed and the manual contouring methods developed by Dr. Mordecai Ezekiel[4] in 1926, long before the advent of the high-speed digital computer. These now-classical correlation techniques were further developed in his book on methods of correlation analysis first published in 1930[5], including an extension by Dr. Frederick V. Waugh[6]. By first subgrouping his data, manually averaging the observations in each subclassification, and then two-way smoothing the averages by employing four successive sets of freehand-fitted approximation curves, Dr. Ezekiel was able to determine the contoured regression surface representing the joint functional relation between three variables. The specific example in his book related expected individual haystack volume with basal diameter and height. Application of the uniform grid method to the haystack data with a grid interval setting of 8 and without smoothing resulted in Figure 3. Even with this grid interval optimization, a great deal of statistical "noise" was evident in the form of jagged curves, and scattered peaks and valleys -- results far different from those inferred by Drs. Ezekiel and Waugh. Only after improvising with the smoothing routine were we able to converge to agreement with the freehand-drawn contours.[7] The resulting family of curves, shown in Figure 4, are practically the same as Ezekiel's, where the standard deviation of the surface residuals is 0.03, significantly lower than its one-dimensional standard deviation of 0.13.

Application of the orthogonal polynomial computational method using a grid interval of 1, a 4th order equation and 12 coefficient terms resulted in Figure 5, practically identical to Ezekiel's inferences and those of Figure 4.

## SUMMARY

Experience has thus indicated that the basic approaches employed by the IBM contouring package provide the analyst with an automatically calculated, more complete picture of his sample data. All machine methods are based on the least squares criterion and consider the data point by point, instead of first averaging the observations as manual methods do. Especially in regard to the uniform grid technique, these "statistically sophisticated" approaches are probably most useful in interpolating between observed points, e.g. a civil engineering surveyor preparing a topographic map with particular interest in local features.

However, in general correlation problems, where we are more concerned with predicting the value of one variable from specified values of two others, the orthogonal polynomial approach is more valid because of its emphasis on general rather than possibly misleading localized trends and because its resulting contour lines are more unique, i.e. less prone to variation caused by required model assumptions. Besides exercising caution in the use of computer generated contour surfaces because of their sensitivity to required assumptions, the correlation analyst, unlike the topographer drawing a detailed elevation map, must be even more cautious that he is not "overcalculating" the solution to his problem. He should regard his machine fits primarily as conveniently calculated. standardized, interpolated sample data tables from which to infer joint functional relations, whose final form may be improved with his logical ingenuity gained from past experience with the process.

## REFERENCES

(1) IBM 1620 Numerical Surface Techniques and Contour Map Plotting (1620-CX-05X) User's Manual, H20-0192-0.

(2) Rivera, J. and Harley, J., "The HASL Bone Program 1961-1964" USAEC Report HASL-163, August 1965.

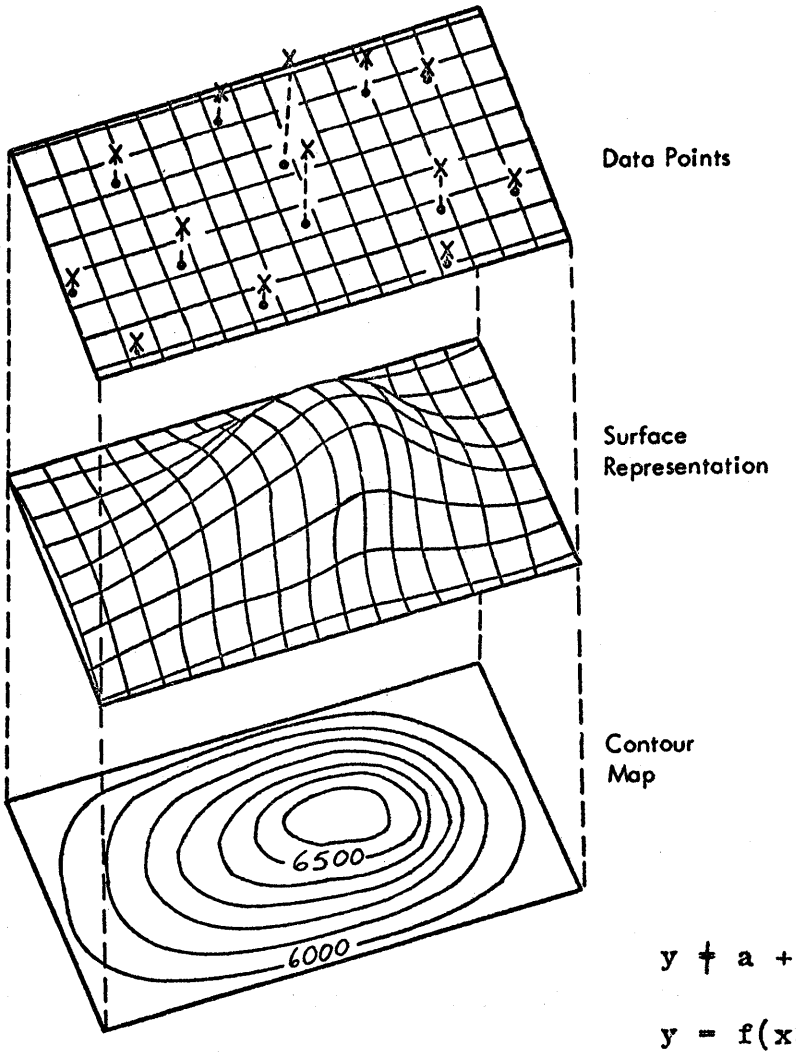
(3) Volchok, H. L., "World Wide Deposition of SR-90 through 1964" USAEC Report HASL-161, July 1965.

(4) Ezekiel, Mordecai, "The Determination of Curvilinear Regression Surfaces in the Presence of Other Variables", *Jour. Amer. Stat. Assoc.,* Vol. XXI, pp. 310-320, Sept. 1926.

(5) Ezekiel, M., *Methods of Correlation Analysis,* John Wiley and Sons, 1930.

(6) Waugh, F. V., "The Use of Isotropic Lines in Determining Regression Surfaces, *Jour. Amer. Stat. Assoc.,* p. 144, June, 1929.

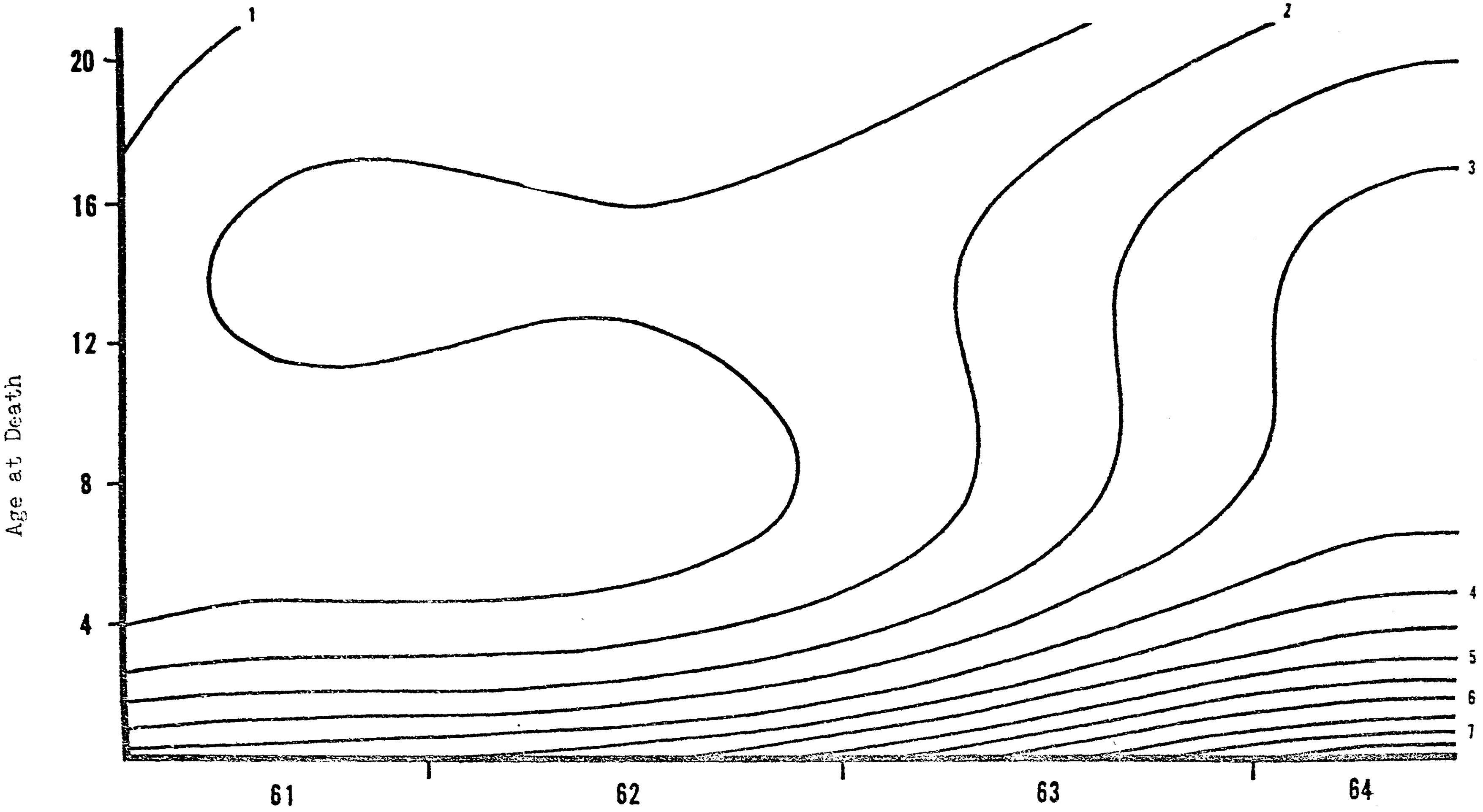
(7) David M. Schalk's assistance in machine-plotting the contour maps is gratefully acknowledged. Whereas the IBM application program's method of single smoothing, based on a least squares-fitted surface, proved unsuitable in our examples, a continuous smoothing procedure based on simple weighted arithemtic mean was developed by Stephen L. Samson and was utilized. Particularly appreciated were Dr. Ezekiel's personal interest and advice concerning the general applicability of computer-generated surface contours.



$$y \neq a + b_1x_1 + b_2x_2$$

$$y = f(x_1,x_2)$$

Numerical Surface Techniques and Contour Map Plotting



Strontium-90 Content of Vertebrae (pc/g Ca) as a
Fraction of Age and Time of Death

FIGURE 1

Monthly Sr-90 Deposition as a Function of Time and Latitude
(Isopleths are mCi/$mi^2$ of Sr-90 per month)

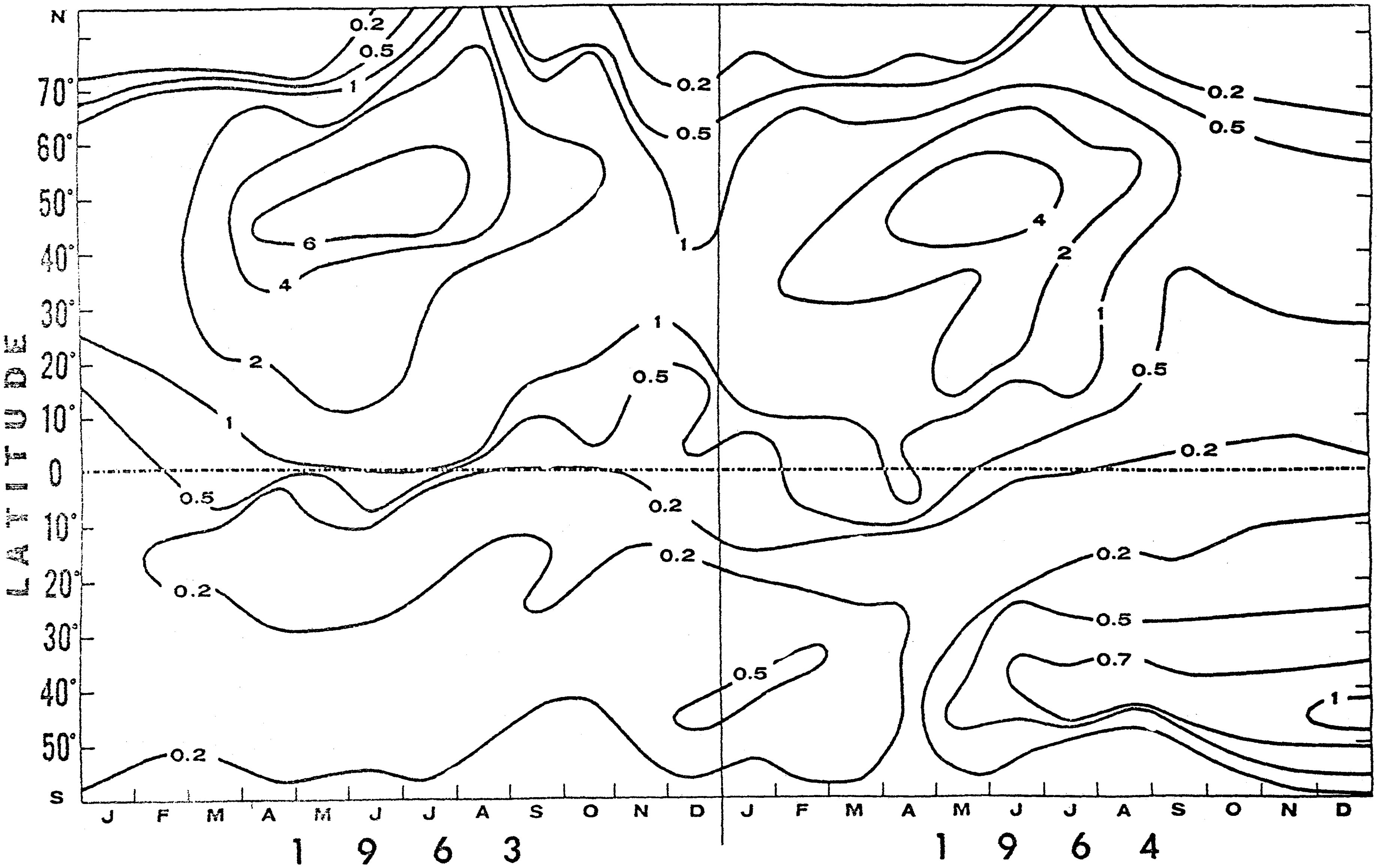


FIGURE 2

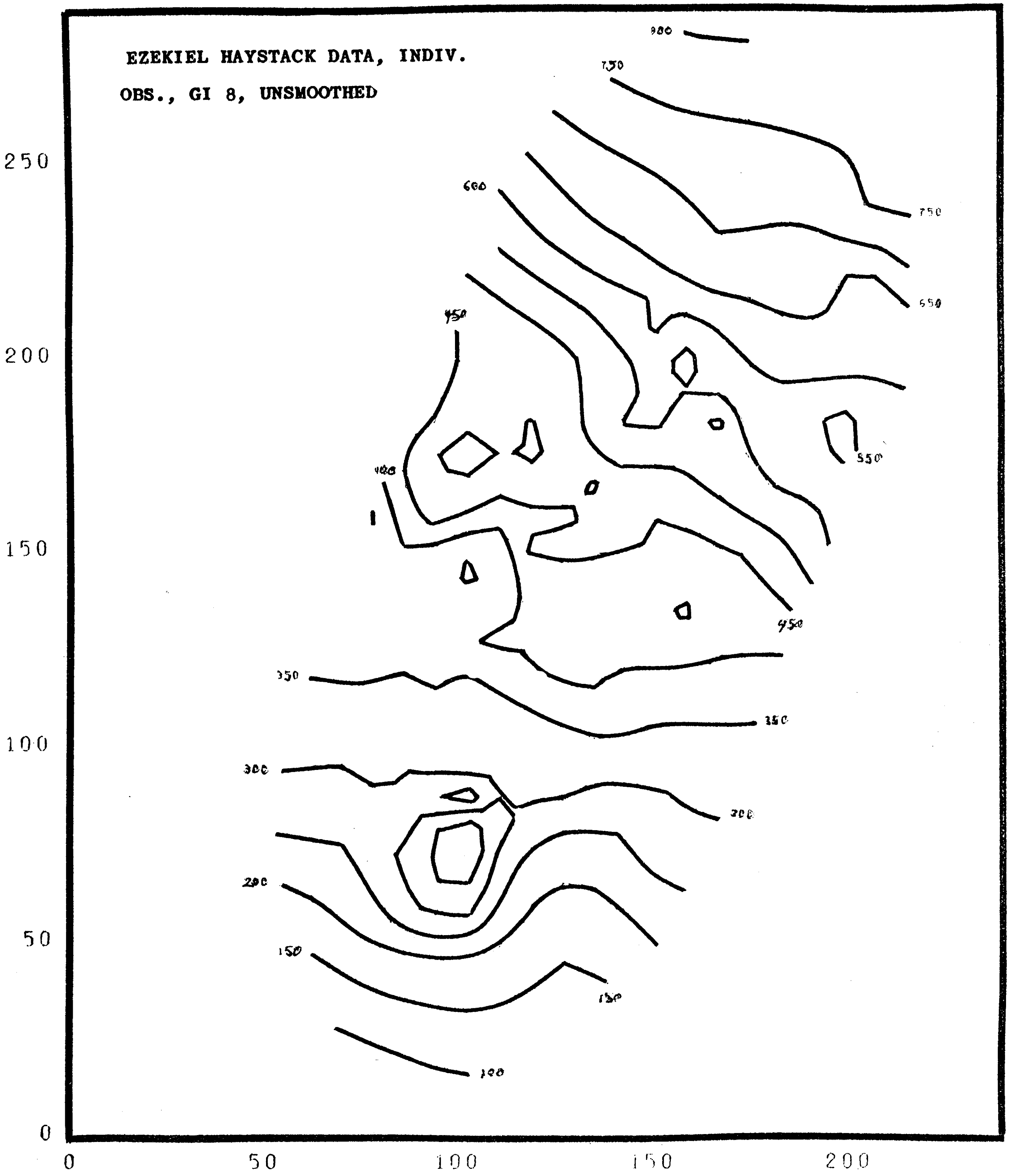

EZEKIEL HAYSTACK DATA, INDIV.
OBS., GI 8, UNSMOOTHED
900
750
750
600
650
450
400
550
450
350
350
300
300
200
150
150
100
250
200
150
100
50
0
0
50
100
150
200

FIGURE 3

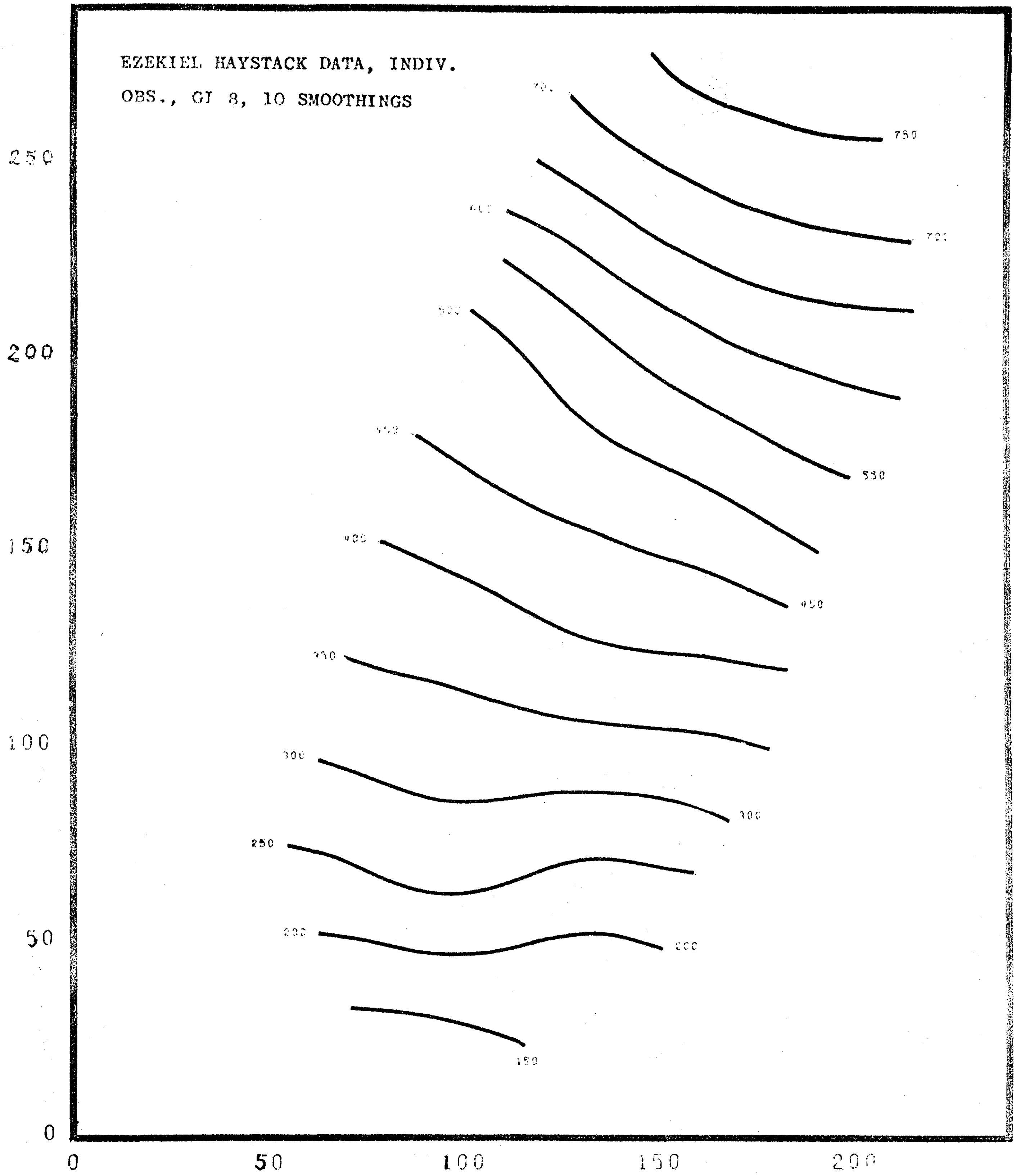

EZEKIEL HAYSTACK DATA, INDIV.
OBS., GI 8, 10 SMOOTHINGS
700
750
650
700
600
550
500
450
450
400
350
300
300
250
200
200
150
0
50
100
150
200
250
0
50
100
150
200

FIGURE 4

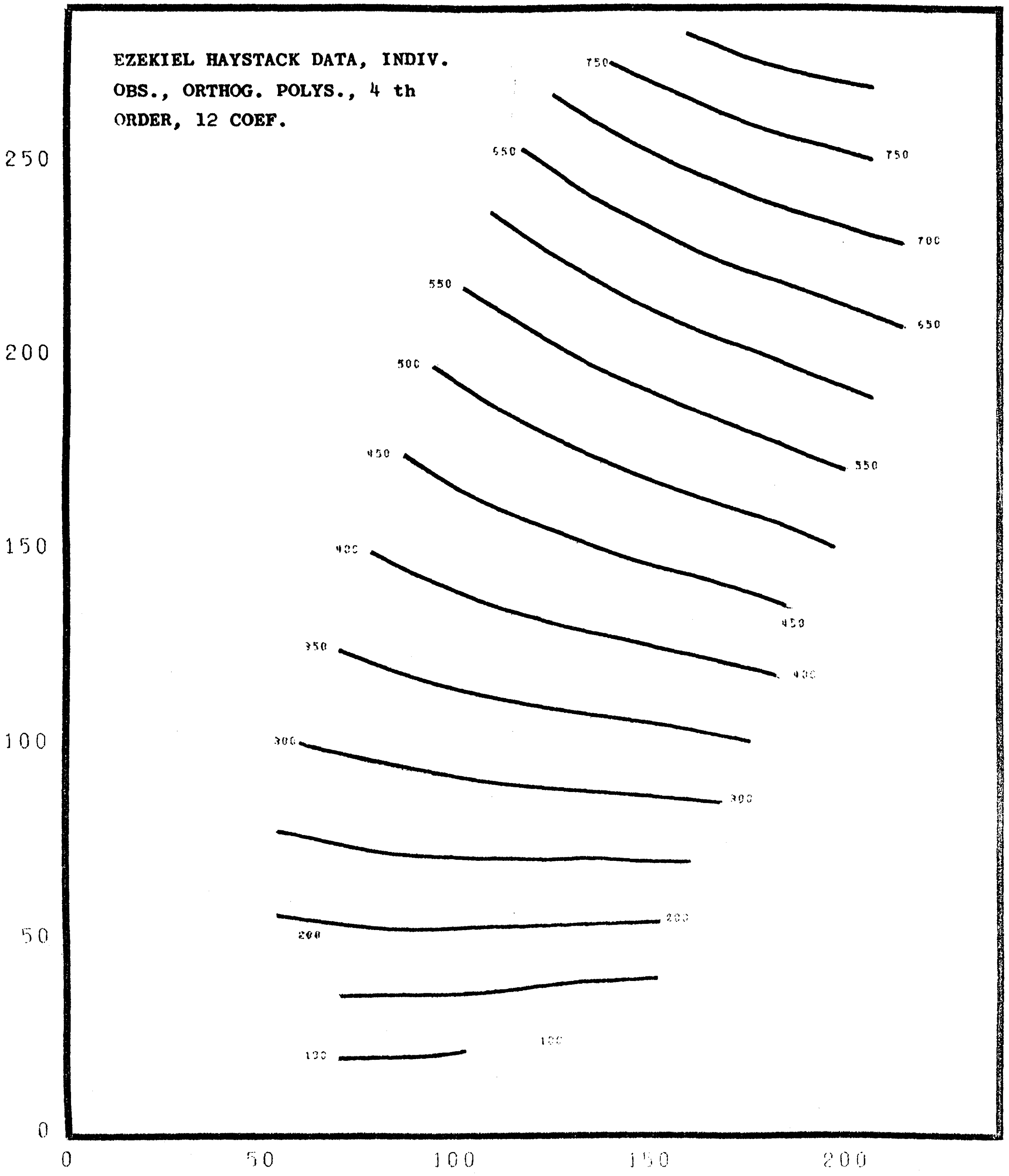

EZEKIEL HAYSTACK DATA, INDIV.
OBS., ORTHOG. POLYS., 4 th
ORDER, 12 COEF.
750
750
650
700
650
550
550
500
450
450
400
400
350
300
300
200
200
100
100
250
200
150
100
50
0
0
50
100
150
200

FIGURE 5

# 1620 Worst-Case Circuit Design Problem

by

S. S. Husson and H. C. Yang
International Business Machines Corporation
Systems Development Division, Poughkeepsie, New York

## INTRODUCTION

In this paper, we present a computer method for solving the d-c circuit design problem. All circuits are designed by considering Kirchhoff's circuit law together with the design constraints. Kirchhoff's circuit equations are usually an underdetermined system; there are more variables than equations.

We define a solution region as indicated in Fig. 1. The additional considerations of the design constraints will form a feasible solution region. Any solution in this region is a successful circuit. In eingineering practice, no attempt is being made to solve this general design problem; instead, many assumptions are made to obtain a design. The intention of these assumptions is to reduce the computational difficulty, and the significance to the design problem is to consider more constraints than are necessary. In other words, the feasible solution region will be narrowed down further such that no design may be achieved at all. The method described in this paper is originated from a realization of the basic design problem, and the method is formulated in such a way that the circuit design can be obtained through the computer without the unnecessary assumptions.

## FORMULATION OF THE D-C DESIGN PROBLEM

Using Kirchhoff's circu -law and considering the design requirements, we formulate the d-c circuit design problem as follows:

$N_e$ = number of elements in the circuit

$N_N$ = number of nodes in the circuit

$N_{VI}$ = number of voltage and current sources in the circuit

There are precisely $N_e - N_N + 1$ independent linear loop voltage equations, consisting of $N_e$ voltage variables, in the circuit.

$$X_{MIN} \le X \le X_{MAX}$$

$$Y_{MIN} \le Y \le Y_{MAX}$$

$$X = Y^2$$

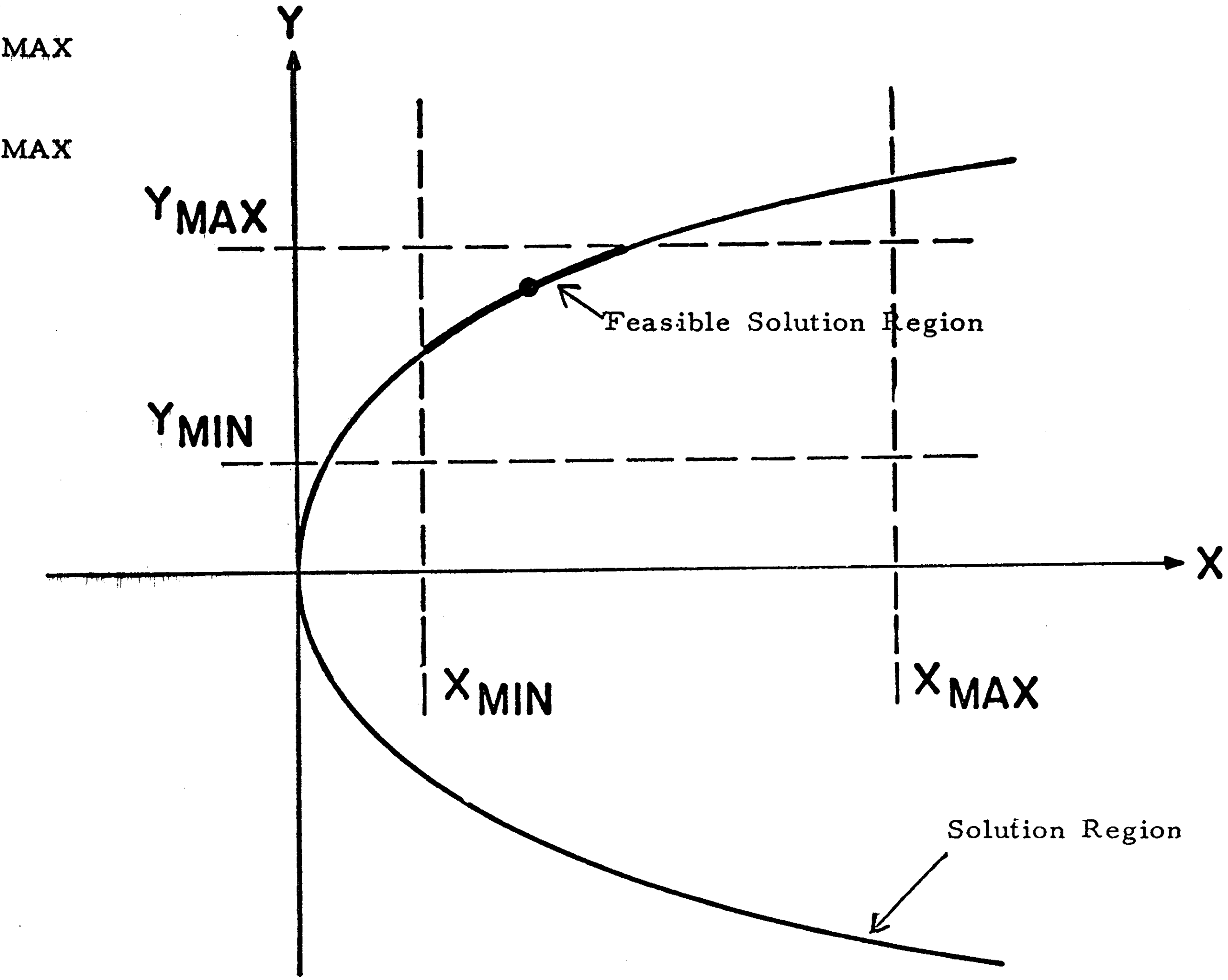


**Figure 1.** Graphical Interpretation of the Design Problem.

$$f_i\,(\bar{V}) = 0 \qquad , \; i = 1, \ldots, N_e - N_N + 1$$

$$\bar{V} = (V_1, \ldots, V_{Ne})$$

There are $N_N - 1$ independent linear node current equations, consisting of $N_e$ current variables, in the circuit.

$$f_i\,(\vec{I}) = 0$$

$$\vec{I} = (I_1, \ldots, I_{Ne}) \qquad , \; j = 1, \ldots, N_N - 1$$

There are $N_e - N_{VI}$ independent terminal equations, intcoducing $N_e - N_{VI}$ resistant variables, in the circuit.

$$V_k = I_k\,R_k \qquad k = 1, \ldots, N_e - N_{VI}$$

The terminal equations for the source variables are simply V = V for voltage source and I = I for current source; therefore, it is not necessary that they appear in the terminal equations.

The general mathematical model for d-c circuit is as follows:

$$f_i\,(\bar{V}) = 0 \qquad i = 1, \ldots, N_e - N_N + 1$$

$$f_j\,(\bar{I}) = 0 \qquad j = 1, \ldots, N_N - 1$$

$$V_k = I_k\,R_k \qquad k = 1, \ldots, N_e - N_{VI}$$

$$\vec{V} = (V_1, \ldots, V_{Ne})$$

$$\vec{I} = (I_1, \ldots, I_{Ne})$$

Total number of variables = $N_e + N_e + N_e - N_{VI} = 3\,N_e - N_{VI}$

Total number of equations = $N_e - N_N + 1 + N_N - 1 + N_e - N_{VI} = 2\,N_e - N_{VI}$

Consequently, this model is an underdetermined system of equations.

Any circuit is designed to meet certain requirements, hence a set of inequalities can be obtained as follows:

$$V_{min} \le V \le V_{max}$$

$$I_{min} \le I \le I_{max}$$

$$R_{min} \le R \le R_{max}$$

Now the d-c design problem can be defined as finding the values of V, I, and R such that the required inequalities and Kirchhoff's equations are all satisfied.

## METHODS USED TO SOLVE THE DESIGN PROBLEM

### The Limited Case

By applying engineering judgment, an undertermined system can be transformed to a determined system by assigning values to certain variables. Hence, this circuit design problem is reduced to a problem of solving a nonlinear system of equations. One method of solving this problem is the successive-approximation procedure of Newton-Raphson. Essentially, this procedure considers the first two terms of the Taylor's series expansion of each equation at an assigned starting point and increments the values of each variable accordingly. The convergence of this procedure depends upon the behavior of the function in the neighborhood of the solution and the nearness of the starting point to the solution.

The Newton-Raphson method is essentially a successive approximation procedure. Mathematically, the system of nonlinear equations

$$f_i(\bar{x}) = f_i(x_1, \ldots, x_N) = 0 \qquad i = 1, \ldots, N$$

can be expanded in a Taylor's series at an assigned point as

$$f_i(\bar{x}) = f_i(\bar{x}^0) + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j}\bigg|_{xj}^{0} (x_j - x_j^0) + \text{higher order term} = 0$$

Suppose $\bar{x}^0$ is sufficiently close to the solution of this system; the higher order term may be neglected. Then we have the following linear system of equations:

$$f_i(\bar{x}^0) + \sum_{j}^{N} \frac{\partial f_i}{\partial x_j}\bigg|_{xj}^{0} \bar{\epsilon}_j^0 \cong 0 \qquad i = 1, \ldots, N$$

where $\epsilon_j^0 = x_j - x_j^0$

$$\bar{\epsilon}^0 = (\epsilon_1^0, \ldots, \epsilon_N^0)$$

Let $\overline{\epsilon^0}$ be the solution of the above system of linear equations. Then $\overline{x}^1 = \overline{x}^0 + \overline{\epsilon}^0$ will be an approximate solution to the system of the nonlinear equations.

In general, let $\overline{x^{i+1}} = \overline{x^i} + \overline{\epsilon^i}$; then $x^{i+1}$ will be used as the assigned point for the next iteration. The convergence of this technique is very much dependent upon the behavior of the function $f_i$ in the neighborhood of the solution and the nearness of the first assigned value $x^0$ to the solution. The idea of this technique will be most easily understood by considering the following one-dimensional example.

Let $f(x) = f(x^0) + \frac{df}{dx}\Big|_{x^0} (x - x^0) + \frac{d^2f}{2!|dx^2.}\Big|_{x^0} (x - x^0)^2 + \ldots \eqsim 0$

Suppose $\overline{x}^0$ is sufficiently close to the solution; then we have the following linear approximation:

$$f(x^0) + \frac{df}{dx}\,_{x^0}\ \epsilon^0 \cong 0$$

where $\epsilon^0 = x - x^0$. Thus, $\epsilon^0 = \dfrac{-f(x^0)}{\frac{df}{dx}\,_{x^0}}$

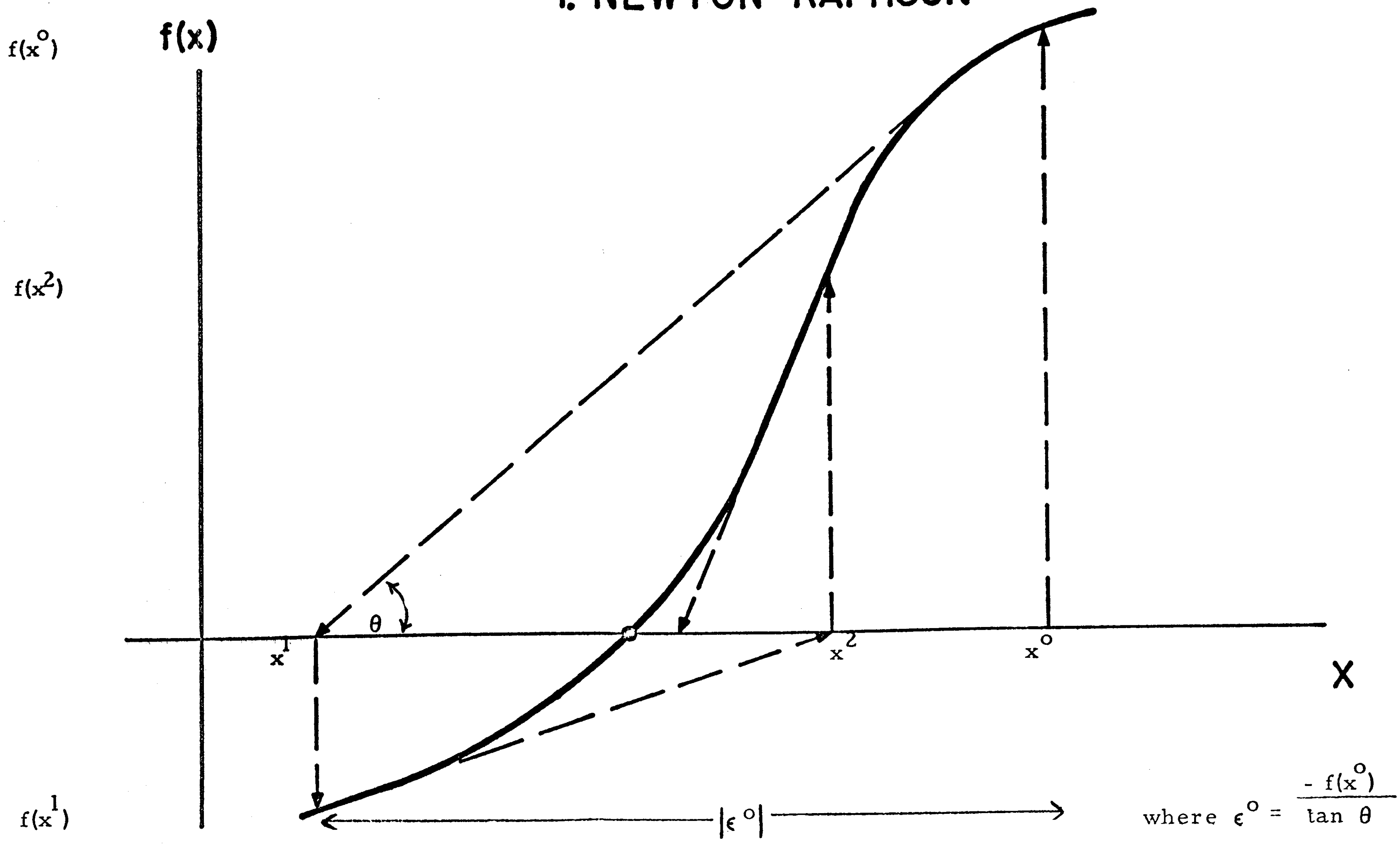
The approximate solution to the nonlinear equation is $x^1 = x^0 - \dfrac{f(x^0)}{\frac{df}{dx}\,_{x^0}}$

Assume $f(x)$ as indicated in Figure 2. As we can see, $x^1$ is closer to the solution of $f(x) = 0$. In general, let $x^{i+1} = x^i + \epsilon^i$; then the successive iterations could generate $x^{i+1}$ that are even closer to the solution.

This method has been very successfully applied to the system of four nonlinear equations in one circuit. Unfortunately this method shows no sign of convergence for the thirteen nonlinear equations of other circuits.

# DESIGN METHOD

## 1. NEWTON-RAPHSON



where $\epsilon^o = \dfrac{-f(x^o)}{\tan\theta}$

Figure 2.

Consequently, this method can serve as a convergence test of the possible convergence of other iterative methods.

## The General Case

Let $f_i = 0$ represent Kirchhoff's circuit equations and define $G = \Sigma f_i^2$. Hence, the circuit design problem is solved by finding the component parameter values within their boundary conditions such that the equation G = 0 is satisfied. The search techniques were used for this method. The G function is minimized by these techniques along a path determined by exploring the relations between the G function and all the circuit parameter variables. This method is well adapted to the computer because of the simplicity in programming. It also eliminates the unnecessary assumption in the limited case method in order to obtain a determined system.

Two search techniques are employed in the program, direct search and orthogonal search. In the method of direct search, the G function is first reduced by exploring each individaal variable in the function. A direction is established after exploring all the variables. The G function is then reduced by moving along this direction until it fails to reduce any further. The process is then repeated by exploring each individual variable again in order to find a new direction. This method is well adapted to consideration of limits of each variable, and it is simple for computer programming. The method will fail if the contour of the residual function has the form shown in Figure 3, which shows that no direction can be found by exploring each individual variable in order to reduce the G function.

The method of orthogonal search is based on the same principle as the method of direct search. The only difference is that the search does not proceed along directions parallel to each individual coordinate. The search is along all the orthonormal coordinates defined by a feasible direction. The feasible

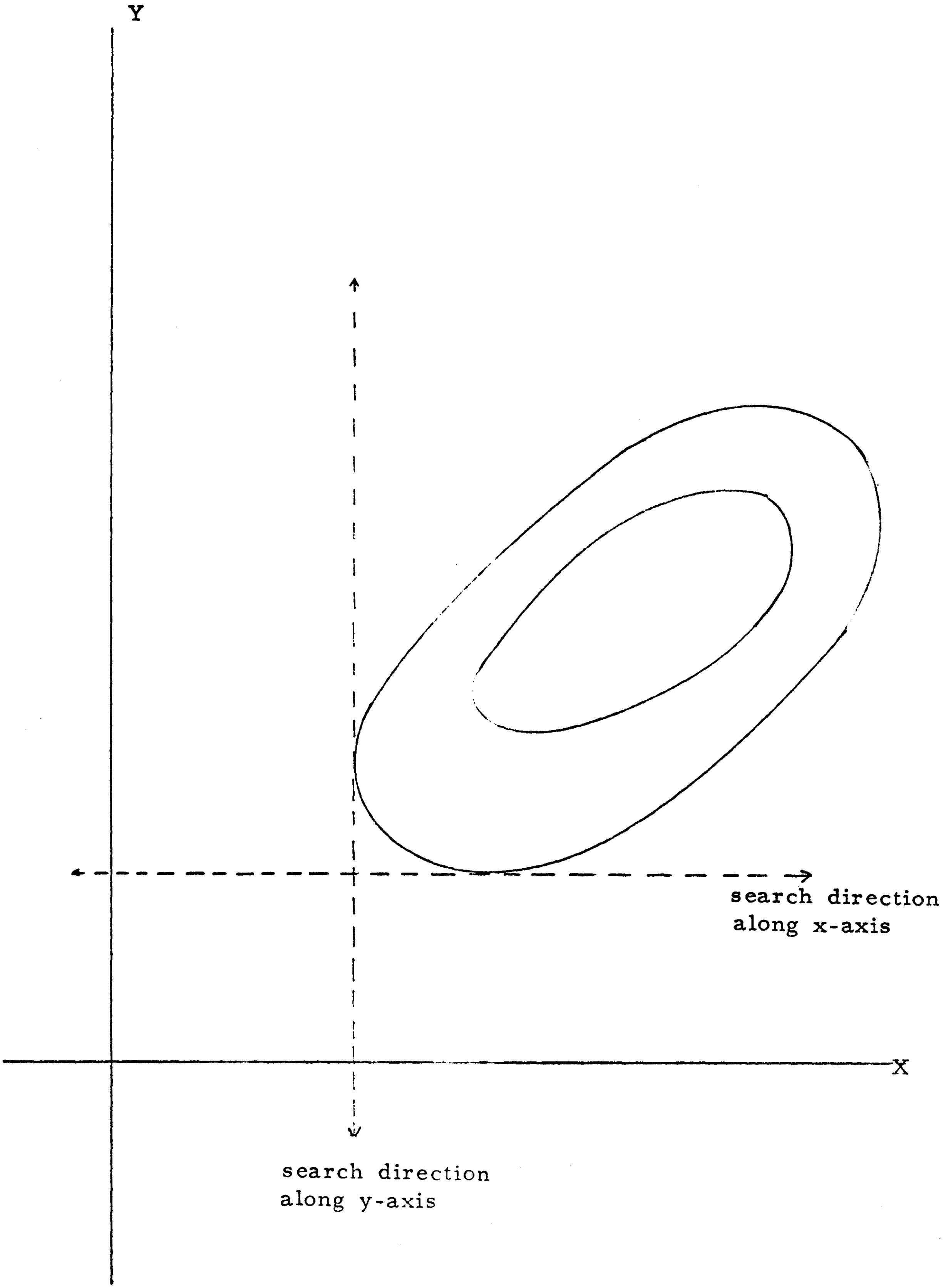

Y
X
search direction along x-axis
search direction along y-axis

Figure 3.

direction can be obtained by the method of direct search or it can be arbitrarily defined. The ortnonormal coordinates will be obtained by the Gram-Schmidt orthogonalization process as follows.

Let $A_1$ represent the feasible direction:

$$A_1 = (a, \dots, a_N)$$

The other N-1 vectors can be defined as follows:

$$A_i = (0, \dots, a_i, a_{i+1}, \dots, a_N)$$

$$\vdots$$

$$A_N = (0, \dots, 0, a_N)$$

Let $B_1 = A_1$ and $\epsilon_1 = B_1 / |B_1|$

Then the orthonormal coordinates $\epsilon_i$ can be obtained by

$$B_i = A_i - \sum_{j=1}^{i-1} (A_i \cdot \epsilon_j) \epsilon_j$$

$$\epsilon_i = B_i / |B_i|$$

This process can be easily understood by considering the following two-dimensional example. See Figure 4.

Let $A_1 = (DX, DY)$

$$A_2 = (0, DY)$$

$$B_1 = A_1$$

$$\epsilon_1 = B_1 / |B_1|$$

Then

$$B_2 = A_2 - (A_2 \cdot \epsilon_1) \epsilon_1$$

$$\epsilon_2 = B_2 / |B_2|$$

Therefore,

$\epsilon_1$ and $\epsilon_2$ are the new orthonormal coordinates.

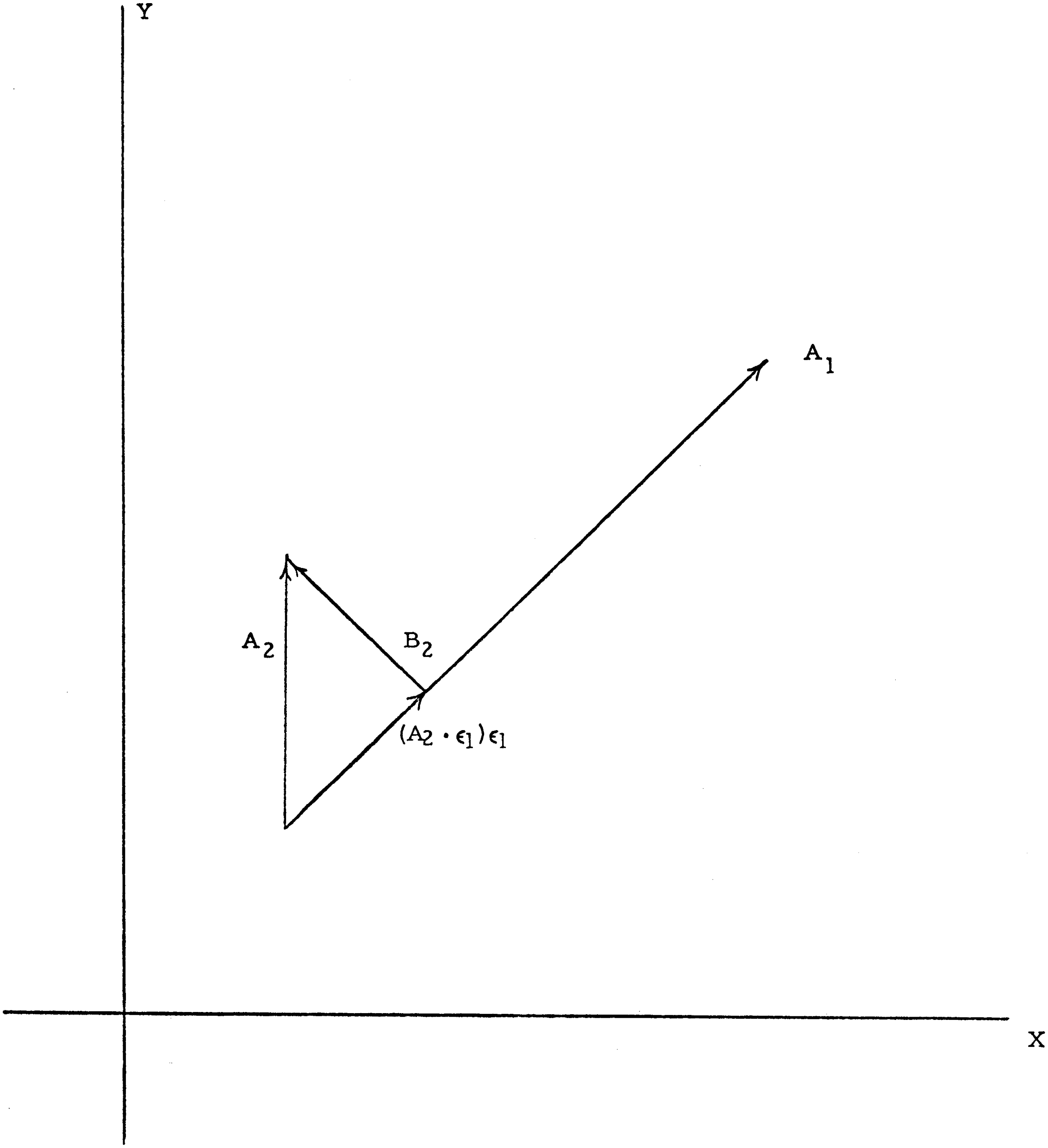


Figure 4.

The advantage of this method over the direct search method is that the problem of interaction between variables is reduced. This means that all variables are being changed simultaneously instead of one at a time during the process of reducing the G function. The disadvantages are: how to select suitable step sizes in exploring along the orthonormal directions and how to handle the limits for the variables.

## COMPUTER PROGRAM

The flow chart of this program is shown in Figure 5. The available memory size of computers is different.t To avoid the difficulty of overflowing the machine, the flow chart above the dotted line can be carried out by hand.

Example

A voltage mode switching circuit consists of four resistors, two power supplies, one transistor, one diode, and two operating states. The circuit diagram is shown in Figure 6. The design requirements are as follows:

1. The input impedance of this circuit has to be larger than a specified value.
2. The input voltage and its noise level are designed to satisfy a given range of variation.
3. The output voltage, current, and voltage noise level are designed to satisfy a given range of variation.
4. Resistors are designed to meet the given tolerances.
5. The power supplies are designed to satisfy a given tolerance and range of variation.
6. The power dissipation of the circuit is designed to be less than a specified value.
7. The circuit uses the specified transistor and diode.

# PROGRAM FLOW CHART

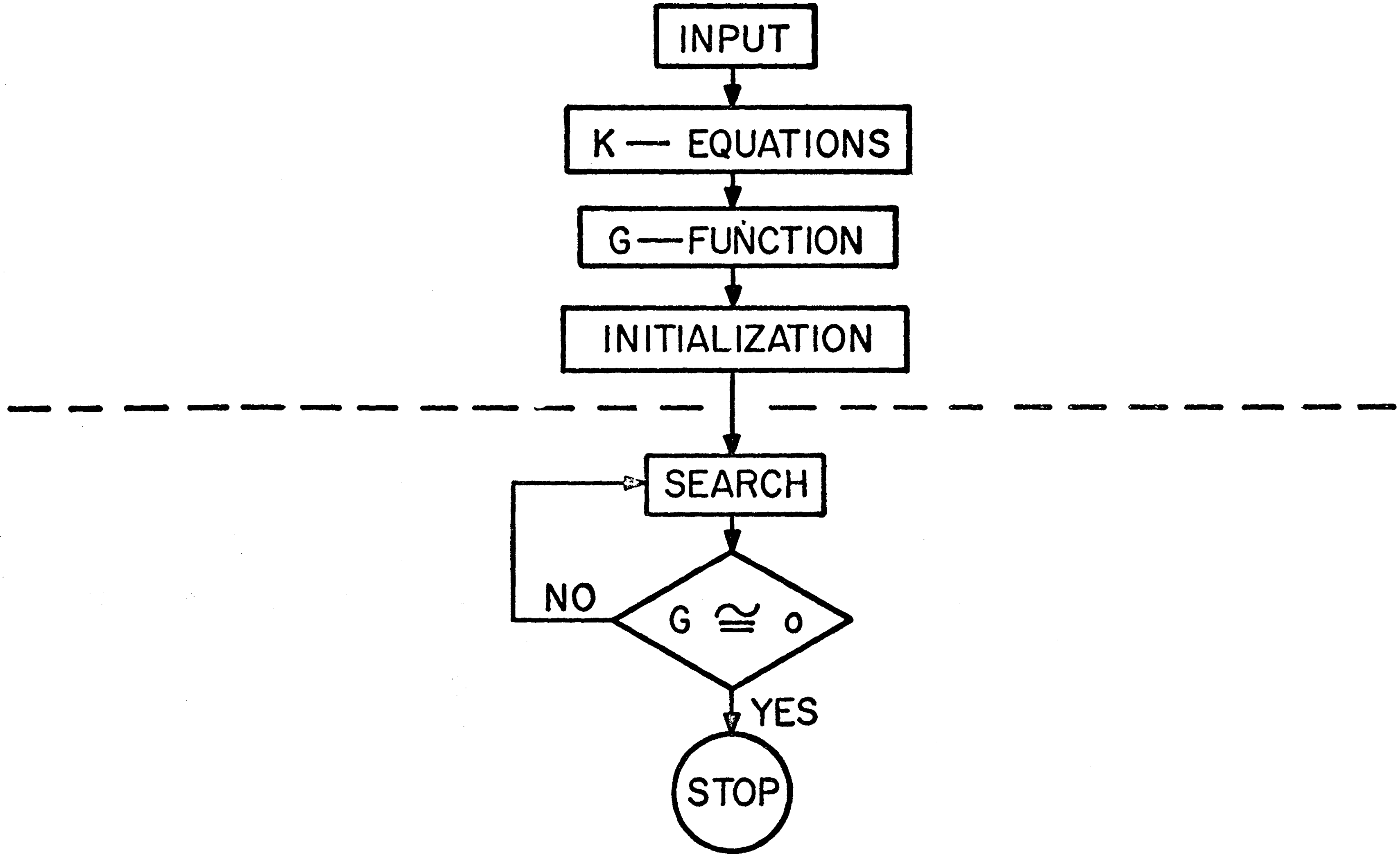


Figure 5.

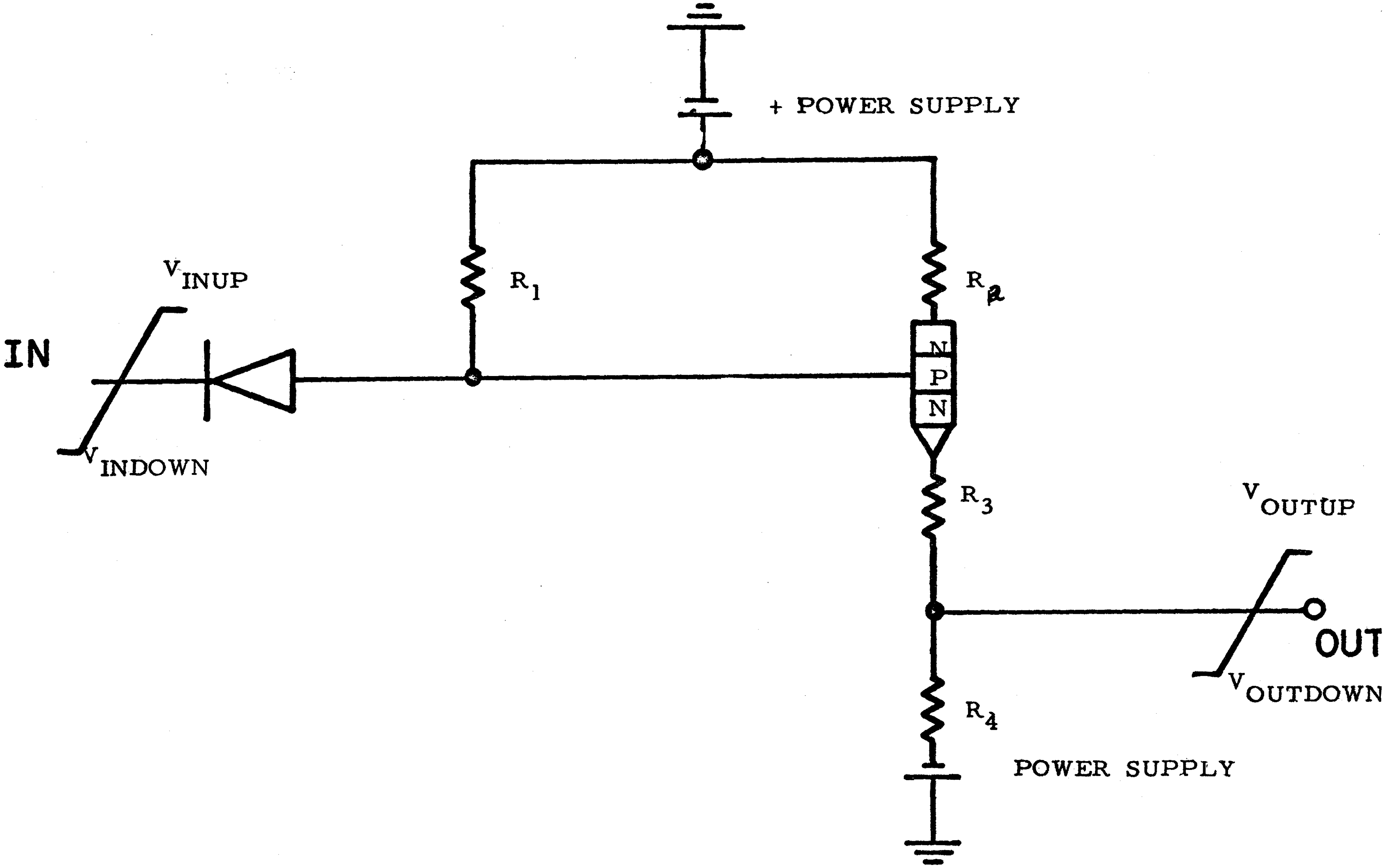

+ POWER SUPPLY
V INUP
IN
V INDOWN
R1
R2
N
P
N
R3
V OUTUP
OUT
V OUTDOWN
R4
POWER SUPPLY

Figure 6.

The linear piecewise approximation of the characteristics of diode and transistor was used in this program as shown in Figure 7.

The system of Kirchhoff's equations consists of twenty-two variables and fourteen equations. The result obtained by this method was successful. The size of the system can be reduced by considering either the independent loop or node equations with additional constraints. The result of using independent loop equations, which consist of sixteen variables and nine equations, is also obtained by this method. Because of the dimension of this design problem, it is difficult to give a graphical interpretation. However, if we consider the solution region defined by input impedance, input noise level, and output noise level, then the feasible solution region is the interception of the solution region defined by other design constraints. The feasible design is obtained by locating a point in the feasible solution region as indicated in Figure 8. The trade-off relations can be demonstrated by repeatedly applying this method with the variation of the design constraints.

## CONCLUSION

This method has been successfully applied to the current mode circuit as well as the voltage mode circuit. The scope of this method is not limited to the feasible design of a circuit. The continuous application of this method will lead to an optimum design by simply modifying the G function as follows:

$$G = W_k f_k^2 + f_M$$

where $f_k$ is the system of the Kirchhoff's equations;

$w_k$ is the weight factor used to keep the solution in the feasible design region;

$f_M$ is the desired function, under optimization.

In view of the optimum design, the importance of the elimination of the unnecessary assumptions is even more significant.

# DIODE CHARACTERISTIC

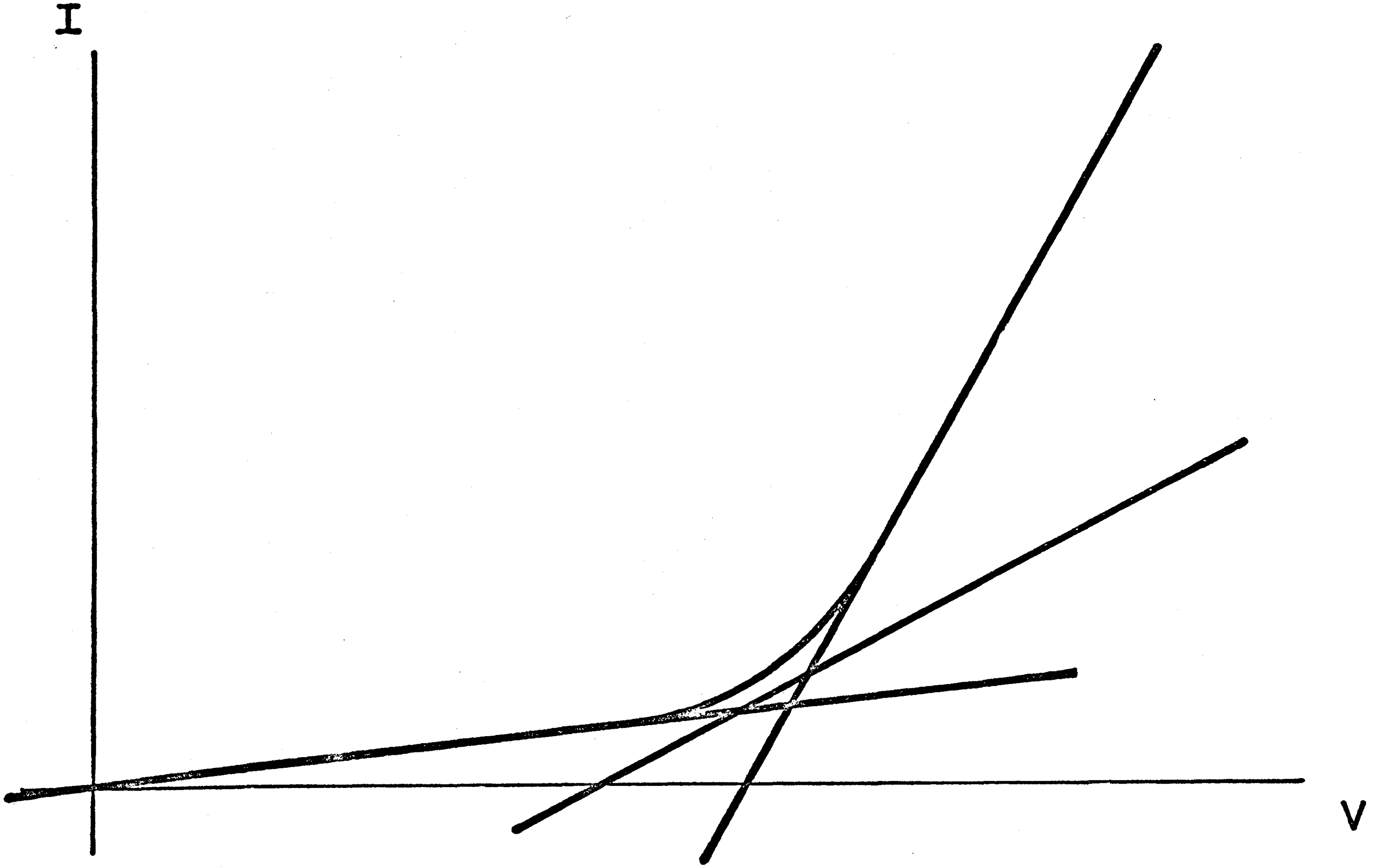


Figure 7.

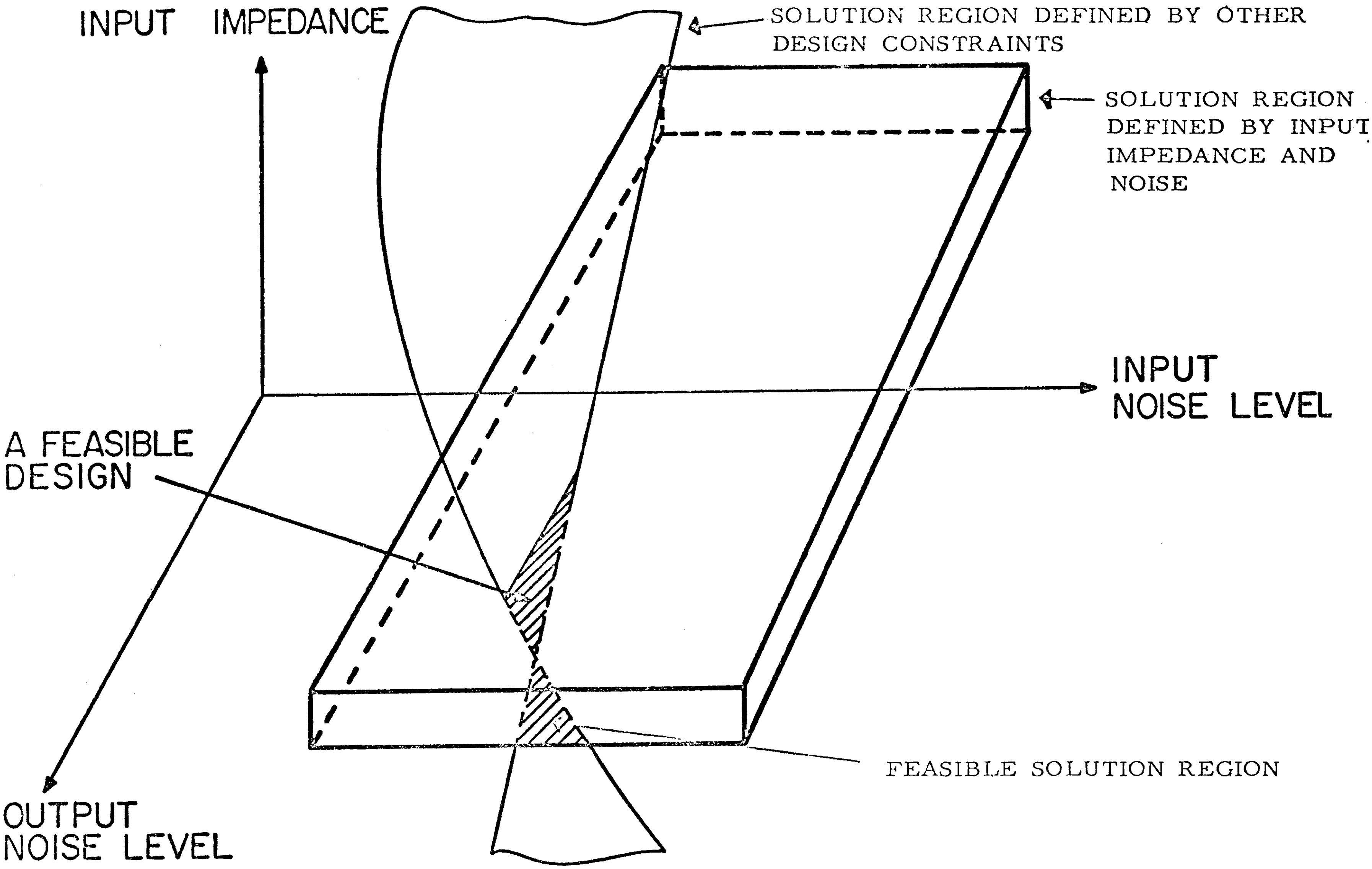

INPUT IMPEDANCE
SOLUTION REGION DEFINED BY OTHER DESIGN CONSTRAINTS
SOLUTION REGION DEFINED BY INPUT IMPEDANCE AND NOISE
INPUT NOISE LEVEL
A FEASIBLE DESIGN
FEASIBLE SOLUTION REGION
OUTPUT NOISE LEVEL

Figure 8

## ACKNOWLEDGMENT

The authors would like to acknowledge Mr. A. Brown and Mr. R. Silveri for their many helpful suggestions.

## BIBLIOGRAPHY

1. Rosenbrock, H., (1960) "An Automatic Method for Finding the Greatest or Least Value of a Function," The Computer Journal, Vol. 3, p. 175.

2. Wilde, D., (1964) "Optimum Seeking Method," Prentice Hall, Inc.

# The Solution of Laplace's Equation in Two Dimensions

by

Oscar N. Garcia, Old Dominion College

Introduction: Partial differential equations of the second order and first degree of the type:

$$A\frac{\partial^2 V}{\partial x^2}+B\frac{\partial^2 V}{\partial x\partial y}+C\frac{\partial^2 V}{\partial y^2}=f(x,y,V,\frac{\partial V}{\partial x},\frac{\partial V}{\partial y})$$

are said to be of the type called elliptic if $B^2 - 4AC < 0$
parabolic if $B^2 - 4AC = 0$
hyperbolic if $B^2 - 4AC > 0$.

The equation that occupies our attention here is of the first type (Poisson's equation):

$$\frac{\partial^2 V}{\partial x^2}+\frac{\partial^2 V}{\partial y^2}=\rho$$

where $\rho$ will be taken to be zero to yield Laplace's equation:

$$\frac{\partial^2 V}{\partial x^2}+\frac{\partial^2 V}{\partial y^2}=0$$

Analysis of the Problem. The numerical solution of this problem for the case where no tractable solution exists is going to consist of two main parts: 1) setting up the difference equations, and 2) solving the set of linear equations originated in (1).

An analysis of the errors which occurr in 1) and 2) and those due to round-off should also be included in the solution.

The program presented here was designed with such an error study in mind, but the results are not part of this presentation and are not complete at the time of this writing.

Setting up of the difference equations. A Taylor series expansion of a function V (x,y) about a point $P_o(x_o,y_o)$ using a $\Delta x = h$, $\Delta y = 0$ is:

$$V(x_0+h,y_0)=V(x_0,y_0)+\frac{\partial V}{\partial x}\Big|_{x_0,y_0}\frac{h}{1!}+\frac{\partial^2 V}{\partial x^2}\Big|_{x_0,y_0}\frac{h^2}{2!}+\frac{\partial^3 V}{\partial x^3}\Big|_{x_0 y_0}\frac{h^3}{3!} \quad (1)$$

or:

$$V(x_0+h,y_0)=\sum_{n=0}^{\infty}\left(\frac{\partial^n V}{\partial x^n}\right)\Big|_{x_0,y_0}\frac{h^n}{n!}$$

Similarly for $\Delta x = -h$, $\Delta y = 0$

$$V(x_0+h,y_0)=\sum_{n=0}^{\infty}\left(\frac{\partial^n V}{\partial x^n}\right)\Big|_{x_0,y_0}\frac{(-h)^n}{n!} \quad (2)$$

Truncating (1) and (2) at n = m we can say that

$$V(x_0+h,y_0)=\sum_{n=0}^{m-1}\left(\frac{\partial^n V}{\partial x^n}\right)\Big|_{x_0,y_0}\frac{h^n}{n!}+\frac{\partial^m V}{\partial x^m}\Big|_{\xi,y_0}\frac{h^m}{m!} \quad (3)$$

where

$$x_0<\xi<x_0+h$$

*and:*

$$V(x_0-h,y_0)=\sum_{n=0}^{m-1}\left[\left(\frac{\partial^n V}{\partial x^n}\right)\Big|_{x_0,y_0}\frac{(-h)^n}{n!}\right]+\frac{\partial^m V}{\partial x^m}\Big|_{\xi,y_0}\frac{(-h)^m}{m!} \qquad (4)$$

where $x_0 - h < \xi < x_0$.

If we add (3) and (4), the terms for odd values of n are eliminated

$$V(x_0+h,y_0)+V(x_0-h,y_0)=2\sum_{N=0}^{m-1}\left(\frac{\partial^N V}{\partial x^N}\right)\Big|_{x_0,y_0}\frac{h^N}{N!}$$
$$+\frac{\partial^m V}{\partial x^m}\Big|_{\xi,y_0}\frac{h^m}{m!}+\frac{\partial^m V}{\partial x^m}\Big|_{\xi,y_0}\frac{(-h)^m}{m!} \qquad (5)$$

where we assume only even values for N (0,2,4...) and a continuous m$^{th}$ derivative about $(x_0, y_0)$ as before.

The usual approach is to truncate this series at the value m = 4. Further investigations, using the methods outlined here, are intended for m = 6 and m = 8 to determine the effect of the truncation error in the solution. There seems to be a prevalent opinion among researchers in this area that the added complexity in the finite difference approximation for $m > 4$ overshadows any gain in accuracy, and use of a smaller value of h is usually preferred.

For m = 4, equation (5) becomes:

$$V(x_0+h,y_0)+V(x_0-h,y_0)=2\left(V(x_0,y_0)+\frac{\partial^2 V}{\partial x^2}\Big|_{x_0,y_0}\frac{h^2}{2!}\right)$$
$$+\frac{h^4}{4!}\left(\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}+\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}\right) \qquad (6)$$

Solving for $\frac{\partial^2 V}{\partial x^2}\Big|_{x_0,y_0}$ in (6):

$$\frac{\partial^2 V}{\partial x^2}\Big|_{x_0,y_0}=\frac{2}{h^2}\Big[\frac{V(x_0+h,y_0)+V(x_0-h,y_0)}{2}-V(x_0,y_0)$$
$$-\frac{1}{2}\frac{h^4}{4!}\left(\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}+\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}\right) \qquad (7)$$

where evidently the first two terms give the difference between the values of V at two points symmetrical with respect to $P_0$ $(x_0, y_0)$, and the value of V at $P_0$.

Simplifying (7) we obtain one of the two approximations for the two dimensional case:

$$\frac{\partial^2 V}{\partial x^2}\Big|_{x_0,y_0}=\frac{V(x_0+h,y_0)+V(x_0-h,y_0)-2V(x_0,y_0)}{h^2}+\frac{h^2}{4!}\left(\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}+\frac{\partial^4 V}{\partial x^4}\Big|_{\xi,y_0}\right) \qquad (8)$$

Similarly an expression for $\frac{\partial^2 V}{\partial y^2}\Big|_{x_0,y_0}$ could be found:

$$\left.\frac{\partial^2 V}{\partial y^2}\right|_{x_0,y_0} = \frac{V(x_0,y_0+h)+V(x_0,y_0-h)-2V(x_0,y_0)}{h^2} + \frac{h^2}{4}\left(\left.\frac{\partial^4 V}{\partial y^4}\right|_{x_0,\eta} + \left.\frac{\partial^4 V}{\partial y^4}\right|_{x_0,\mu}\right) \quad (9)$$

Neglecting the $\frac{h^2}{4!}$ terms in (8) and (9) and adding to obtain Laplace's equation we find that:

$$V(x_0+h,y_0) + V(x_0-h,y_0) + V(x_0,y_0+h) + V(x_0,y_0-h) - 4V(x_0,y_0) = 0 \quad (10)$$

which is the basic difference equation to be used.

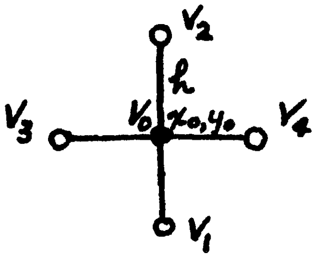


Figure 1.

Using the symbolism of Figure 1, equation (10) may be written as:

$V_1 + V_2 + V_3 + V_4 - 4V_o = 0.$

More generally for a point not on the boundary of the k x $\ell$ grid in Figure 2, we have:

$$V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - 4V_{i,j} = 0 \quad (11)$$

which is called the 5 point approximation.

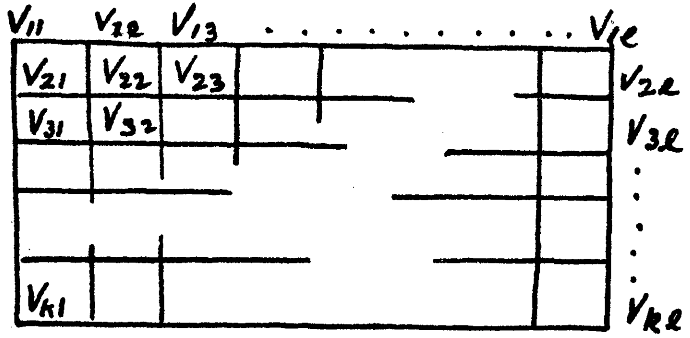


Figure 2.

If we try to solve for all internal points on the rectangular grid of Figure 2, we have a system of linear equations for each i =2, 3, ..., (k - 1) when j = 2, 3, ... ($\ell$ - 1). As an example Figure 3 shows the coefficients for the different values of $V_{i,j}$ corresponding to k = 5, $\ell$ = 5. If the values of $V_{i,j}$ are specified at the (rectangular) boundary, as they usually are, the terms whose coefficients are circled in Figure 3 may be added to form a constant and the matrix of coefficients of the system of equations now looks like Figure 4. It should be noticed that this is a symmetrical matrix. If the four points about $V(x_o,y_o)$ were not at a distance h from it, the symmetry of the matrix is not assured.

| i | j | 1,2 | 1,3 | 1,4 | 2,1 | 2,2 | 2,3 | 2,4 | 2,5 | 3,1 | 3,2 | 3,3 | 3,4 | 3,5 | 4,1 | 4,2 | 4,3 | 4,4 | 4,5 | 5,2 | 5,3 | 5,4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | ⓪ | | | ⓪ | -4 | 1 | | | | 1 | | | | | | | | | | | | = | 0 |
| 2 | 3 | | ⓪ | | | 1 | -4 | 1 | | | | 1 | | | | | | | | | | | = | 0 |
| 2 | 4 | | | ⓪ | | | 1 | -4 | ⓪ | | | | 1 | | | | | | | | | | = | 0 |
| 3 | 2 | | | | | 1 | | | | ⓪ | -4 | 1 | | | | 1 | | | | | | | = | 0 |
| 3 | 3 | | | | | | 1 | | | | 1 | -4 | 1 | | | | 1 | | | | | | = | 0 |
| 3 | 4 | | | | | | | 1 | | | | 1 | -4 | ⓪ | | | | 1 | | | | | = | 0 |
| 4 | 2 | | | | | | | | | | 1 | | | | ⓪ | -4 | 1 | | | ⓪ | | | = | 0 |
| 4 | 3 | | | | | | | | | | | 1 | | | | 1 | -4 | 1 | | | ⓪ | | = | 0 |
| 4 | 4 | | | | | | | | | | | | 1 | | | | 1 | -4 | ⓪ | | | ⓪ | = | 0 |

Fig. 3 (Circles Indicate Boundary Points)

| i | j | 2,2 | 2,3 | 2,4 | 3,2 | 3,3 | 3,4 | 4,2 | 4,3 | 4,4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | -4 | 1 | | 1 | | | | | | = | $K_1$ |
| 2 | 3 | 1 | -4 | 1 | | 1 | | | | | = | $K_2$ |
| 2 | 4 | | 1 | -4 | | | 1 | | | | = | $K_3$ |
| 3 | 2 | 1 | | | -4 | 1 | | 1 | | | = | $K_4$ |
| 3 | 3 | | 1 | | 1 | -4 | 1 | | 1 | | = | $K_5$ |
| 3 | 4 | | | 1 | | 1 | -4 | | | 1 | = | $K_6$ |
| 4 | 2 | | | | 1 | | | -4 | 1 | | = | $K_7$ |
| 4 | 3 | | | | | 1 | | 1 | -4 | 1 | = | $K_8$ |
| 4 | 4 | | | | | | 1 | | 1 | -4 | = | $K_9$ |

Fig. 4

Solution of the System of Equations. The system of equations set in Figure 4 may be solved by any of a number of methods. A survey of the literature shows a wide variety of approaches with their relative merits and drawbacks.

Figure 5 is a chart that demonstrates the diversity of approaches. One of the most important features of any method, however, is its simplicity and the availability of a good body of theory behind it validating the approach.

- Methods of Solution for the Systems of Equations
  - Point Iterative
    - Gauss-Seidell (successive displacement)
    - Jacobi (simultaneous displacements)
    - Successive Overrelaxation (Liebmann's when applied to Laplace's equation)
    - Richardson's Method
    - Sheldon's Method
  - Line Iterative
    - Successive Row Iteration
    - Simultaneous Row Iteration
    - Successive Line Overrelaxation
  - Alternating-direction Implicit
    - Peaceman-Rachford Method
    - Douglas-Rachford Method
  - Block Iteration
    - Simultaneous Block Iteration
    - Successive Block Iteration

Figure 5.

A very simple method of the relaxation type ("Introduction to Engineering Analysis", IBM F20-8077-1, page 96) in a modified version, has been used to solve Laplace's equation when the boundary conditions are numerically specified at a number of regular points in a closed polygonal perimeter. This modified program is shown in Appendix A.

A method which has the desired characteristics described above is the Gauss-Seidel method. At the same time, convergence of the iteration process used in this method is assured if the sum of the absolute magnitude of the coefficients of the non-diagonal elements is equal to, or less than, the magnitude of the corresponding diagonal elements, with the inequality holding for at least one equation. We see that this is the case in Figure 4). Furthermore, since it is not necessary to store the "residuals" R(I,J), more storage is now available for the solution of a larger net. A net of somewhat more than 400 points may be solved in the basic 1620. A program using this approach has been written and is given in Appendix B. It has been found that for the same net of 72 points this program runs in better than two thirds of the time taken by the program in

Appendix A for the same accuracy. Higher gains are expected for larger nets. (In the block relaxation program of Appendix A, twenty iterations were necessary while the Gauss-Seidel program required only sixteen).

Conclusions. It has been shown that Laplace's equation may be approximately by finite differences leaving an error term of the order of $h^2$. Furthermore this set of approximate difference equations are usually solved using iterative techniques. Two of such techniques, chosen primarily because of their simplicity, were considered. The estimation of how well the finite difference equations approximate the Taylor series, is a complex one. On one hand, although a maximum error bound may be found, this involves an estimate of the fourth order derivatives of the function which complicates the computations. The alternate possibility, on the other hand, is to decrease the size of h by a factor $[k] < 1$ resulting in the increment by a factor of $[\frac{1}{k}]^2$ in the number of points of the grid, and therefore, more usage of machine time for the solution of the larger system. There seems to be some preference given to the latter solution. This may be partially due, perhaps, to the scarcity of literature in English and of examples of studies using the former approach.

## APENDIX A

```
      DIMENSION V(30,8),R(30,8),IX(30,2)
   21 READ1,M,DEL
      DO3I=1,M
      READ2,V(I,1),V(I,2),V(I,3),V(I,4),V(I,5),V(I,6),V(I,7),V(I,8),J,K
      IX(I,1)=J
    3 IX(I,2)=K
      DO 20 I=1,M
      DO 20 J=1,8
   20 R(I,J)=0.
      L=M-1
      DO13 I=2,L
      L1=IX(I,1)+1
      L2=IX(I,2)-1
      DO 13 J=L1,L2
   13 R(I,J)=V(I-1,J)+V(I+1,J)+V(I,J-1)+V(I,J+1)-4.*V(I,J)
      ITCT=0
    4 ITCT=ITCT+1
      K=2
      DO 9 I=2,L
      L1=IX(I,1)+1
      L2=IX(I,2)-1
      DO 9  J=L1,L2
      RAB=R(I,J)
C
```

```
   IF (RAB) 10,7,7
10 RAB=-R(I,J)
 7 IF(RAB-DEL)9,9,8
 8 K=1
   RDEL=0.25*R(I,J)
   V(I,J)=V(I,J)+RDEL
   R(I,J)=0.
   R(I-1,J)=R(I-1,J)+RDEL
   R(I+1,J)=R(I+1,J)+RDEL
   R(I,J-1)=R(I,J-1)+RDEL
   R(I,J+1)=R(I,J+1)+RDEL
 9 CONTINUE
   GO TO(4,11),K
11 PUNCH 1,ITCT
   DO 5 I=1,M
 5 PUNCH2, V(I,1),V(I,2),V(I,3),V(I,4),V(I,5),V(I,6),V(I,7),V(I,8)
   GO TO 21
 1 FORMAT(I5,F10.0)
 2 FORMAT(8F8.4,2I2)
   END
```

SAMPLE DATA

```
    9     0.01
       .       .       .       .       .      4.      3.      2.     6 8
       .       .       .       .       .      4.       .      2.     6 8
       .       .       .       .      4.       .       .      2.     5 8
       .       .       .      4.       .       .       .      2.     4 8
       .       .      4.       .       .       .       .      2.     3 8
       .       .      4.       .       .       .       .      2.     3 8
       .      4.       .       .       .       .       .      2.     2 8
      4.       .       .       .       .       .       .      2.     1 8
      4.    3.72    3.44    3.16    2.84    2.56    2.28      2.     1 8
```

RESULTS

```
   20
   .0000   .0000   .0000   .0000   .0000  4.0000  3.0000  2.0000
   .0000   .0000   .0000   .0000   .0000  4.0000  2.9363  2.0000
   .0000   .0000   .0000   .0000  4.0000  3.4820  2.7549  2.0000
   .0000   .0000   .0000  4.0000  3.6425  3.1759  2.6074  2.0000
   .0000   .0000  4.0000  3.7475  3.3970  2.9777  2.5024  2.0000
   .0000   .0000  4.0000  3.5997  3.2302  2.8390  2.4248  2.0000
   .0000  4.0000  3.7460  3.4250  3.0852  2.7303  2.3664  2.0000
  4.0000  3.8203  3.5683  3.2765  2.9593  2.6402  2.3197  2.0000
  4.0000  3.7200  3.4400  3.1600  2.8400  2.5600  2.2800  2.0000
```

## APENDIX B

```
      DIMENSIONV(50,8),IX(50,2)
   21 READ 1,M,DEL,ITEND
      DO3I=1,M
      READ2,V(I,1),V(I,2),V(I,3),V(I,4),V(I,5),V(I,6),V(I,7),V(I,8),J,K
      IX(I,1)=J
    3 IX(I,2)=K
      L=M-1
      ITCT=0
   14 K=0
      DO 9 I=2,L
      L1=IX(I,1)+1
      L2=IX(I,2)-1
      DO 9  J=L1,L2
      Z=0.25*(V(I+1,J)+V(I-1,J)+V(I,J+1)+V(I,J-1))
      DISC=Z-V(I,J)
      V(I,J)=Z
      IF(DISC) 10,7,7
   10 DISC=-DISC
    7 IF(DISC-DEL) 9,6,6
    6 K=K+1
    9 CONTINUE
      ITCT=ITCT+1
      IF(ITCT-ITEND) 17,17,11
   17 IF(K)15,11,14
   11 PUNCH 1,ITCT
      DO5I=1,M
    5 PUNCH2, V(I,1),V(I,2),V(I,3),V(I,4),V(I,5),V(I,6),V(I,7),V(I,8)
      GO TO 21
   15 STOP
    1 FORMAT(I5,F10.0,I5)
    2 FORMAT(8F8.4,2I2)
      END
```

SAMPLE DATA

```
    9    0.01     1000
 .        .        .        .        .       4.       3.        2.      6 8
 .        .        .        .        .       4.       .         2.      6 8
 .        .        .        .       4.       .        .         2.      5 8
 .        .        .       4.        .       .        .         2.      4 8
 .        .       4.        .        .       .        .         2.      3 8
 .        .       4.        .        .       .        .         2.      3 8
 .       4.        .        .        .       .        .         2.      2 8
4.        .        .        .        .       .        .         2.      1 8
4.       3.72     3.44     3.16     2.84    2.56     2.28       2.      1 8
```

RESULTS

```
   16
   .0000    .0000    .0000    .0000    .0000   4.0000   3.0000   2.0000
   .0000    .0000    .0000    .0000    .0000   4.0000   2.9381   2.0000
   .0000    .0000    .0000    .0000   4.0000   3.4791   2.7547   2.0000
   .0000    .0000    .0000   4.0000   3.6359   3.1697   2.6048   2.0000
   .0000    .0000   4.0000   3.7414   3.3879   2.9698   2.4989   2.0000
   .0000    .0000   4.0000   3.5923   3.2203   2.8340   2.4248   2.0000
   .0000   4.0000   3.7461   3.4216   3.0807   2.7303   2.3692   2.0000
  4.0000   3.8221   3.5705   3.2770   2.9595   2.6428   2.3230   2.0000
  4.0000   3.7200   3.4400   3.1600   2.8400   2.5600   2.2800   2.0000
```

# STRUCTURAL ANALYSIS USING THE 1620 COMPUTER

Tony A. Ross
University of Mississippi
University, Mississippi

## INTRODUCTION

Structural analysis, a familiar phrase to all civil engineers, is often a dull and repetitious undertaking, and much of this so-called analysis is simply "turning the crank." With the assistance of an automatic computer, the engineer is now able to devote more time to actual analysis, while the computer, with the proper instructions turns the crank.

The purpose of this paper is to develop a program to analyze various structures using the IBM 1620 computer. The program developed can be used on any computer capable of compiling a Fortran program, and the method presented will analyze most of the common structures encountered by the present-day civil engineer. Example problems have been worked to illustrate the versatility of the program, and a generalized step by step procedure is presented in order to simplify the preparation of the input data.

The slope deflection method of analysis is used in the program presented in this paper. C. K. Wang (1) presents a matrix formulation of the slope deflection equations, and

1. Wang, C. K. Matrix Formulations of Slope Deflection Equations. ASCE Transactions, 1958, Vol. 84, p. 1819.

from his presentation, the following matrices need to be completed:

Matrix — Description

[BI] - The relative I of each member

[BL] - The length of each member

[A] - A matrix whose elements are the coefficients by which the distributed end moments* are multiplied to obtain the balancing joint moments and sidesway forces

[FEM] - A matrix expressing the fixed end moments acting on each member

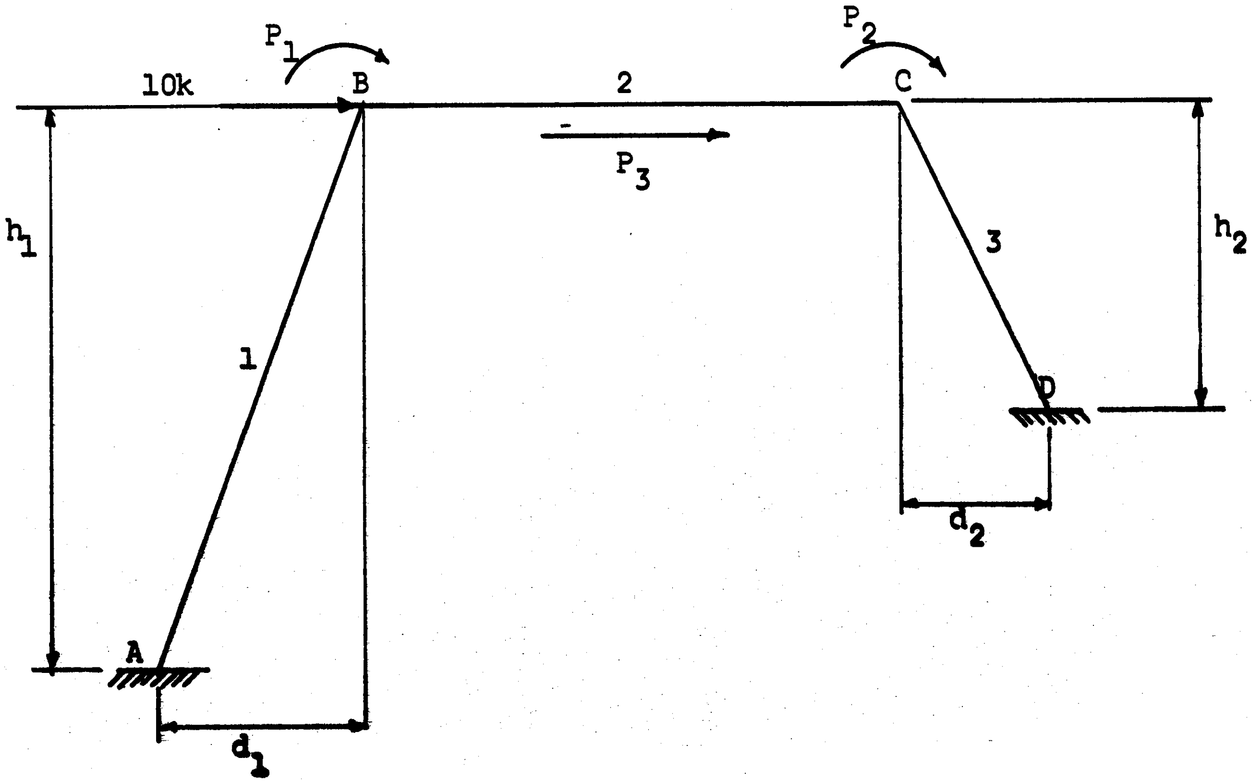
[P] - A matrix expressing the balancing sidesway forces acting at each joint or on each member

It now becomes our task to define each of these matrices. The following section contains a step by step procedure, and if followed closely, this will minimize the errors likely to be encountered in preparing the input data.

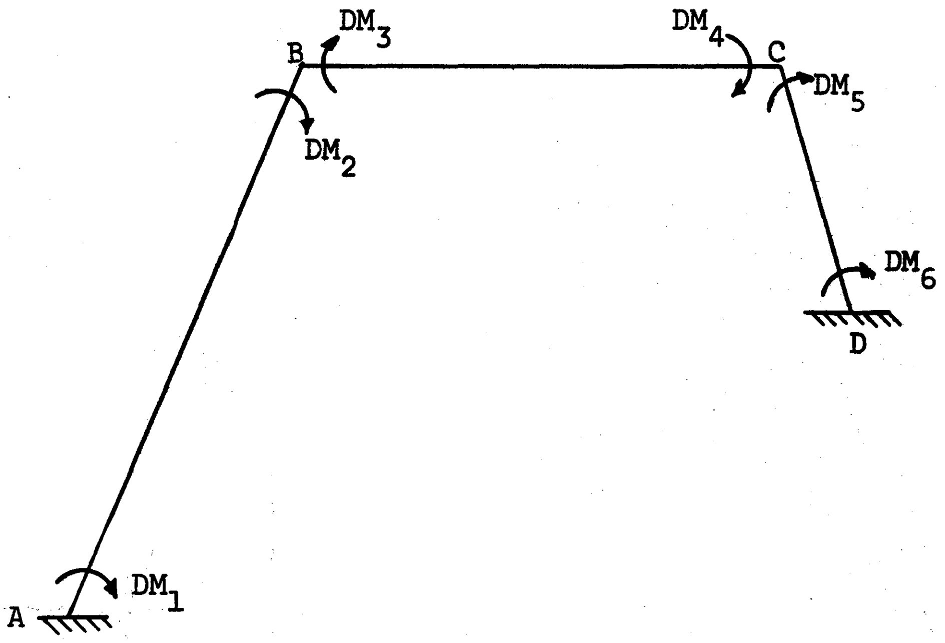
*Distributed end moments are the balancing moments distributed to the ends of the member such that the structure is held in static equilibrium under the action of the unbalanced end moments and sidesway forces.

## General Solution

This section contains a brief description of how to prepare the input data for the general bent shown below. This step by step procedure, if followed closely, will minimize the input errors that are likely to occur.

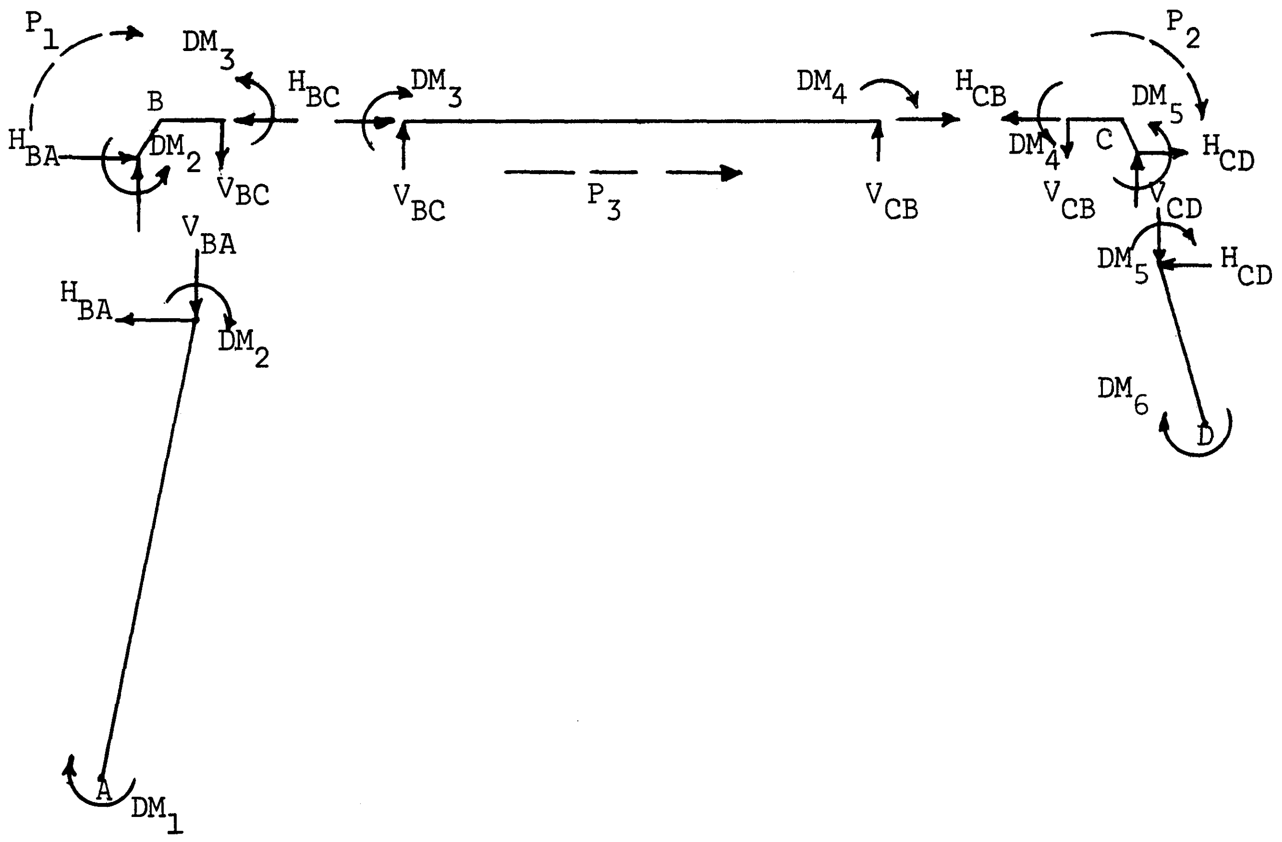


I. Draw a sketch of the structure to be analyzed. On this sketch, (1) number the members (1 through m), and determine the number of unknown joint rotations (j) and unknown sidesway displacements (s). (2) Draw the P forces and name them consecutively from 1 to n = (j+s).



II. Draw a second sketch and on this sketch draw the distributed end moments and number them from 1 to 2m. These moments do not include any fixed end moments due to loads or settlements.

III. Formulate the [A] matrix expressing the P forces in terms of the distributed moments. This matrix is formed by observing the freebody diagrams of steps I and II.



$$P_1 = DM_2 + DM_3$$

$$P_2 = DM_4 + DM_5$$

$$P_3 = -H_{BC} - H_{CB}$$

$$\sum M_a = 0 \qquad\qquad \sum M_c = 0$$

$$DM_1 + DM_2 + V_{BA}(d_1) - H_{BA}(h_1) = 0 \qquad\qquad DM_3 + DM_4 + V_{BC}(L_{BC}) = 0$$

$$H_{BA} = \frac{DM_1 + DM_2}{h_1} + \frac{d_1}{h_1} \; V_{BA} \qquad\qquad V_{BC} = - \frac{DM_3 + DM_4}{L_{BC}}$$

$$H_{BC} = H_{BA} \qquad\qquad V_{BC} = V_{BA}$$

$$H_{BC} = \frac{DM_1}{h_1} + \frac{DM_2}{h_1} - \frac{d_1}{h_1}\left(\frac{1}{L_{BC}}\right)(DM_3) - \frac{d_1}{h_1}\left(\frac{1}{L_{BC}}\right)(DM_4)$$

In similar manner,

$$H_{CB} = \frac{DM_5}{h_2} + \frac{DM_6}{h_2} - \frac{d_2}{h_2}\left(\frac{1}{L_{BC}}\right)(DM_3) - \frac{d_2}{h_2}\left(\frac{1}{L_{BC}}\right)(DM_4)$$

Therefore:

$$P_3 = - \frac{1}{h_1}DM_1 - \frac{1}{h_1}DM_2 + \frac{1}{L_{BC}}\left(\frac{d_1}{h_1} + \frac{d_2}{h_2}\right) DM_3 + \frac{1}{L_{BC}}\left(\frac{d_1}{h_1} + \frac{d_2}{h_2}\right) DM_4$$

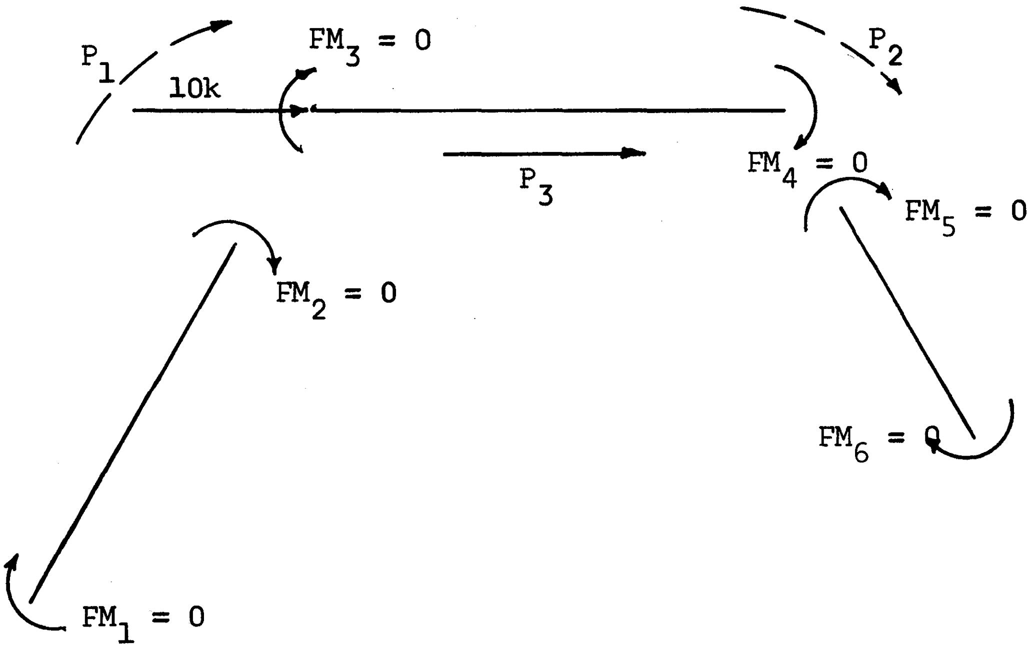
$$- \frac{1}{h_2}DM_5 - \frac{1}{h_2}DM_6$$

Now the [A] matrix becomes

| | $DM_1$ | $DM_2$ | $DM_3$ | $DM_4$ | $DM_5$ | $DM_6$ |
|---|---|---|---|---|---|---|
| $P_1$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $P_2$ | 0 | 0 | 0 | 1 | 1 | 0 |
| $P_3$ | $-\frac{1}{h_1}$ | $-\frac{1}{h_1}$ | $+\frac{1}{L_{BC}}\left(\frac{d_1}{h_1} + \frac{d_2}{h_2}\right)$ | $+\frac{1}{L_{BC}}\left(\frac{d_1}{h_1} + \frac{d_2}{h_2}\right)$ | $-\frac{1}{h_2}$ | $-\frac{1}{h_2}$ |

IV. Record moment of inertia and the length of each member.

V. Calculate the fixed end moments for each mamber and then record these values along with the P values resulting from these fixed end moments. (Note the P forces are the unbalanced force and not the balancing forces.)

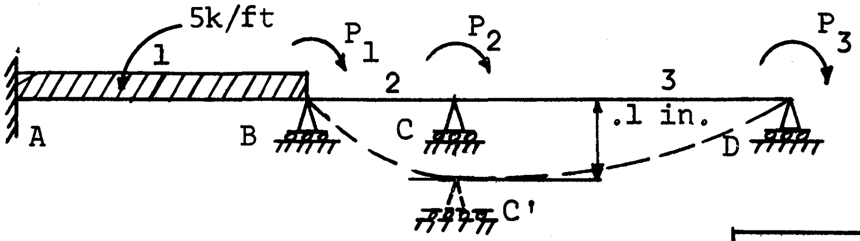


| | | | | | |
|---|---|---|---|---|---|
| $P_1$ | = | 0 | $FM_1$ | = | 0 |
| $P_2$ | = | 0 | $FM_2$ | = | 0 |
| $P_3$ | = | +10k | $FM_3$ | = | 0 |
| | | | $FM_4$ | = | 0 |
| | | | $FM_5$ | = | 0 |
| | | | $FM_6$ | = | 0 |

With the preceding steps in mind, four example problems are shown in detail along with computer input and results obtained.

## Example Problems

### Example 1



m = 3 n = 3

| Member | Rel. I's | Length (ft.) |
|---|---|---|
| 1 | 2.0 | 20.0 |
| 2 | 1.0 | 10.0 |
| 3 | 6.0 | 30.0 |

A (matrix)

|  | $DM_1$ | $DM_2$ | $DM_3$ | $DM_4$ | $DM_5$ | $DM_6$ |
|---|---|---|---|---|---|---|
| $P_1$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $P_2$ | 0 | 0 | 0 | 1 | 1 | 0 |
| $P_3$ | 0 | 0 | 0 | 0 | 0 | 1 |

Fixed end moments including moments due to settlement of support

|  |  |
|---|---|
| $FM_1$ | -166.5 |
| $FM_2$ | 166.5 |
| $FM_3$ | -124.9 |
| $FM_4$ | -124.9 |
| $FM_5$ | 46.0 |
| $FM_6$ | 121.0 |

| P (matrix) |  |
|---|---|
| $P_1$ | -41.6 |
| $P_2$ | 78.9 |
| $P_3$ | -121.0 |

Input Data

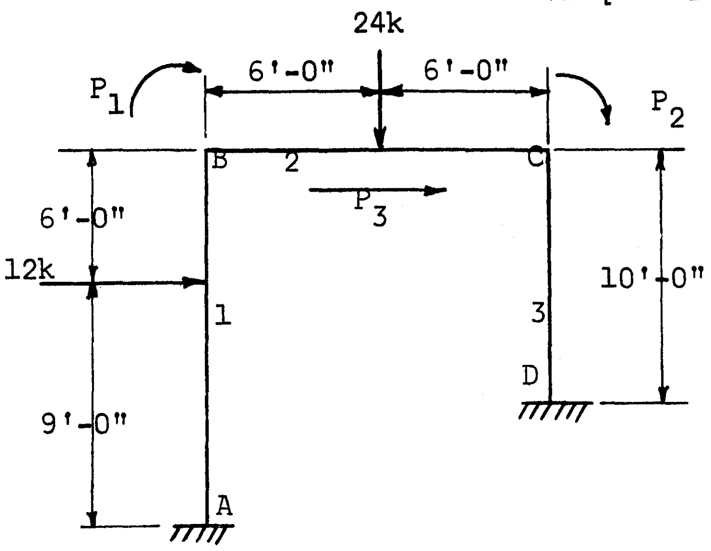
```
0303
2.0
1.0
6.0
20.0
10.0
30.0
1 01 02 1.0
1 01 03 1.0
1 02 04 1.0
1 02 05 1.0
2 03 06 1.0
1 01 -166.5
1 02 166.5
1 03 -124.9
1 04 -124.9
1 05 46.0
2 06 121.0
1 01 -41.6
1 02 78.9
2 03 -121.0
```

Output Data

Total End Moment

```
M1 = -184.78421
M2 =  129.93158
M3 = -129.93158
M4 =  -80.11053
M5 =   80.11053
M6 =     .00000
```

## Example 2



$m = 3 \quad n = 3$

| Member | Rel. I's | Length (ft.) |
|---|---|---|
| 1 | 1.0 | 15.0 |
| 2 | 2.0 | 12.0 |
| 3 | 1.0 | 10.0 |

A (matrix)

| | $DM_1$ | $DM_2$ | $DM_3$ | $DM_4$ | $DM_5$ | $DM_6$ |
|---|---|---|---|---|---|---|
| $P_1$ | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| $P_2$ | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| $P_3$ | -0.0666 | -0.0666 | 0.0 | 0.0 | -0.1 | -0.1 |

| Fixed End Moments | |
|---|---|
| $FM_1$ | -17.28 |
| $FM_2$ | 25.92 |
| $FM_3$ | -36.00 |
| $FM_4$ | 36.00 |
| $FM_5$ | 0.0 |
| $FM_6$ | 0.0 |

| P (matrix) | |
|---|---|
| $P_1$ | 10.08 |
| $P_2$ | -36.0 |
| $P_3$ | 7.777 |

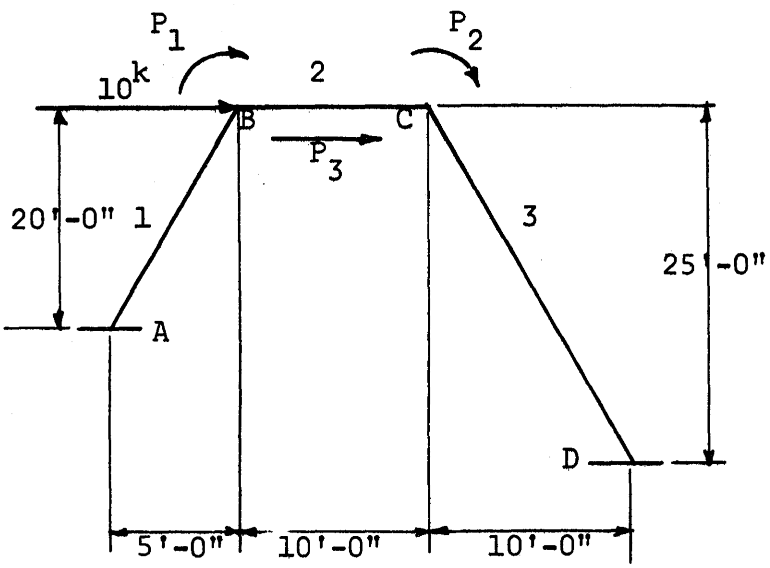
Input Data

```
0303
1.0
2.0
1.0
15.0
12.0
10.0
1 01 02 1.0
1 01 02 1.0
1 02 04 1.0
1 02 05 1.0
1 03 01 -0.0666
1 03 02 -0.0666
1 03 05 -0.10
2 03 06 -0.10
1 01 -17.28
1 02 25.92
1 03 -36.0
2 04 36.0
1 01 10.08
1 02 -36.0
2 03 7.777
```

Output Data
Total End Moment

```
M1 = -26.33809
M2 =  20.91634
M3 = -20.91634
M4 =  35.75900
M5 = -35.75900
M6 = -32.64587
```

Example 3



m = 3 n = 3

| Member | Rel. I's | Length (ft.) |
|---|---|---|
| 1 | 20.6 | 20.6 |
| 2 | 20.0 | 10.0 |
| 3 | 80.7 | 26.9 |

(Fixed end moments) $FM_i = 0.0$ for $i = 1, 2, \ldots 2m$

A (matrix)

|  | $DM_1$ | $DM_2$ | $DM_3$ | $DM_4$ | $DM_5$ | $DM_6$ |
|---|---|---|---|---|---|---|
| $P_1$ | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| $P_2$ | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| $P_3$ | -0.05 | -0.05 | 0.065 | 0.065 | -0.04 | -0.04 |

| P (matrix) |  |
|---|---|
| $P_1$ | 0.0 |
| $P_2$ | 0.0 |
| $P_3$ | 10.0 |

Input Data

```
0303
20.6
20.0
80.7
20.6
10.0
26.9
1 01 02 1.0
1 01 03 1.0
1 02 04 1.0
1 02 05 1.0
1 03 01 -0.05
1 03 02 -0.05
1 03 03 0.065
1 03 04 0.065
1 03 05 -0.04
2 03 06 -0.04
2 01 0.0
2 03 10.0
```

Output Data
Total End Moment

$M_1$ = -22.66109

$M_2$ = -27.59842

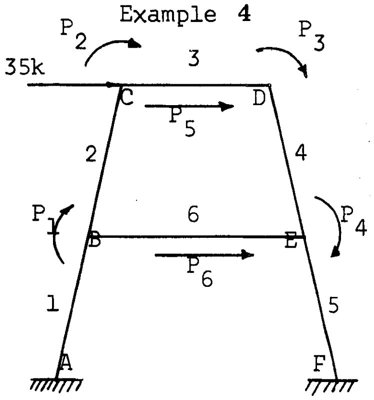
$M_3$ = 27.59842

$M_4$ = 38.73907

$M_5$ = -38.73907

$M_6$ = -40.63805

Example 4



m = 6     n = 6

| Member | Rel. I's | Length (ft) |
|---|---|---|
| 1 | 3.0 | 15.133 |
| 2 | 3.0 | 15.133 |
| 3 | 9.0 | 12.000 |
| 4 | 3.0 | 15.133 |
| 5 | 3.0 | 15.133 |
| 6 | 4.0 | 16.000 |

| Input Data | |
|---|---|
| 0606 | 1 04 09 1.0 |
| 3.0 | 1 04 12 1.0 |
| 3.0 | 1 05 03 -0.0667 |
| 9.0 | 1 05 04 -0.0667 |
| 3.0 | 1 05 05 0.0222 |
| 3.0 | 1 05 06 0.0222 |
| 4.0 | 1 05 07 -0.0667 |
| 15.133 | 1 05 08 -0.0667 |
| 15.133 | 1 06 01 -0.0667 |
| 12.0 | 1 06 02 -0.0667 |
| 15.133 | 1 06 03 0.0667 |
| 15.133 | 1 06 04 0.0667 |
| 16.0 | 1 06 07 0.0667 |
| 1 01 02 1.0 | 1 06 08 0.0667 |
| 1 01 03 1.0 | 1 06 09 -0.0667 |
| 1 01 11 1.0 | 1 06 10 -0.0667 |
| 1 02 04 1.0 | 1 06 11 0.01667 |
| 1 02 05 1.0 | 2 06 12 0.01667 |
| 1 03 06 1.0 | 2 01 0.0 |
| 1 03 07 1.0 | 2 05 35.0 |
| 1 04 08 1.0 | |

| P (matrix) | |
|---|---|
| $P_1$ | 0.0 |
| $P_2$ | 0.0 |
| $P_3$ | 0.0 |
| $P_4$ | 0.0 |
| $P_5$ | 38.0 |
| $P_6$ | 0.0 |

| Output Data<br>Total End Moment | |
|---|---|
| $M_1$ = -104.53591 | $M_7$ = -133.14998 |
| $M_2$ = -73.84198 | $M_8$ = -84.90203 |
| $M_3$ = -84.90202 | $M_9$ = -73.84198 |
| $M_4$ = -133.14998 | $M_{10}$ = -104.53591 |
| $M_5$ = 133.14998 | $M_{11}$ = 158.74401 |
| $M_6$ = 133.14998 | $M_{12}$ = 158.74401 |

(Fixed End Moments)   $FM_i$ = 0 for i = 1, 2, ..., 2m

A (matrix)

| | $DM_1$ | $DM_2$ | $DM_3$ | $DM_4$ | $DM_5$ | $DM_6$ | $DM_7$ | $DM_8$ | $DM_9$ | $DM_{10}$ | $DM_{11}$ | $DM_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| $P_2$ | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $P_3$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $P_4$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| $P_5$ | 0.0 | 0.0 | -.0667 | -.0667 | .0222 | .0222 | -.0667 | -.0667 | 0.0 | 0.0 | 0.0 | 0.0 |
| $P_6$ | -.0667 | -.0667 | .0667 | .0667 | 0.0 | 0.0 | .0667 | .0667 | -.0667 | -.0667 | .01667 | .01667 |

102

## PROGRAM FOR IBM 1620-60K

```
C  C  MATRIX SOLUTION OF SLOPE DEFLECTION EQUATIONS
C     BY TONY A ROSS - UNIVERSITY OF MISSISSIPPI
      DIMENSION P(15),FM(30),A(15,30),S(30,30),UI(15,15)
      DIMENSION BI(15),BL(15),TR(30,15),BM(30),AT(30,15)
1     FORMAT(2I2)
2     FORMAT(F15.8)
3     FORMAT(I1,1X,I2,1X,I2,1X,F15.8)
5     FORMAT(4X,3HBM(,I2,3H) =,1X,F16.5)
6     FORMAT(4X,3H R(,I2,3H) =,1X,F16.5)
7     FORMAT(4X,3H X(,I2,3H) =,1X,F16.5)
9     FORMAT(I1,1X,I2,1X,F10.2)
931   FORMAT(/48HFINAL JOINT ROTATIONS AND SIDESWAY DISPLACEMENTS)
933   FORMAT(/16HTOTAL END MOMENT)
C     READ M (NUMBER OF MEMBERS) AND
C     N (NUMBER OF JOINT ROTATIONS AND SIDESWAY FORCES)
4444  READ 1,M,N
C     READ REL. MOMENT OF INERTIA
      DO 100 J=1,M
100   READ 2,BI(J)
C     READ MEMBER LENGTHS
      DO 101 J=1,M
101   READ 2,BL(J)
      M=2*M
      DO 102 J=1,N
      P(J)=0.0
      DO 102 K=1,M
      FM(K)=0.0
102   A(J,K)=0.0
C     READ IN THE (A) MATRIX
104   READ 3,I,J,K,A(J,K)
      GO TO (104,108),I
C     READ THE VALUES FOR THE FIXED END MOMENTS
108   READ 9,I,J,FM(J)
      GO TO (108,109),I
C     READ IN THE (P) MATRIX
109   READ 9,I,J,P(J)
      GO TO (109,110),I
C     CALCULATE THE STIFFNESS MATRIX (S)
110   DO 105 J=1,M
      DO 105 K=1,M
105   S(J,K)=0.0
      DO 106 J=2,M,2
      K=J/2
      S(J,J)=4.0*BI(K)/BL(K)
      S(J-1,J-1)=S(J,J)
      S(J,J-1)=2.0*BI(K)/BL(K)
106   S(J-1,J)=S(J,J-1)
```

```
C      TRANSPOSE MATRIX (A) INTO MATRIX (AT)
107    DO 200 J=1,N
       DO 200 K=1,M
200    AT(K,J)=A(J,K)
C      DETERMINE THE MULTPLIER MATRIX     UI=A*S*AT
       DO 300 I=1,M
       DO 300 J=1,N
       TR(I,J)=0.0
       DO 300 K=1,M
300    TR(I,J)=TR(I,J)+S(I,K)*AT(K,J)
       DO 301 I=1,N
       DO 301 J=1,N
       UI(I,J)=0.0
       DO 301 K=1,M
301    UI(I,J)=UI(I,J)+A(I,K)*TR(K,J)
C      BEGIN MATRIX INVERSION
       DO 400 J=1,N
       DO 400 K=1,N
       A(J,K)=UI(J,K)
400    UI(J,K)=0.0
       DO 401 J=1,N
401    UI(J,J)=1.0
       DO 501 JK=1,N
       DO 501 J=JK,N
       X=A(J,J)
       DO 405 K=1,N
       UI(J,K)=UI(J,K)/X
405    A(J,K)=A(J,K)/X
       LL=J+1
       JL=J-1
       IF(JL)403,310,403
403    DO 311 L=1,JL
       X=A(L,J)
       DO 311 K=1,N
       UI(L,K)=UI(L,K)-UI(J,K)*X
311    A(L,K)=A(L,K)-A(J,K)*X
310    IF(LL-N)412,412,501
412    DO 312 L=LL,N
       X=A(L,J)
       DO 312 K=1,N
       UI(L,K)=UI(L,K)-UI(J,K)*X
312    A(L,K)=A(L,K)-A(J,K)*X
501    CONTINUE
C      END MATRIX INVERSION
C      CALCULATE THE (X) MATRIX      X=UI*P
 1111  DO 500 J=1,N
       BL(J)=0.0
       DO 500 K=1,N
500    BL(J)=BL(J)+UI(J,K)*P(K)
```

```
C      DETERMINE THE CARRY OVER MOMENT    CM=S*AT*X
       DO 600 J=1,M
       TR(J,1)=0.0
       DO 600 K=1,N
600    TR(J,1)=TR(J,1)+AT(J,K)*BL(K)
       DO 601 J=1,M
       BM(J)=0.0
       DO 601 K=1,M
601    BM(J)=BM(J)+S(J,K)*TR(K,1)
C      CALCULATE THE FINAL MOMENT    BM=CM+FM
       DO 700 J=1,M
700    BM(J)=BM(J)+FM(J)
       PUNCH 933
       DO 916 J=1,M
916    PUNCH 5,J,BM(J)
       GO TO 4444
       END
```

## SUMMARY

The method of structural analysis presented in this paper is a direct application of the already familiar slope-deflection equations. This property of the program makes it especially useful as a teaching aid in the third and fourth years of engineering education.

Although the program could be used by anyone, small engineering firms access to a 1620 computer would probably find it more useful than would the larger firms. The larger firms would probably have access to larger computers and more sophisticated programs, but the smaller firms with limited funds can now begin to make use of this modern technique of structural analysis.

# SIMULATION OF UPTAKE AND DISTRIBUTION OF ANESTHETIC AGENTS

Gerald A. Kien, Ph.D.
Northwestern Medical School
303 E. Chicago Avenue
Chicago, Illinois
and
Floyd N. Heller, M.D.
University of Illinois R & E Hospital
840 S. Wood Street
Chicago, Illinois

Among the many diversified uses of the digital computer, there have been relatively few dealing with simulation of organ system interactions and the influence of drugs on them. Computers are being used routinely in the medical field for the conventional data processing tasks of statistical analysis, data retrieval, pattern recognition and patient monitoring. Perhaps the lag in the utilization of the engineering technique of system simulation has been due both to the apparent complexity of biological systems and to the lack of individuals specifically oriented toward the construction of simulation models.

It is noteworthy that many of the physiologic responses known may be described mathematically. Warner[1] recently stated that many models with unrealistic properties have been proposed and justified by their originators in order to obtain an analytical solution of the model. Computer simulation has now invalidated this argument since the analytical solution is no longer necessary or even sought in most cases. Perhaps the most important contribution which digital computer simulation makes to the building of mathematical models is that it no longer restricts the model builder to models for which he can devise an analytical solution. Moreover, it is possible to deviate from the rigorous mathematical model approach by the use of a transaction or process-oriented digital computer language. Simscript and GPSS may be the best examples of this.

It is a frequent comment that in dealing with the simulation of biological events, there are other drawbacks. First, there are certainly areas for which data are essentially unavailable, or at least unmeasurable at the present time. Second, when dealing with the human organism certain intangibles arise, such as atypical individual responses to specific stimuli, giving conflicting responses in both the same individual and in different individuals, as well as in different species. Third, the practical applications may be obscure to the casual observer.

In regard to the unknown data, this is not as insurmountable as it would seem at first. We know, for example, that the gross response to a specific stimulus (e.g., the administration of an anesthetic drug) reflects itself with an overall effect on the body, affecting many of its physiologic parameters. We can measure and record these changes. Some of the specific modes of action are unknown, but the effect has occurred nevertheless. All we can do is simulate the response. If our theories as to how the response occurred are eventually proved incorrect, it is of no major significance as long as the response that occurs is accurate. When we do not know a minute detail, or when this detail is of minimum consequence in the overall picture, it can be approximated or neglected completely, depending on what responses we really need for the general simulation. In the simulation of uptake and distribution of anesthetic agents, there can be no better example than the realization that the exact mechanism by which an anesthetic actually obtunds the function of a nerve cell is as yet unknown. Nevertheless, this does not hinder the anesthesiologist in knowing that it does indeed induce an anesthetic state, nor does it hamper him in the slightest in how he administers the drug. The overall response is the important aspect.

The variability of response to specific stimuli is considered by generating a probabilistic likelihood for a given occurrence from a specific set of stimuli. This attempts to develop a more complete simulation by allowing some of the variable factors that do indeed exist in nature to become part of the model. That we are unaware of a specific cause relating to these probabilistic events is unfortunate. However, it occurs within the model with the same uncertainty that it may occur unexpectedly in real life. Thus, although a specific anesthetic agent almost always produces anesthesia at a given blood level, it is possible for the simulation model to allow the patient to require much more than the normal dose level before initiation of a pharmacologic response pattern.

We feel finally that this computer simulation will have some practical significance, as a tool for both investigation and teaching. The simulation program has been designed for maximum interaction between the executing simulation model and the "anesthesiologist". Thus, he may change any of the variables in the program or the status of the simulation during its execution. He may thereby investigate the dynamics of drug and organ system inter-relationships.

A number of mathematical models of the uptake and distribution of anesthetic agents have been reported and some relevant predictions have been made from electrical analogs. Since these electrical analogs do not provide for changes in organ system characteristics seen during the course of clinical anesthesia, their usefulness and accuracy is limited solely to very low dose levels of the drug. Similarly, because of the assumptions and simplifications necessary for construction of a mathematical model, these also tend to suggest conclusions which are not necessarily complete under the variety of conditions that do indeed exist.

As an anesthetic gas enters the lungs, it undergoes a number of changes. It is diluted by the gases already present. It is heated and expands. Water vapor is added to it and increases its volume, while decreasing the partial pressure of any other gas in the system. A certain amount of anesthetic gas is lost to lung tissue and circulating blood, thus decreasing the remaining total gas volume and the volume of the anesthetic gas within the total lung volume. If alveolar ventilation is to remain constant, then the gas taken up must be replaced by gas drawn in. This increases inflow volume by the amount taken up, but maintains the outflow volume constant. Loss of the anesthetic from the blood to the peripheral tissues then occurs and is dependent on such factors as tissue solubility and blood flow per unit volume of tissue.

Immediately on administration of the anesthetic agent, and throughout the duration of the anesthetic state, there occurs a series of pharmacologic response patterns which cause alterations of the physiologic mechanisms controlling the uptake and distribution of the anesthetic agent. As the concentration of the anesthetic agent increases in the myocardium there may be a decrease in cardiac output resulting in a reduced presentation of the drug to all organ systems. Increasing concentrations of the anesthetic in the central nervous system may cause a reduction in the respiratory tidal volume and, in the absence of respiratory assistance, will reduce the amount of drug presented to the lungs.

The simulation of uptake and distribution of anesthetic agents is a transaction or process-oriented computer simulation model. The primary language used is Fortran II-D with SPS subroutines. In its general organization, the simulation model resembles GPSS and SIMSCRIPT.

The model consists of a mainline control program and a series of subroutines representing the interrelated systems and interface systems of the body. The pulmonary circulation, the heart and the lungs are considered systems. The transport of gas from the lung alveoli to the pulmonary capillary blood is considered an interface system. These subroutines are under supervision of a main timing routine and are called by the mainline control program.

The attributes of the system as a whole, or of its sub-systems, defines the status of the simulation model. These are modified by a series of intrinsic events which are caused to occur at fixed or probabilistic intervals in simulated time depending upon the values of one or more attributes of the system. Execution of a system or interface system subroutine is under the control of a main timing routine which keeps track of simulated time and calls events in their proper sequence.

The attributes of the system or its components may also be changed during the simulation by a series of extrinsic events. The extrinsic events may be generated at the start of the simulation run, in which case the event and the time it is scheduled to occur will be stored in a future events list. Alternatively, the simulation may be interrupted during its execution by the console operator. The status of the system variables can be printed at selected time intervals. Depending on the value of these variables, the anesthesiologist may request additional data and then change any of the system variables under his control (i.e., anesthetic gas concentration, ventilation volume, etc.). The simulation can then be restarted. On termination of the simulation, the history of the anesthetic course is printed including multiple page plots and tables.

[1] Warner, H.R., Simulation as a Tool for Biological Research, Simulation, Vol. 3, Number 4, October, 1964.

# A SELECTIVE DISSEMINATION OF INFORMATION SYSTEM FOR MEDICAL LITERATURE

by

James L. Grisell, Ph.D. and Roger Gudobba

The Lafayette Clinic Medical SDI system has been designed to keep scientific investigators routinely informed of the world's literature in any selected area of medicine. The system can provide either a current awareness of new articles or a bibliographic search of an entire file of articles for references relating to any topic in the area of specialization. At the Lafayette Clinic we use the system for the literature on the mental illness of schizophrenia.

The current awareness portion of the system is designed to work on an automatic basis. Each investigator serviced by the system has on file in the Computing Laboratory a list of key terms, designated as a profile, which define that segment of the schizophrenia literature of interest to him. The key terms used in profiles are obtained from a dictionary of all terms occurring in the entire file. Once a month, all profiles are compared against all new articles entered into the system during the preceding four weeks. If key terms in a profile match the terms in a new entry, then the article should be of interest to that investigator.

If an investigator wants a bibliography derived from the complete file of articles, he prepares a profile which defines his area of research interest. This profile is then matched against all articles in the system. When the key terms in the profile match the key terms in an article, this article will be included in the bibliography. The investigator receives a complete list of all articles which matched his profile. If the investigator wants to be kept informed of new articles in the area of interest defined by his bibliographic search profile, then this profile can be added to the current awareness file of profiles to be searched on a monthly basis.

This system has been designed to provide a maximum of flexibility in writing profiles so that each investigator can define his area of interest with optimum precision. Continuing feedback is also provided regarding profile key terms which are matched with article key terms. If a key term keeps finding articles which are not of interest to the investigator the profile can be modified to eliminate those key terms. On the other hand, if a given profile passes over articles of interest, then appropriate key terms for finding these articles can be added. The ultimate goal of the system is to bring to the attention of each investigator only those articles which are of interest to him.

## Description of Entries

The ultimate success or failure of any current awareness or bibliographic search system is the adequacy of the bibliographic reference data which is used. This data must meet two important criteria: (1) it must cover the world's literature of the area of interest as thoroughly as possible and (2) it must contain an adequate, but concise, description of the contents of each entry.

A source of bibliographic information which adequately meets both of these criteria is the MEDLARS system of the National Library of Medicine. The National Library is currently indexing 16,000 journal issues from all over the world. These journals presently contain 160,000 articles of medical interest. By 1969 the Library will index 25,000 journals containing 250,000 medical articles. As each journal is received at the National Library it is checked for articles of medical interest. For medical journals, all articles are routinely indexed. For a journal such as Science, only those articles pertaining to medical topics are indexed.

Each article is read by an indexer. The indexer then assigns a series of tags, or index words, which define the subject content of the article. These tags are words which appear in Medical Subject Headings (MESH). Currently, an average of 10 tags are assigned to each article. The tags or index words are also the subject headings under which articles are listed in the Index Medicus. Since cross-referencing an article on the average of ten times would produce a voluminous Index Medicus, the indexing is done on two levels. The first level consists of tags most representative of the article's contents. These are used for inclusion in Index Medicus and are preceded by an x in a MEDLARS listing. The other tags are not used for Index Medicus but are available for searches of the MEDLARS' master file. Thus, a search of their file will yield more articles than you would get by looking in Index Medicus under the same headings used for the search.

The entries in the MEDLARS system are available from the National Library on the basis of either a demand search, or on the basis of a recurring search. To receive a demand search they require a list of tags from MESH and the dates within which the search is to be performed. A recurring search is a routine search made of all new articles within a specified time period. Thus, if one wishes this on a monthly basis, once a month the list of tags defining an area of interest are routinely run against the new entries for the month and the list provides all articles containing one or more of the tags.

## The Format of an Article in MEDLARS

Each article in the MEDLARS system contains certain basic information descriptive of the articles. The following items are included:

1) The author's name. If the article has multiple authorship all are included in the listing, with the senior author listed first. The last name is given first, followed by the author's initials. No first names are given. If there is no author (1.5°/∘) it is listed as anonymous.

2) The title of the article.
If the paper is in English the title will be listed exactly as it appears in the article. If the paper is in a foreign language, the title will be in English translation and in the Roman alphabet. The title itself will be enclosed in parentheses. The language in which the article was written is also indicated by a standard abbreviation, also enclosed in parentheses.

3) The source in which the article was published.
The journals are given in standard abbreviated form. The volume number, month and year of publication and pagination are also given.

4) <u>Index terms or tags.</u>
The index terms which describe the subject contents of the article are also included. An index term must be in MESH. Index Terms which are used in Index Medicus are preceded by an asterisk.

<u>Preparation of MEDLARS entries for use in the SDI system.</u>

All articles received to date, and they now number 2300, have been prepared for entry into our SDI system. One of the guiding principles of our system is that it be as automatic as possible. Consequently, we do a minimum of pre-editing of article listings as they are received. Some modifications of the original listings are necessary, but these are limited to the assigning of identification codes to the various parts of an entry so the computer can more readily identify these parts. The order in which the entry parts are listed is consistent, but the number of lines each may require is not. Usually, the title will be on only one line, but may require two. The number of lines of index terms may range from 1 to 4. Consequently, each line of an entry must be labeled to indicate whether it is the author (A), title (T), source (S) or index terms, (designated key terms (K)). Each line is also given a sequence number and each article is given a document number.

<u>Error checking procedures for all entries.</u>

Every attempt is made to see that the entries are keypunched as accurately as possible and that the identifying information is correct. Also, the contents of the entries are checked for proper spacing. This is necessitated by the fact that the system programs which process the entries assume that each entry is in the standard format.

After the entries have been keypunched, they are verified. When all errors found in the verification process have been corrected, the entries are then run through an error checking program. For each entry the program checks the following:

1) all cards in the same entry must have the same document number

2) the alphabetic card designation must be in the sequence A, T, S, K.

3) the card sequence numbers must start with 1 and be consecutive

4) in the contents of the entry itself, each card is scanned to see if anything has been punched after the program has detected two consecutive blanks or has come to the end of the card. This is important because the presence of two consecutive blanks or the end of card is the only way subsequent programs in the system know they have reached the end of valid information on a card.

If any of the above error indications are found, an appropriate error message is provided specifying the type of error and the document number with which it occurred. No entries are used in the system until they go through the error checking program with no error messages. Figure 1 is a sample of article entries after they have been prepared for entry into the system.

FIGURE 1

Article Entries as They are Used in the SDI SYSTEM

```
1  A  1  AALL L
1  T  2  (SYMPOSIUM ON SCHIZOPHRENIA-LIKE PSYCHOSES AND ETIOLOGY OF
1  T  3  SCHIZOPHRENIA.  EXPERIENCES REGARDING THE TOPIC IN TANGANYIKA)  (GER)
1  S  4  SCHWEIZ ARCH NEUROL PSYCHIAT 93,377-9, 1964
1  K  5  *SCHIZOPHRENIA, TANGANYIKA (1)

2  A  1  AARONSON BS
2  T  2  AGING, PERSONALITY CHANGE, AND PSYCHIATRIC DIAGNOSIS.
2  S  3  J GERONT 19,144-8, APR 64
2  K  4  ADOLESCENCE, ADOLESCENT PSYCHOLOGY, *AGING, DIAGNOSIS, *MENTAL
2  K  5  DISORDERS, *MMPI, *PERSONALITY, SCHIZOPHRENIC PSYCHOLOGY,
2  K  6  SOCIOPATHIC PERSONALITY

3  A  1  ABELY P, LAUZIER B
3  T  2  (THE FATE OF THE CONCEPTS OF PERIODICITY, ATYPISM AND INCURABILITY
3  T  3  IN PRACTICAL PSYCHIATRY)  (FR)
3  S  4  ANN MEDICOPSYCHOL (PARIS) 122,729-46, MAY 64
3  K  5  CLASSIFICATION, *MENTAL DISORDERS, *NOMENCLATURE, PERIODICITY,
3  K  6  PROGNOSIS, *PSYCHIATRY, PSYCHOTHERAPY, SCHIZOPHRENIC PSYCHOLOGY

4  A  1  ABRAHAM G
4  T  2  (THE PROBLEM OF MIXED PSYCHOSES)  (FR)
4  S  3  ANN MEDICOPSYCHOL (PARIS) 122,481-90, NOV 64
4  K  4  CLASSIFICATION, *DEPRESSION, *EPILEPSY, *NEUROSES, *PSYCHOSES,
4  K  5  *PSYCHOSES, MANIC-DEPRESSIVE, *SCHIZOPHRENIA

5  A  1  ABRAMS S
5  T  2  A VALIDATION OF PIOTROWSKI'S ALPHA FORMULA WITH SCHIZOPHRENICS
5  T  3  VARYING IN DURATION OF ILLNESS.
5  S  4  AMER J PSYCHIAT 121,45-7, JUL 64
5  K  5  DIAGNOSIS, DIFFERENTIAL, *RORSCHACH TEST, *SCHIZOPHRENIA

6  A  1  ABRAMSON HA
6  T  2  ANTISEROTONIN ACTION OF LSD-25 AND OTHER LYSERGIC ACID DERIVATIVES,
6  T  3  FACT AND FICTION.
6  S  4  J ASTHMA RES 1,207-11, MAR 64
6  K  5  *ALLERGY, ASTHMA, AUTISM, CHILD, *HALLUCINOGENS, *LYSERGIC
6  K  6  ACID DIETHYLAMIDE, METHYSERGIDE (3), MIGRAINE, PHARMACOLOGY,
6  K  7  SCHIZOPHRENIA, CHILDHOOD, *SEROTONIN INHIBITORS, TOXICOLOGIC
6  K  8  REPORT (4)

7  A  1  ACHILLES M
7  T  2  (ATTEMPT AT A STATISTICAL DIAGNOSIS OF THE DRIVE STRUCTURE IN
7  T  3  PROBLEM STUDENTS) (GER)
7  S  4  PRAX KINDERPSYCHOL 13,177-81, JUL 64
7  K  5  ADOLESCENCE, AGGRESSION, AUTISM, CHILD, *CHILD BEHAVIOR
7  K  6  DISORDERS, EDUCATION OF MENTALLY DEFECTIVE, MOTIVATION,
7  K  7  PERSONALITY, PUBERTY, SEX, STATISTICS, THEMATIC APPERCEPTION
7  K  8  TEST
```

## Preparation of the Dictionaries.

One of the most important aspects of our SDI system is the preparation of the dictionary of terms contained in the MEDLARS references. This dictionary provides information which is the basis for making interest profiles. You can't know what to ask for until you know what's there. This system uses three separate kinds of dictionaries: (1) a dictionary of terms derived from both the title and index tags, (2) a dictionary of authors and (3) a dictionary of sources.

## The Dictionary of Key Terms.

This dictionary is generated by a computer program from both the title of the article and from the index tags. The program rules for finding terms in titles are different than the rules for finding terms from tags. For titles, each is scanned by the program going from left to right. The beginning and end of a word are identified by the presence of a blank. If the first character of any word is a special character, such as a parenthesis or a quotation mark, it is ignored. If the last character of a word is a parenthesis, comma, period or quotation mark, it is also ignored.

When a word has been found, it is compared against a trivial word list. This list consists of such words as: the, it, and, but, etc. There is provision in the program for 450 such terms; currently we use 100. If a word is not in the trivial word list it is considered a valid term for the dictionary and is written out on disk. If a word is in the trivial word file it is ignored.

Since the goal of the SDI system was to make it as automatic as possible, only single terms are extracted from article titles. To get multiple word terms would require the pre-editing of the titles and designating which consecutive word groups should be treated as a single term by manually inserting some special character before the first word and at the end of the last. This is done in some SDI systems. For example, one may wish to designate "social factors" as a two-word term. However, this will appear in the dictionary as two terms; "social" and "factors". While this may be regarded as a shortcoming of the system, provision can be made in the construction of profiles to treat these two words as a unit.

After all non-trivial words have been found in the title, the program then scans the K cards for the index terms assigned to the article by the National Library. When dealing with these terms, multiple word terms are treated as one. The purpose of this was to retain all of the terms as they appear in MESH. The logic of the program is very simple. It regards anything between commas as a term. Anything within parentheses is ignored. As each term is found it is also written on disk.

When the program is run with a number of articles it searches alternately title cards and key term cards. When all entries have been processed, all key terms have been written on magnetic disk in the order found. The next problem is to alphabetize them and count the frequency with which each term appeared. It is also necessary to designate new terms which have never appeared before. The alphabetizing is accomplished by sorting all key terms into alphabetical sequence. The IBM 1620 sort-merge program is used for this purpose.

The dictionary generator program has been written so that it can either generate an original dictionary, or update an old one when new entries are added to the system. A separate tabulating program has been written to work in conjunction with Phase 4 of the IBM sort program. After the terms have been sorted in alphabetical sequence, each term is made available to the tabulating program. If an original dictionary is being generated the program merely tabulates the frequency with which each term occurs. When a new term is found, the previous term, together with its frequency of occurrence, is punched on a card. When all terms have been tabulated the cards produced by the program are then listed to get a readable copy of the dictionary.

When the Dictionary of Key Terms is being updated with the key terms from a new set of entries, the procedure is slightly different. After all the terms have been identified and sorted the tabulation program requires the cards from the old dictionary. The first of these is read in and then the first word on disk is compared with it. If they are the same, the word from disk is counted and the next word from disk is brought in and compared. This procedure is repeated until the word from disk is not the same as the word on the card. When this happens the frequency of the previous word on disk is added to the cumulative total for that word on the dictionary card. Then the new cumulative total, the tabulation frequency for the current run and the term itself are punched on a card.

When the program encounters a word on disk which was not previously in the dictionary it is a new term. When the term is punched on a card an asterisk is put in front of it. When a term on cards is found but which is not in the current entries, the monthly total is set to zero and the cumulative total is unchanged. Figure 2 is a sample page of the Dictionary of Key Terms.

## FIGURE 2

Sample Page of the Dictionary of Key Terms

DICTIONARY OF KEY TERMS
LAFAYETTE CLINIC MEDICAL SDI SYSTEM
UPDATED AS OF 8/24/65

| CUMULATIVE TOTALS | TOTAL FOR MONTH | | |
|---|---|---|---|
| 1 | 0 | ..... | ABBREVIATION |
| 1 | 0 | ..... | ABERRATIONS |
| 1 | 0 | ..... | ABILITIES |
| 5 | 0 | ..... | ABILITY |
| 5 | 0 | ..... | ABNORMAL |
| 8 | 0 | ..... | ABNORMALITIES |
| 1 | 0 | ..... | ABNORMALITY |
| 1 | 0 | ..... | ABNORMALTIES |
| 1 | 0 | ..... | ABO FACTORS |
| 5 | 0 | ..... | ABORTION |
| 1 | 1 | ..... | * ABREACTION |
| 2 | 0 | ..... | ABRUPT |
| 1 | 0 | ..... | ABSCESS |
| 1 | 0 | ..... | ABSOLUTE |
| 1 | 0 | ..... | ABSORPTION |
| 2 | 0 | ..... | ABSTRACT |
| 1 | 0 | ..... | ABSTRACTION |
| 1 | 0 | ..... | ABSURDITY |
| 1 | 0 | ..... | ACANTHROCYTOSIS |
| 1 | 1 | ..... | * ACCEPTABILITY |
| 1 | 0 | ..... | ACCEPTANCE |
| 1 | 0 | ..... | ACCEPTORS |
| 1 | 0 | ..... | ACCOMMODATION |
| 1 | 0 | ..... | ACCOMPANYING |
| 1 | 0 | ..... | ACEPROMAZINE |
| 4 | 0 | ..... | ACETATES |
| 1 | 0 | ..... | ACETIC |
| 5 | 0 | ..... | ACETOPHENAZINE |
| 7 | 0 | ..... | ACETYLCHOLINE |
| 4 | 0 | ..... | ACHIEVEMENT |
| 1 | 0 | ..... | ACHIEVEMENTS |
| 1 | 0 | ..... | ACHILLES |
| 3 | 0 | ..... | ACHILLES TENDON |
| 31 | 3 | ..... | ACID |
| 1 | 1 | ..... | * ACIDOSIS |
| 5 | 0 | ..... | ACIDS |

After the dictionary has been updated, the next step is to prepare a list of new entries. A program has been written for this purpose. The cards comprising the dictionary are read into the computer. Each card is examined for the presence of an asterisk denoting a new term. If an asterisk is found the word on the card is punched out. When the program has read through all of the dictionary cards and the output listed, we have a readable copy of all new terms which have been added to the dictionary. This listing is made available to all of the investigators being serviced by the SDI system. The list of new terms is submitted to them prior to running their profiles against the new entries so the investigator can include any of the new terms in his profile to get the articles containing them. At some reasonable interval each user will be given a new dictionary.

## Dictionary of Authors and Dictionary of Sources

In addition to the Dictionary of Key Terms, two other dictionaries are prepared: one of all authors and the other of all of the sources in which the literature has appeared. The entries in both of these dictionaries may be used in constructing profiles. Thus, an investigator interested in getting all of the publications of a particular author may do so. Also, he can ask for all of the articles appearing in a given journal.

Both of these dictionaries are prepared in the same way as the dictionary of key terms. The difference is that for the dictionary of authors, only the author card is processed. The authors are listed separately even though an article may have multiple authorship. For the dictionary of sources only the source card is scanned. In the listing of sources only the journal name is included. All information pertaining to volume, year and pagination is ignored. Listings of new entries are prepared for both of these dictionaries in the same way as with the Dictionary of Key Terms.

## Profiles

The ultimate success or failure of an SDI system is a function of the ease and accuracy with which an investigator can define his area of interest on the basis of the terms in the articles being searched. This is done by constructing a profile of terms. The terms in the profile are then compared with the terms in an article. If the computer finds a match in these sets of words, then the article is designated as being of interest to the investigator. If the profile is a good one it will maximize the number of articles of interest it finds and minimize the number of articles designated as interesting, but which are not. If the profile is not a good one, the reverse will then be true.

The profiles used in this system are similar to the profiles used in the IBM SDI system, but with modifications. In the Lafayette Clinic Medical

SDI system a profile has a hit level, which may range from -9 to +99. Each term in the profile has a weight. This also has a range from -9 to +99. Each term in the profile is compared with each term in the article. Each time a match is found the weight of the term in the profile is summed. After the last comparison is made, the sum of the terms on which a match occurred is compared to the hit level of the profile. If the sum is equal to or greater than the hit level, then the article is designated as being of interest. If the sum is less than the hit level the article is ignored.

In addition to a total hit level for the profile and a weight for each term, there are also two kinds of terms which may be used: complete terms and root terms. A complete term is one which appears in the profile exactly as it does in the dictionary of key terms. The term must appear in the article exactly as it is in the profile to result in an equal compare. A root term, on the other hand, will result in comparing only as many letters in the article word as are in the root term in the profile. For example if "child" is designated as a root term in the profile then it will result in an equal compare with children, children's, childhood and, of course, child. The use of root terms is a convenient way of encompassing all variants of a term which may have one of several different endings.

As one final feature, each term in a profile may also have one of two modifiers: _must_ and _not_. A must modifier simply indicates that any time a must term is found the investigator will get that article whether the hit level has been reached or not. A not term will do the opposite. When a not term is found the article will be skipped even though the hit level has been equaled or exceeded. The only exception occurs when a must term and a not term are both found in the same article. Under these circumstances the must term overrides the not term. Either a root or a complete term can have either of the modifiers. Figure 3 is a sample profile.

FIGURE 3

Sample Profile

```
                 LAFAYETTE CLINIC MEDICAL SDI SYSTEM
                        BIBLIOGRAPHIC SEARCH
                       SEARCH MADE ON  9/29/65


                   KEY TERM PROFILE DESCRIPTION

IDNO =  10001 NAME - SPECIAL                       LOCATION  LC HIT LEVEL = 2

        MODI-  WORD
        FIER   TYPE   WT.        KEY TERM

        MUST   COMP    1      ABRAMS S
         NOT   COMP    1      CHILD BEHAVIOR DISORDERS
               COMP    1      CLASSIFICATION
               COMP    2      DIAGNOSIS
               COMP    1      NOMENCLATURE
         NOT   ROOT    1      RORSCHACH
```

## Searches

This SDI system has been designed to perform two kinds of searches: an SDI search and a bibliographic search. The SDI search is performed on a monthly basis and essentially it compares each profile with each of the monthly articles. The articles are read from cards.

The bibliographic search essentially matches one profile against all articles in the system. For the bibliographic search, the master file of articles is stored on disk.

In an SDI search, all profiles are first read in and then written out on disk (on drive 0), in a packed format. After the profiles have been stored the first article is read in from cards. The program then brings in each profile from disk and compares the terms in the profile with the terms in the article. If the sum of the weights of the terms on which a comparison is made equals or exceeds the hit level of the profile, or if a "must" term is found, the article is said to match the profile. When this occurs certain identifying information is written on the second disk drive (drive 1). This consists of

1) the profile number
2) document number of the article
3) the article sequence number (which indicates if it is the first, second, etc. article on which a match was found)
4) the words on which a comparison was made between profile terms and article terms

The <u>first</u> time a match is found on an article, the article itself is written on disk (drive 0). The following information is needed:

1) the document number

2) the authors of the article

3) the title of the article

4) the source in which the article appears

The index terms are not recorded. An article is written on disk only once, the first time a match is found between the article and some profile. If no matches occur the article is not written out. After all profiles have been compared against an article, the next article is read in and all profiles are compared against it, etc. until all the articles have been processed. After all of the articles have been processed, the contents of the disk containing the data for the hits (drive 1) are sorted in numerical sequence by profile number and by document number within the profile number.

The program which punches out the results works in conjunction with Phase 4 of the sort program. The output consists of the profile, followed by the articles on which a match was found. The articles appear in numerical sequence by document number. Figure 4 shows a sample output. This output was the result of running the profile in Figure 3 against the articles in Figure 1.

The bibliographic search is designed to compare one profile with the entire file of literature references. At the beginning of the program the profile is read in from cards but instead of being written out on disk it is retained on core storage. In addition, it is punched out as the initial part of the

## FIGURE 4

Articles from Figure 1 which Match Profile in Figure 3

```
         THESE ARTICLES MATCH THE PROFILE DESCRIBED ABOVE


2  A  AARONSON BS
   T  AGING, PERSONALITY CHANGE, AND PSYCHIATRIC DIAGNOSIS.
   S  J GERONT 19,144-8, APR 64

        MODI-  WORD
        FIER   TYPE   WT.       KEY TERM

               COMP    2    DIAGNOSIS


3  A  ABELY P, LAUZIER B
   T  (THE FATE OF THE CONCEPTS OF PERIODICITY, ATYPISM AND INCURABILITY
   T  IN PRACTICAL PSYCHIATRY) (FR)
   S  ANN MEDICOPSYCHOL (PARIS) 122,729-46, MAY 64

        MODI-  WORD
        FIER   TYPE   WT.       KEY TERM

               COMP    1    CLASSIFICATION
               COMP    1    NOMENCLATURE


5  A  ABRAMS S
   T  A VALIDATION OF PIOTROWSKI'S ALPHA FORMULA WITH SCHIZOPHRENICS
   T  VARYING IN DURATION OF ILLNESS.
   S  AMER J PSYCHIAT 121,45-7, JUL 64

        MODI-  WORD
        FIER   TYPE   WT.       KEY TERM

        MUST   COMP    1    ABRAMS S
               COMP    2    DIAGNOSIS
         NOT   ROOT    1    RORSCHACH TEST
```

output. The articles against which the profile are compared are stored on disk. The articles are read one at a time and compared against the profile. When a match is found, the reference is punched out immediately. The output of this section of the program has the same output format as the SDI search.

## Description of SDI Monitor Pack

The operation of this system is greatly facilitated by setting up a special Monitor disk pack. Since several of the operations performed by the system involve a series of linked programs, they can be executed sequentially without changing packs. By putting on this pack only those parts of Monitor which are necessary for the operation of the system under Monitor control, and only those programs which pertain to the system, large areas of disk are made available for storing data. Work storage is defined as being the entire second drive (drive 1).

The sections of Monitor I loaded are:

1. The supervisor routines
2. The DUP routines
3. The DIM table
4. The Equivalence table
5. The sequential program table
6. The disk pack label areas

The system programs are stored in sectors adjacent to the DIM table and in the upper part of the disk. Sectors 00000-04799 and sectors 05200-16999 are available for data storage. The first area is used for storing profiles in an SDI search. The second area is used for storing articles on which a profile match was found during an SDI search.

The DIM, equivalence and sequential program tables were read in without modification from the Monitor deck. After loading, the DIM entries for the SPS assembler and Fortran compiler were deleted using the Monitor DUP routine. This made the storage space assigned to these packages available for storing the SDI programs.

Before the SDI Monitor deck can be set up, all of the SDI programs must be assembled with a standard Monitor pack. At the end of each assembly an object deck is punched out. Before these can be loaded the SPS and Fortran DIM entries must be deleted. The object decks can then be loaded.

The SDI system described in this paper is currently in operation at the Lafayette Clinic. The system, as described, was written to run on an IBM 1620, Model 1 with 60,000 digits of internal core storage with two 1311 disk drives. In addition, the special instructions transmit numeric strip, transmit numeric fill, and indirect addressing are required. With very minimal modifications the program package will run on a 40K machine. This can be accomplished by merely reducing the size of the trivial word file used in the program for generating the dictionary of key terms.

As yet, this system has not been submitted to the Users Group but it is available directly from the Lafayette Clinic to anyone who is interested in using this system with some aspect of the medical literature which is made available by the National Library of Medicine.

**********************************************************************

## HISTORY OF THE DEVELOPMENT OF PART

AT TENNESSEE TECH, STUDENTS ARE TAUGHT MACHINE LANGUAGE PROGRAMMING ON THE IBM 1620 COMPUTER, BEING REQUIRED TO WRITE COMPUTER PROGRAMS FOR THE SOLVING OF CERTAIN SPECIFIED PROBLEMS. FOR THE PAST TWO YEARS, WE HAVE BEEN USING A MODIFIED PROGRAM CALLED ... MARCAT ..., WRITTEN BY PROFESSOR GUY RICKER OF NEW JERSEY STATE COLLEGE WHICH EXECUTED AND CHECKED THE RESULTS OF MACHINE LANGUAGE PROGRAMS FOR THE 1620. AFTER THE EXPERIENCE GAINED USING MARCAT, AND DUE TO CERTAIN FEATURES OF MARCAT THAT WERE UNDESIRABLE FOR OUR APPLICATIONS, IT WAS DECIDED TO WRITE OUR OWN PROGRAM TO SATISFY OUR NEED FOR A MEANS OF PROCESSING, EXECUTING, AND CHECKING LARGE VOLUMES OF STUDENT WRITTEN MACHINE 'ANGUAGE PROGRAMS. OUR USE OF THE ... PART ... PROGRAM PERMITS WIDE USE OF THE MARK SENSING CAPABILITIES OF OUR IBM 514 REPRODUCING PUNCH TO EXPEDITE THE PUNCHING AND PROCESSING OF THE PROGRAMS. HOWEVER, THERE IS NO REASON WHY, AS LONG AS PROPER CARD FORMATS ARE FOLLOWED, THE PART PROGRAM COULD NOT BE USED WITH ALL INFORMATION BEING HAND PUNCHED AND THE USE OF MARK SENSE EQUIPMENT ELIMINATED.

**********************************************************************

## THE PURPOSE OF PART

THE PURPOSE OF PART IS TO GIVE THE COMPUTER CENTER A CONVENIENT MEANS OF HANDLING STUDENT WRITTEN MACHINE LANGUAGE PROGRAMS AND, AT THE SAME TIME, GIVE THE STUDENTS A DEFINITE ANSWER AS TO THE STATUS OF THEIR PROGRAMS. IT YIELDS TO THE INSTRUCTOR A CONVENIENT MEANS OF ANALYZING THE PROGRESS OF EACH OF THE STUDENTS. FURTHERMORE THE STUDENTS ARE GIVEN PERTINENT INFORMATION RELATING TO THE SOURCE OF THEIR ERRORS. THE MAJOR PURPOSE OF PART IS TO CHECK ASSIGNED PROBLEMS FOR PREVIOUSLY DETERMINED ANSWERS.

**********************************************************************

## BRIEF DESCRIPTION OF PART

PART OPERATES ON THE BASIS OF EXECUTING STUDENT WRITTEN MACHINE LANGUAGE PROGRAMS AND COMPARING THE DERIVED ANSWERS WITH PREVIOUSLY STORED CORRECT ANSWERS. ALL COMPARISONS ARE MADE AS FIELDS.

FIRST, THE ANSWERS ARE STORED FOR ALL OF THE PROBLEMS INVOLVED. SECOND, THE STUDENT PROGRAM IS LOADED INTO CORE BY THE PART PROCESSOR. THEN THE STUDENT PROGRAM IS EXECUTED UNDER CONTROL WHILE CONSTANTLY BEING CHECKED FOR ERRORS. IF AN ERROR IS ENCOUNTERED, THE STUDENT PROGRAM IS DISCONTINUED AND THE APPROPRIATE ERROR MESSAGE IS GIVEN OUT. PROVIDED NO OBVIOUS ERRORS EXIST, THE ANSWERS GIVEN BY THE PROGRAM ARE CHECKED BY REFERENCE TO THE PRE-ASSIGNED LOCATIONS. INFORMATION CORRESPONDING TO THE STATUS OF THE ANSWERS IS GIVEN OUT AND PART CONTINUES BY ACCEPTING ANOTHER STUDENT PROGRAM.

**********************************************************************

## THE PHASES OF PART

ANSWER LOADER - THIS SECTION IS USED TO STORE THE ANSWERS AND THE CORRESPONDING TABLES FOR TABLE LOOK-UP. THIS SECTION OF PART IS CLEARED WHEN THE FIRST STUDENT PROGRAM IS LOADED.

THEREFORE, THE ANSWER LOADER CANNOT BE RE-ENTERED.

LOADER FOR STUDENT PROGRAMS - THIS PHASE LOADS THE STUDENT PROGRAM, TRAPS THE FOLLOWING CARD (IF PRESENT) AND THEN GIVES CONTROL TO THE MONITORING ROUTINE.

MONITORING ROUTINE - THIS PHASE BEGINS WITH THE FIRST INSTRUCTION OF THE PROGRAM. THE INSTRUCTION IS FULLY CHECKED AND THEN EXECUTED UNDER CONTROL, IF PROVEN TO BE PERMISSIBLE. THE NEXT ADDRESS IS DETERMINED AND THE INSTRUCTION THERE IS USED TO REPEAT THE PROCESS. THIS IS CONTINUED UNTIL AN ERROR IS ENCOUNTERED OR THE PROGRAM IS TERMINATED. INPUT DATA IS ALSO HANDLED IN THIS PHASE BY TRAPPING ONE CARD AHEAD OF THE STUDENT PROGRAM. NOTE THAT THE PROGRAM IS ALSO TERMINATED WHEN THE DATA CARDS ARE EXHAUSTED.

ANSWER CHECK ROUTINE - THIS PHASE USES THE PROBLEM NUMBER TO FIND THE STORED ANSWER INFORMATION AND TO DETERMINE THE NUMBER OF PASSES FOR THE PROBLEM. THIS MAY BE ENTERED SEVERAL TIMES FROM THE MONITORING ROUTINE. ENTRY POINTS ARE AT THE OCCURANCE OF AN INPUT-OUTPUT INSTRUCTION AND WHEN THE STUDENT TERMINATES HIS PROGRAM. THE PASSES OCCURING IN THE PROBLEM ARE CHECKED UNTIL PROVEN TO BE CORRECT. TO BE CORRECT, ALL PARTS OF THE PASS MUST BE CORRECT. ONCE A PASS IS FOUND TO BE CORRECT, IT IS NOT CHECKED AGAIN. FOLLOWING THE CHECK AFTER TERMINATION OF THE STUDENT PROGRAM, THIS ROUTINE GIVES OUT THE RES''LT OF THE ANSWER CHECKS. CONTROL IS THEN GIVEN BACK TO THE LOADER FOR STUDENT PROGRAMS.

*******************************************************************************

## ANSWER STORAGE

ANSWERS ARE STORED AS FIELDS WITH A MAXIMUM LENGTH OF 15 DIGITS. ANSWERS MAY BE BROKEN UP INTO AS MANY AS THREE PASSES. EACH PASS MAY BE BROKEN UP INTO AS MANY PARTS AS DESIRED, ALL OF WHICH MUST BE VERIFIED BY THE STUDENT WRITTEN PROGRAM BEFORE RECEIVING AN OK FOR THAT PASS. THE ANSWERS ARE ENTERED ON CARDS WITH FROM ONE TO THREE ANSWERS PER CARD. EACH ANSWER MUST CONTRAIN WITH IT TWO DIGITS SPECIFYING THE PROBLEM NUMBER, AND ONE DIGIT SPECIFYING THE PASS. A DIGIT IS PROVIDED TO NUMBER THE PART OF THE PASS. THE HIGHEST PASS MUST APPEAR BEFORE THE OTHERS. ALL PARTS OF ONE PASS MUST BE TOGETHER. THE ANSWER PARTS MAY BE CONTINUED ON AS MANY CARDS AS NECESSARY. IT IS NOT NECESSARY TO USE A NEW CARD TO BEGIN A DIFFERENT PASS OR A DIFFERENT PROBLEM. FORMAT MUST BE FOLLOWED (SEE LATER IN THIS PAPER FOR PROPER FORMAT). THE ANSWERS MUST BE FOLLOWED BY A CARD CONTAINING SOMETHING OTHER THAN A RECORD MARK IN COLUMN 80.

*******************************************************************************

## SPECIAL PROVISIONS AND/OR LIMITATIONS OF THE PART PROCESSOR

ANY IBM 1620 MACHINE LANGUAGE PROGRAM THAT FALLS WITHIN THE BOUNDS DESCRIBED BELOW, THAT DOES NOT CALL FOR SPECIAL CHARACTERS AS INPUT AND DOES NOT CALL FOR PROGRAM HALTS CAN BE PROCESSED BY THE PART PROCESSOR WHEN THE PROPER FORMAT IS FOLLOWED.

ALL STUDENT WRITTEN PROGRAMS MUST BE WRITTEN SO AS TO OCCUPY LOCATIONS 00401 THROUGH 07999. UNLESS OTHERWISE SPECIFIED BY AN ADDRESS DEFINITION CARD, THE STUDENT PROGRAM WILL BE LOADED INTO LOCATION 00500 AND SUCCESSIVELY HIGHER LOCATIONS. ADDRESS DEFINITION

CARDS MAY BE USED AT ANY POINT OF THE STUDENT DECK TO CAUSE A DIFFERENT LOADING SEQUENCE TO BE ESTABLISHED. STUDENT WRITTEN PROGRAM CARDS MUST HAVE TWO INSTRUCTIONS PER CARD AND ADDRESS DEFINITION MAY NOT BE MADE BETWEEN ANY TWO INSTRUCTIONS ON THE SAME CARD

PROGRAMS NOT STORED IN THE ANSWER TABLES WILL BE COMPLETELY MONITORED BUT THE ANSWERS WILL NOT BE CHECKED. THE MESSAGE ...PROB NOT IN TABLE ... IS GIVEN FOR THESE PARTICULAR PROGRAMS. THIS ALLOWS THE STUDENT TO EXPERIMENT WITH PARTICULAR PROGRAMS OF INTEREST TO HIM OTHER THAN THOSE ASSIGNED BY HIS INSTRUCTOR.

THE STUDENT IS FREE TO CHOOSE HIS FORM OF INPUT DATA. THIS IS DUE TO THE FACT THAT PART CHECKS ANSWERS IRREGARDLESS OF THE ORDER OF THEIR OCCURANCE.

PART ALLOWS FOR BOTH BATCH PROCESSING AND SINGLE PROCESSING. IN SINGLE PROCESSING, THERE IS NO NEED TO CLEAR OUT THE PUNCH SIDE OF THE 1622 AS A BLANK CARD IS PUNCHED AFTER EACH PROGRAM IS COMPLETED.

ERROR MESSAGES ARE EASILY UNDERSTOOD BY THE STUDENTS. THE ERRORS THAT PART DETECTS ARE LARGELY OF THE TYPE THAT WOULD GIVE MACHINE CHECK STOPS. STUDENTS ARE FURNISHED WITH THE LOCATION OF THE ERROR, THE INSTRUCTION CAUSING THE ERROR, AND THE CORRESPONDING ERROR MESSAGE.

THERE IS AN ADVANTAGE IN THE POSSIBILITY OF CONSOLE DEBUGGING AS THE STUDENT PROGRAM IS NOT ALTERED IN ANY WAY BY THE PROCESSOR.

THE STUDENT IS FREE TO USE THE LAST CARD INDICATOR, EVEN ON BATCH COMPILING, AS IT IS SET UP AS A CHECK DIGIT.

ALL INPUT TO PART IS ALPHAMERIC. PROGRAM CARDS ARE STRIPPED INTO THE PROGRAM AREA AND THE FLAGS ARE REPLACED. DATA CARDS ARE ALSO READ IN BY AN ALPHAMERIC READ. IF THE STUDENT CALLS FOR A NUMERIC READ, PART STRIPS HIS DATA CARD INTO THE PLACE CALLED FOR. THE FLAGS AND RECORD MARKS ARE RETAINED. SPECIAL CHARACTERS ARE LOST IN THE STRIPPING OPERATION. HOWEVER, THE STUDENT MAY INPUT THE SPECIAL CHARACTERS BY AN ALPHAMERIC READ. DUE TO THE NECESSITY FOR A NUMERIC BLANK, ONE HAS BEEN PLACED AT LOCATION 08001 FOR STUDENT USE. FOR CONVENIENCE, A RECORD MARK HAS "EEN PLACED AT LOCATION 00400.

THE PROBLEM OF EARLY TERMINATION OF INFINITE LOOPS IS NOT COMPLETELY TAKEN CARE OF. A COUNT IS MAINTAINED OF EACH USAGE OF OUTPUT FOR EACH PROGRAM. THAT IS, EACH TIME A 38 OR 39 INSTRUCTION IS USED IN EACH PROGRAM. A MAXIMUM HAS BEEN SET AT 50. THERE IS NO PROVISION FOR EARLY TERMINATION OF LOOPS THAT DO NOT CONTAIN AN OUTPUT INSTRUCTION.

AS THE HALT (48) IS CONSIDERED A TERMINATION, THE STUDENT IS NOT FREE TO STOP HIS PROGRAM DURING AN EXECUTION TO EXERCISE ANY SWITCH OPTIONS, ETC. ALL SWITCHES MUST BE SET BEFORE EXECUTION.

ONLY A MAXIMUM OF THREE PASSES FOR ANSWER CHECKS HAVE BEEN ALLOWED.

PART DISCONTINUES PROCESSING A PROGRAM ON ENCOUNTERING ONLY ONE ERROR.

AS ANSWERS ARE CHECKED ON ENCOUNTERING INPUT-OUTPUT INSTRUCTIONS, THERE IS THE RARE POSSIBILITY THAT A STUDENT MAY BE CHECKED OK AND OUTPUT THE ANSWERS INCORRECTLY.

**********************************************************************

## NOTES ON TERMINATION OF STUDENT PROGRAMS

OCCURANCE OF ANY OF THE ERRORS CAUSES IMMEDIATE TERMINATION WITH THE ERROR INFORMATION TYPED AND PUNCHED.

OCCURANCE OF THE HALT (48) INSTRUCTION CAUSES TERMINATION OF THE PROGRAM AND INITIATES THE FINAL ANSWER CHECK.

THE PROGRAM BEING PROCESSED IS TERMINATED AND THE ANSWER CHECK IS INITIATED WHEN A READ IS CALLED FOR AFTER THE LAST DATA CARD HAS BEEN USED BY THE STUDENT PROGRAM. THIS ALLOWS THE STUDENT TO USE AN OPEN-END METHOD OF PROGRAMMING WITHOUT LEGAL TERMINATION IF CARE IS TAKEN TO FURNISH ALL THE DATA CALLED FOR.

**********************************************************************

## ERROR MESSAGES

ARITH OVERFLOW - FROM EITHER Q OPERAND LONGER THAN THE P OPERAND OR CARRY BEYOND P OPERAND HIGH ORDER DIGIT

WRITE CHECK - WRONG INFORMATION ASSEMBLED IN A WRITE ALPHAMERIC ON CARDS OR ON THE TYPEWRITER

NO FLAG IN Q - NO FLAG ON Q OPERAND OR Q OPERAND LONGER THAN 99 DIGITS

INSUFFICIENT DATA - PROGRAM CALLS FOR DATA WHEN NONE HAS BEEN SUPPLIED.

RECORD MARK IN FIELD - PROCESSOR HAS SENSED THE PRESENCE OF A RECORD MARK IN AN ARITHMETIC FIELD

35 INSTRUCTION NOT ALLOWED - YOU HAVE ATTEMPTED TO USE A DUMP NUMERIC (35) IN YOUR PROGRAM. THIS INSTRUCTION WOULD BE NON -TERMINATING.

PARITY ERROR - ODD - YOU HAVE PROBABLY ATTEMPTED AN ALPHAMERIC INPUT OR OUTPUT TO OR FROM AN EVEN NUMBERED LOCATION OR ATTEMPTED TO EXECUTE AN INSTRUCTION ORIGINATING IN AN ODD NUMBERED LOCATION.

ADDRESS OUT OF BOUNDS - ATTEMPTED USAGE OF LOCATION OTHER THAN 00401 THROUGH 08001 OR 80 THROUGH 90. THIS CAN OCCUR BY EITHER END OF A FIELD OR RECORD BEING OF THE RANGES SPECIFIED BEFORE OR AFTER EXECUTION.

FIELD OVER 99 DIGITS - EXPLANATION IS OBVIOUS

RECORD OVER 99 DIGITS - EXPLANATION IS OBVIOUS

THIS INDICATOR NOT ALLOWED - INVALID INDICATOR CODE NUMBER OR INDICATOR CODE NUMBER OTHER THAN 1 THROUGH 4, 9, OR 11 THROUGH 13.

ILLEGAL INPUT-OUTPUT - Q8 Q9 DOES NOT MATCH SPECIFIED INPUT OR OUTPUT DEVICE

ILLEGAL EXPONENT IN P - THESE TWO MESSAGES APPLY TO FLOATING POINT<br>
ILLEGAL EXPONENT IN Q - CALCULATIONS. EXPONENT LONGER THAN 2 DIGITS.

P ADDRESS MUST BE EVEN - P OPERAND ADDRESS OTHER THAN EVEN NUMBERED ADDRESS FOR A BRANCH.

RECORD OF ONLY RECORD MARK - ALTHOUGH NOT A TRUE LOGIC ERROR, YOU ARE PROBABLY ATTEMPTING A RECORD TRANSMISSION FROM THE WRONG END OF RECORD

42 USED BEFORE 17 OR 27 - YOU HAVE ATTEMPTED A BRANCH BACK OPERATION BEFORE USING A BRANCH AND TRANSMIT OR A BRANCH AND TRANSMIT IMMEDIATE. (NO ADDRESS STORED IN IR 2.)

ILLEGAL CONTROL STATEMENT - SOMETHING OTHER THAN 1 IN Q9, SOMETHING OTHER THAN 1, 2, OR 8 IN Q11 OF A CONTROL (34) STATEMENT.

REF ADDR HAS ILLEGAL OP CODE - EXECUTION ATTEMPTED OF AN INSTRUCTION WITH AN INVALID OP CODE. YOU HAVE PROBABLY BRANCHED TO A WRONG ADDRESS.

ILLEGAL OP CODE - EXPLANATION IS OBVIOUS. YOU HAVE PROBABLY MISMARKED YOUR CARD OR HAVE CALLED FOR AN INSTRUCTION IN YOUR PROGRAM TO BE MODIFIED OR CLOBBERED

PROG DOES NOT TERMINATE - THE STUDENT HAS ATTEMPTED TO CALL FOR OUTPUT MORE THAN THE MAXIMUM OF 50 TIMES, OR PROGRAM DOES NOT LEGALLY TERMINATE.

INVALID CARD CODE - AN INPUT CARD CONTAINS AN INVALID PUNCH.

OUTPUT TOO LONG - TYPEWRITER HAS BEEN CALLED UPON TO OUTPUT MORE THAN 80 CHARACTERS.

DESTROYED REC MARK IN 400 - THE INSTRUCTION JUST EXECUTED HAS OVERRIDDEN THE RECORD MARK STORED IN LOCATION 00400.

PROB NOT IN TABLE - THE PROBLEM SPECIFIED BY THE IDENTIFICATION CARD DOES NOT APPEAR IN THE ANSWER TABLES.

P ADDRESS MUST BE ODD - P ADDRESS FOR THIS INSTRUCTION MUST BE ODD.

***********************************************************************

THE FORM OF OUTPUT FOR ERRORS IS AS GIVEN BY THE FOLLOWING EXAMPLE ...

```
52245   00800   491200000000    ADDR OUT OF BOUNDS

IDENTIFICATION...LOCATION...INSTRUCTION...MESSAGE
```

THIS EXAMPLE WOULD INDICATE THAT CLASS NUMBER 5, STUDENT NUMBER 22, AND PROBLEM NUMBER 45 HAD AN ERROR IN THE INSTRUCTION BEGINNING IN LOCATION 00800. THE INSTRUCTION CALLS FOR A BRANCH TO 12000 WHICH IS ILLEGAL IN THAT IT CALLS FOR AN ADDRESS OUT OF BOUNDS (BRANCHES INTO THE PART PROGRAM ITSELF)

***********************************************************************

## INSTRUCTIONS FOR THE MARKING (OR PUNCHING) OF CARDS FOR USE WITH THE PART PROGRAM

THE STUDENT WILL BE GIVEN FOUR TYPES OF CARDS (EITHER MARK SENSE OR STOCK CARDS). THESE CARDS ARE ... IDENTIFICATION CARDS, PROGRAM CARDS, DATA CARDS, AND ADDRESS DEFINITION CARDS.

THE CARDS ARE TO BE MARKED (OR PUNCHED) AS FOLLOWS

IDENTIFICATION CARDS ...

COLUMN 1 ... CLASS NUMBER (TO BE ASSIGNED BY THE INSTRUCTOR)
COLUMNS 2 AND 3 ... STUDENT NUMBER
COLUMNS 7 THROUGH 27 ... AVAILABLE FOR STUDENT USE FOR NAME IN ALPHA. IF USED, EXTREME CARE MUST BE TAKEN TO USE VALID CARD CODES, OTHERWISE, THE CARDS WILL NOT READ INTO THE COMPUTER.

PROGRAM CARDS ... 2 INSTRUCTIONS PER CARD

COLUMNS 1 THROUGH 12 ... FIRST INSTRUCTION
COLUMNS 13 THROUGH 24 ... SECOND INSTRUCTION

DATA CARDS ...

BEGINNING IN COLUMN 1, THE DESIRED DATA (UP TO A MAXIMUM OF 27 DIGITS, IN THE CASE OF MARK SENSE CARDS) IS TO BE MARKED. FLAGS MAY BE INDICATED BY MARKING THE 11 ZONE PUNCH AS WELL AS THE NUMBER DESIRED.

ADDRESS DEFINITION CARD ....

TO ENABLE A STUDENT TO BEGIN HIS PROGRAM AT A LOCATION OTHER THAN 00500, HE MAY BE ALLOWED TO USE AS MANY OF THESE CARDS AS NECESSARY.

COLUMNS 1 THROUGH 5 ... BEGINNING LOCATION OF THE FIRST INSTRUCTION ON THE PROGRAM CARD WHICH IMMEDIATELY FOLLOWS.

THE AFOREMENTIONED CARDS WILL BE ARRANGED AS SHOWN BELOW.

NOTE ... ADDRESS DEFINITION CARDS MUST IMMEDIATELY PRECEDE THE FIRST PROGRAM CARD FOR WHICH THE ADDRESS IS INTENDED

**********************************************************************

# A STUDENT SCHEDULING SYSTEM

Michael Kennedy
Western Carolina College
(User No. 1396)

If one reads only the newspapers he is aware that the use of computers in assigning students to classes in colleges and high schools is not a simple or trivial application. The problem is complex to a degree most easily measured by the number of schools which have tried it and failed, with results varying from annoying to catastrophic. The difficulty of the problem turns on two main points: 1) The system requires the co-operation, or at least compliance, of large numbers of students, faculty, and others, most of whom, at best, don't care if the application is a success or not, and 2) computer people often don't understand the problems related to registration of students, and university officials seldom understand computers. What I will describe is one system which works for one particular school, Western Carolina College, in hopes that some of the methods used there will be useful to others.

Western Carolina College is a liberal arts college of about 2800 students which is buried deep in the Appalachian Mountains of North Carolina. The college is quite isolated--located 50 miles south of Asheville, N. C. The college obtained its first data processing equipment in Fall of 1963. This equipment consisted of a 20K IBM 1620 card system and a series 50 tab installation. The Computing Center was organized primarily for instruction and research. In January of 1964 the Computing Center was asked to automate the college's registration and scheduling procedure which was getting out of hand due to rapidly increasing enrollment. The first application was conducted during November of the same year at the winter quarter registration and has been used, with modification, each subsequent quarter.

Prior to the undertaking of such a system it was necessary to determine its general philosophy. First, we felt that the faculty's part should be limited to advisement functions only. Further, we felt that as large a part of the faculty as possible should be absent on registration day. We felt that every student should get an even chance for the courses he desired. He would not, however, be able to choose times, instructors, or rooms. In fact, at the time the student made known his desires for the coming quarter, the master schedule for that quarter would not even be prepared. We would schedule students in an order dictated by the number of quarters which remained until their graduation because we knew that space problems would prevent some students from getting the courses they desired. Our objective was to obtain for each student a conflict-free schedule consistent with his needs as determined in conference with his advisor. We further wished to balance the numbers of students in all multisectioned courses. Finally, we wished to complete the registration of any given quarter (Fall quarter excepted) prior to the end of final examinations for the preceeding quarter. Thus "registration" or registration day for any quarter would consist only of processing new and transfer students and modifying the schedules of those students who needed to make changes.

The system which was developed will be discussed under four general headings: 1) Pre-registration; 2) Master Schedule Development; 3) Scheduling; and 4) Drop-Add Procedures.

## PRE-REGISTRATION

At approximately the middle of each quarter each department chairman submits a list of courses, without reference to times, instructors, etc., which his department intends to offer the succeeding quarter. He does not even specify how many sections of each course he intends to offer. The Computing Center assigns to each course a three digit call number starting with 002 for the first one and continuing sequentially. At about the same time the Computing Center prints a pressure—sensitive label for each student giving his name and number and the name and number of his advisor.

Using the 407 tabulator, the data center then prepares offset masters which, when printed, produce the course request forms. The pressure-sensitive labels are affixed to the printed forms and, at the beginning of the eighth week of the quarter, they are made available to students at a central location. (See Figure 1.) The eighth week of the quarter is designated as pre-registration week and it is during this time that the student is to arrange a conference with his advisor to determine the courses which he will take the following quarter. While the length of the advisory conference will vary from student to student the marking of the course request form consumes at most a minute. If the student must be absent during certain periods of the day he may request "absence" courses which appear at the back of the form with their call numbers just as do other courses. If the student requests a free period for lunch the call number 001 is assumed. At the end of the advisory conference the advisor keeps the form which is turned in, with others collected during the day, to the departmental office. The forms are delivered daily to the Computing Center for keypunching. The student is finished with registration until he picks up his schedule shortly before the final exam period. The faculty member is finished with registration until next quarter unless he is the departmental representative designated to assist with the registration of new and transfer students on "registration day".

The rather orthodox technique of keypunching student requests was used for several reasons after careful consideration of other methods. It was felt that the printed form shown in the illustration would be simplest for use by the advisor and student. The marks required were unacomplicated and could be made by any sort of writing implement, the courses available were clearly spelled out, and no cross reference of any kind was required. In addition, we felt that the speed and verification methods available with keypunching would result in very accurate data, a "must" in a system such as this, in a fairly short time. It turned out that one keypunch operator could punch each day's request cards before the next day's requests arrived so that we had the data on cards within 24 to 48 hours of the time it was collected.

It should be pointed out that at no point did students have access to IBM cards nor did they possess the Course Request Form after it had been marked. Considering the ingenuity of students, we counted this an advantage.

The appropriate contents of the course request form for each student were punched into a single card. The student and advisor numbers were punched into the card and the student name was added later by a gangpunch operation. (It should be noted that the student was not requested even to write his name or number; these were supplied by the label on the form--a feature which made the procedure just described safe enough to use). The three digit call numbers were punched consecutively and as many as fourteen requests were possible. If a course were requested as "repeat", "graduate", or "audit", the first, second, or third digit of the call number was flagged, respectively.

## MASTER SCHEDULE DEVELOPMENT

We were then ready to develop the master schedule. In an ideal system, the computer would be presented with the student course request cards along with a statement of school operating hours, a list of rooms and their uses, and the abilities of the faculty members together with any special considerations (Professor Jones has to pick up his child at kindergarten on Tuesdays if it's raining), and the computer would then produce an optimum master schedule. A great deal of work has been done on-such systems on large computers by the producers of GASP, by Stanford and Purdue Universities, and others. Such systems were, unfortunately, beyond the scope of our project at WCC.

We could and did, however, use the course request data to provide each department chairman with a planning packet to facilitate his formulation of his part of master schedule.

(See Figure 2.) The first and most basic tool of our system is the simple tally. The first column indicates the call number of the course; the second indicates the number of students who requested that course. Thus, the department chairman can plan the number of sections needed for each course.

(See Figure 3.) A slightly more sophisticated tool which is given to each department chairman for courses within his department is the conflict matrix or cross-tally. The call numbers lie above the slanting line while the tally lies below; the tally is used much as a triangular mileage chart would be. In the example it can be seen that eight students requested both course 002 and 006. This aid is invaluable in determining the time placment of single section courses. A department chairman can also request a cross tally between any two course-offering areas in order to avoid interdepartmental single section conflicts.

After the development of the master schedule by the department chairman and the registrar, the computing center punches it onto cards. The schedule designates sections, times, and number of seats allowed, rooms, and instructors.

(See Figure 4.) By comparison of the master schedule and the simple tally a surplus seat report is produced. What is of primary importance is that careful study be given to those courses with negative surplus seats. In the example it can be seen that four students will not get schedules, if for no other reason, because they have requested Leather Design which will close before their cards are processed. We felt that it was extremely important to have these data available in a tabular form.

## SCHEDULING

After the master schedule has been modifed as a result of the surplus seat report, the Computing Center is ready to begin trial scheduling runs. Usually between three and six of these are made with no output; the results of these runs usually dictate modifications of the master schedule.

At this point I feel I should emphasize the importance of a fast scheduling program. The program used by this system schedules about one student a second or 3600 hour when run without output. Its speed makes it possible to make a trial run on the entire student body in about forty minutes with the punch inhibited. The final scheduling run with the output of one card for each class for each student requires about an hour and forty minutes. If the scheduling program is fast there is less chance that a machine breakdown during the scheduling period will be catastrophic and it is possible to make several runs with changes in the various parameters of the system.

The first of the preliminary runs is usually made with the seat allowances for each section increased by 500. After this run a look at the numbers of seats remaining in the various sections gives important information about the time conflicts inherent in the master schedule. If the master schedule is perfect, with respect to time, then no student will be rejected and the sections of all multisectioned courses will have the same number of seats remaining within one seat. Deviations from this perfect result indicate flaws in the master schedule and the most critical or blatant deviations are investigated. Ususally a run is made with absence and lunch courses removed to see what part these requests play in the overall reject rate. While no output is obtained for the scheduled students, output is punched for rejected students. This output takes the form of the input course request card which may then be treated just as a new request card. These reject cards are spot checked to see if a pattern is observable and a simple tally is made of the courses requested. If a cross tally is indicated it is also made. The absence and lunch request are usually removed from the reject cards and they are resubmitted to the computer.

A final scheduling run is usually made about three days before the beginning of final exams. The result is usually about 80% scheduled, 10% rejected because of time conflicts, and 10% rejected because of lack of classroom space.

(See Figure 5.) The output of the scheduling program for students who were scheduled consists of class cards from which are printed the student schedules and the class rolls. The scheduling program actually supplies only the student information and the call and section number plus repeat, graduate, and audit information. The rest is added by a gangpunching operation. As can be seen, the card is also used in a mark sense grade reporting operation.

(See Figure 6.) Our operation was on a tenuous if not non-existant financial basis which explains, in part, the reason for the use of stock paper for the printing of the schedules. The scheduled student picks up his schedule usually on the last day of classes before finals. He can elect to pay his fees for the succeeding quarter during exam week thus completing his registration for the next quarter.

(See Figure 7.) About 20% of the students receive reject notices. These students are supplied with a copy of the master schedule, a list of closed sections, a fresh course request form, and access to the seats remaining information. With these items the student drafts a new course request form and resubmits it, together with proof that the requested courses can be scheduled. About 90% of these students will be scheduled and pick up their schedules prior to leaving campus at the end of the final exam period. The rejects from this second run are given the opportunity to resubmit their course request forms but will have to wait until the day of registration of new students to pick up their schedules. There will normally be at most sixty students in this latter group.

At the end of the quarter, the Computing Center turns its attention to grade reporting via the mark sense grade card which was the output of the scheduling project of the preceeding quarter. It also prints class rolls which will be ready for instructors on the first day of classes of the quarter about to start.

At the beginning of the new quarter several different types of scheduling data are arriving in the Computing Center. Various routing forms are used to direct data flow. A routing form is used to tag the new course requests of non-scheduled students.

New students must be scheduled and the same procedure is followed for them as was used with students during pre-registration.

On some occasions, when a student needs to make extensive changes in his schedule, it is necessary for him to submit a new course request form.

Errors are bound to occur with individual students--let's hope there are none in the master schedule. An error correction routing form is little used but vital. In many cases a student will say an error has occurred when what has actually happened is that he has changed his mind or has failed a course which is prerequisite to one he has requested. The error correction card makes it possible to track these down quickly. If an error has indeed occurred, it is to the advantage of all concerned that it be corrected quickly and efficiently.

These routing forms may seem to be simply an administrative nuisance, but they are necessary if duplicate sets of class cards are to be avoided for students. If a student has two sets of class cards he will be taking up space which could be occupied by some other student. Further, it is possible that a schedule will appear for such a student showing a load of forty hours or so and some courses which meet at identical times. This is most embarassing to the system.

## DROP-ADD

We will now consider the automated drop-add operation.

The fact that a pre-registration has been used will increase the magnitude of the drop-add problem. Since students will fail courses, change majors, etc., the drop-add machinery needs to be ready. At least one school using a computer scheduling system requires the student to risk his entire schedule when he attempts to make a schedule change; this policy c reates general dissatisfaction with the system. The procedure used at WCC allows the student to request changes in his schedule, with his advisor's consent, while retaining his original schedule. From his request is punched a drop-add header card. His existing class cards are collated behind this header and submitted to the computer which checks the master schedule to see if seats are available. If so the computer punches out a set of class cards which represents the student's former schedule plus and minus the requested changes. If one or more courses do not have seats available the students former schedule is duplicated with an explantory note. The input is then discarded. At any time during drop-add period, or before a department chairman can increase the number of seats in any section.

At the end of the drop-add period final class rolls are printed and the process is complete. Complete, that is, until three weeks hence when the whole operation begins again.

It might be said that we feel there are four important ingredients in a successful computer registration: 1) A clear understanding of what is to be accomplished; 2) a pre-registration which allows evaluation of students' requests as opposed to a system which uses a computer just to take the place of so-called "call pulling" in the gymnasium; 3) a fast and well checked out scheduling program which will be used to make several runs on the course request data; and 4) an automated drop-add procedure which dovetails with the scheduling system and offsets the drop-add problems created by the pre-registration.

WESTERN CAROLINA COLLEGE

FIGURE I

COURSE REQUEST FORM

PRE-REGISTRATION FOR
FIRST SESSION SUMMER 1965

808080 J D STUDENT

999 A PEDAGOGUE

1. MARK X IN PARENTHESES TO INDICATE COURSES REQUESTED.
STUDENTS SHOULD INDICATE REQUESTS BY MARKING LIGHTLY WITH A REGULAR PENCIL PRIOR TO THE ADVISORY CONFERENCE.
ADVISORS SHOULD MARK APPROVED COURSE REQUESTS DARKLY WITH A RED PENCIL.

2. CIRCLE R (FOR REPEAT) IF REQUESTED COURSE HAS BEEN TAKEN BEFORE.

3. CIRCLE G IF REQUESTED COURSE IS TO BE TAKEN FOR GRADUATE CREDIT.

4. CIRCLE A IF REQUESTED COURSE IS TO BE AUDITED.

5. IS A FREE PERIOD FOR LUNCH DESIRED? YES _____ NO _____

6. NUMBER OF QUARTERS UNTIL GRADUATION (CIRCLE ONE) 1 2 3 4 5 6 7 8 9-OR-MORE

7. TO REQUEST A COURSE MARKED WITH AN ASTERISK THE STUDENT MUST OBTAIN WRITTEN APPROVAL FROM THE DEPARTMENT OFFERING THAT COURSE. STUDENTS WHO FAIL TO OBSERVE THIS REQUIREMENT WILL BE DROPPED FROM CLASS.

8. ADVISORS ARE ASKED TO ALLOW STUDENTS TO REQUEST ABSENCE COURSES ONLY IN CASES OF NECESSITY.

STUDENT IS NOT TO WRITE IN THIS SPACE.

TOTAL NUMBER OF QUARTER HOURS REQUESTED _______

ADVISOR'S SIGNATURE ____________________

| REQUEST | HRS. CR. | COURSE | TITLE |
|---|---|---|---|
| | | | ACCOUNTING |
| 002( ) R G A | 4 | 2401 | ACCOUNTING I |
| 003( ) R G A | 2 | 2411 | ACCOUNTING IIA |
| 004( ) R G A | 4 | 2421 | ACCOUNTING III |
| 005( ) R G A | 4 | 4411 | AUDITING |
| | | | ART |
| 006( ) R G A | 3 | 130 | HIST ART SURVEY |
| 007( ) R G A | 3 | 131 | INTRO DR PT DESIGN |
| 008( ) R G A | 2 | 220A | DESIGN LEATHER |
| 009( ) R G A | 2 | 220B | DESIGN PLASTICS |
| 010( ) R G A | 2 | 224 | CERAMICS |
| 011( ) R G A | 3 | 235 | ADVANCED CERAMICS |
| 012( ) R G A | 3 | 331 | INTER DRAW PAINT |
| 013( ) R G A | 3 | 333 | INTER COMPOSITION |
| 014( ) R G A | 3 | 433D | ART IN EL SCHOOL |
| 015( ) R G A | 3 | 433E | ARTS CRAFTS TEACH |
| 016( ) R G A | 3 | 437 | ADV DR PT COMP DES |
| 017( ) R G A | 3 | 438 | ADV DR PT COMP DES |
| 018( ) R G A | 3 | 439 | ADV DR PT COMP DES |
| | | | BIOLOGY |
| 019( ) R G A | 5 | 150 | PRINS CELL BIOL |
| 020( ) R G A | 5 | 151 | GEN ZOOL INVERT |

FIGURE 2

```
CALL  SEATS       COURSE
NO.   REQUESTED   DESCRIPTION

                 ACCOUNTING
002     0033        2401  ACCOUNTING I
003     0024        2411  ACCOUNTING IIA
004     0015        2421  ACCOUNTING III
005     0014        4411  AUDITING
                 ART
006     0037        130   HIST ART SURVEY
007     0020        131   INTRO DR PT DESIGN
008     0014        220A  DESIGN LEATHER
009     0013        220B  DESIGN PLASTICS
010     0009        224   CERAMICS
011     0009        235   ADVANCED CERAMICS
012     0009        331   INTER DRAW PAINT
013     0005        333   INTER COMPOSITION
014     0005        433D  ART IN EL SCHOOL
015     0005        433E  ARTS CRAFTS TEACH
016     0005        437   ADV DR PT COMP DES
017     0000        438   ADV DR PT COMP DES
018     0000        439   ADV DR PT COMP DES
                 BIOLOGY
019     0025        150   PRINS CELL BIOL
020     0024        151   GEN ZOOL INVERT
```

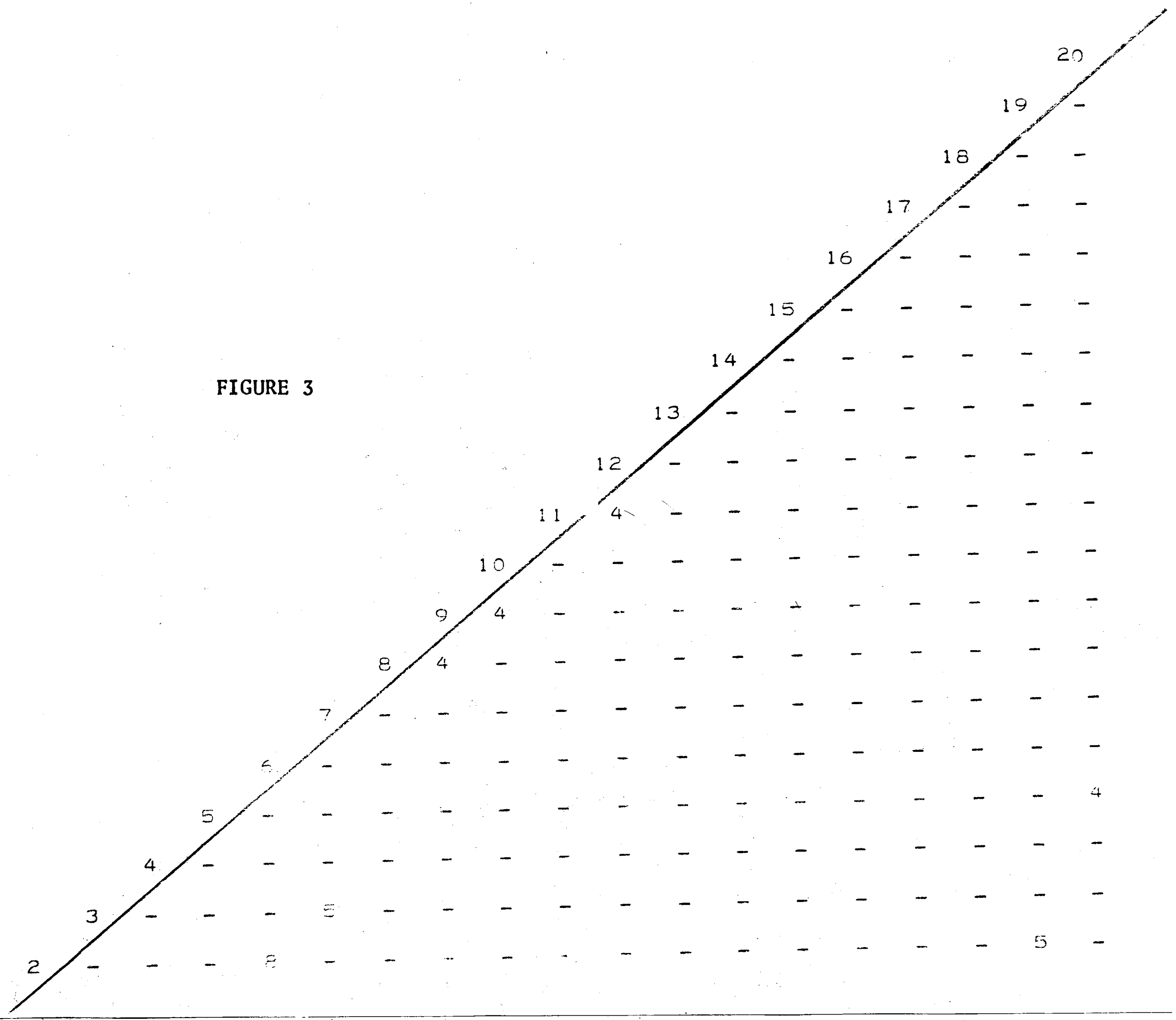
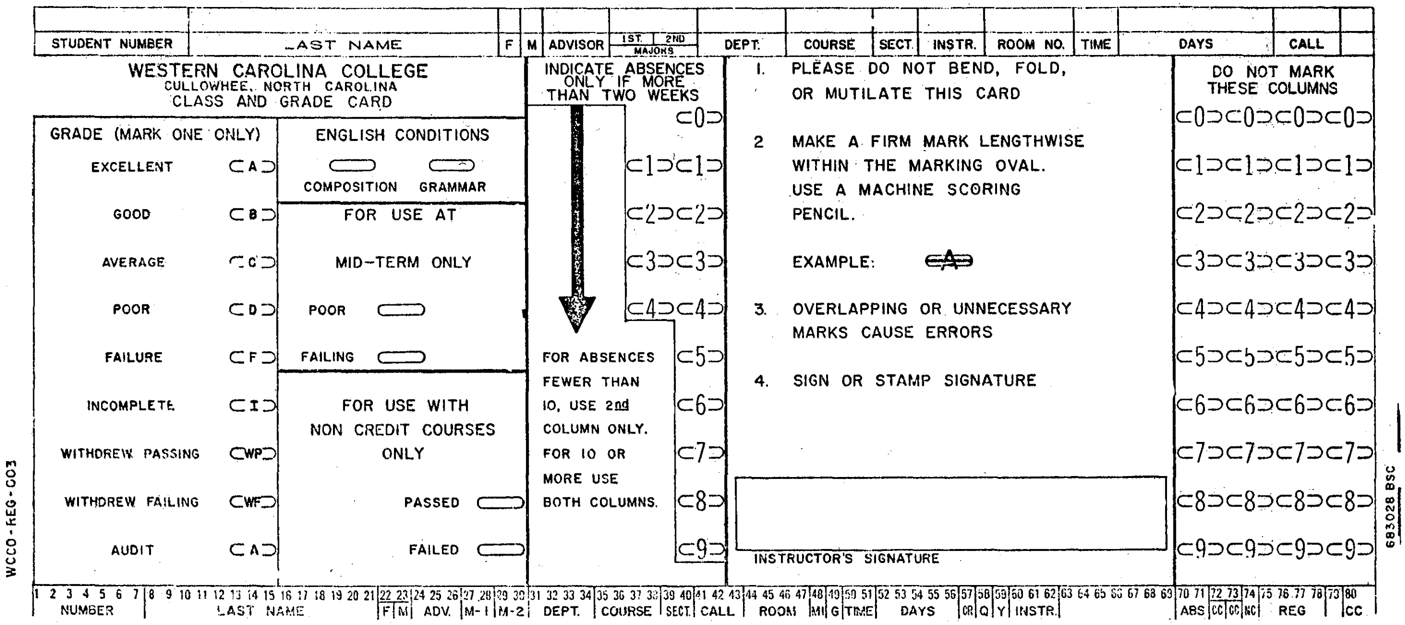


FIGURE 3

FIGURE 4

```
CALL  SEATS       COURSE                                  SEATS   SEATS   SEATS
NO.   ASKED       DESCRIPTION                            ALLOWED  ASKED   SURPLUS

                ACCOUNTING
002    0033       2401 ACCOUNTING I                        035     033     +002
003    0024       2411 ACCOUNTING IIA                      035     024     +011
004    0015       2421 ACCOUNTING III                      035     015     +020
005    0014       4411 AUDITING                            035     014     +021
                ART
006    0037       130  HIST ART SURVEY                     044     037     +007
007    0020       131  INTRO DR PT DESIGN                  024     020     +004
008    0014       220A DESIGN LEATHER                      010     014     -004
009    0013       220B DESIGN PLASTICS                     010     013     -003
010    0009       224  CERAMICS                            010     009     +001
011    0009       235  ADVANCED CERAMICS                   010     009     +001
012    0009       331  INTER DRAW PAINT                    006     009     -003
013    0005       333  INTER COMPOSITION                   004     005     -001
014    0005       433D ART IN EL SCHOOL                    010     005     +005
015    0005       433E ARTS CRAFTS TEACH                   010     005     +005
016    0005       437  ADV DR PT COMP DES                  005     005     +000
017    0000       438  ADV DR PT COMP DES                  005     000     +005
018    0000       439  ADV DR PT COMP DES                  005     000     +005
                BIOLOGY
019    0025       150  PRINS CELL BIOL                     028     025     +003
020    0024       151  GEN ZOOL INVERT                     035     024     +011
```

FIGURE 5



| STUDENT NUMBER | LAST NAME | F | M | ADVISOR | 1ST. MAJORS | 2ND MAJORS | DEPT. | COURSE | SECT. | INSTR. | ROOM NO. | TIME | DAYS | CALL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

WESTERN CAROLINA COLLEGE
CULLOWHEE, NORTH CAROLINA
CLASS AND GRADE CARD

GRADE (MARK ONE ONLY)

| | |
|---|---|
| EXCELLENT | ⊂A⊃ |
| GOOD | ⊂B⊃ |
| AVERAGE | ⊂C⊃ |
| POOR | ⊂D⊃ |
| FAILURE | ⊂F⊃ |
| INCOMPLETE | ⊂I⊃ |
| WITHDREW PASSING | ⊂WP⊃ |
| WITHDREW FAILING | ⊂WF⊃ |
| AUDIT | ⊂A⊃ |

ENGLISH CONDITIONS

⊂⊃ COMPOSITION ⊂⊃ GRAMMAR

FOR USE AT MID-TERM ONLY

POOR ⊂⊃

FAILING ⊂⊃

FOR USE WITH NON CREDIT COURSES ONLY

PASSED ⊂⊃

FAILED ⊂⊃

INDICATE ABSENCES ONLY IF MORE THAN TWO WEEKS

⊂0⊃
⊂1⊃⊂1⊃
⊂2⊃⊂2⊃
⊂3⊃⊂3⊃
⊂4⊃⊂4⊃
⊂5⊃
⊂6⊃
⊂7⊃
⊂8⊃
⊂9⊃

FOR ABSENCES FEWER THAN 10, USE 2nd COLUMN ONLY. FOR 10 OR MORE USE BOTH COLUMNS.

1. PLEASE DO NOT BEND, FOLD, OR MUTILATE THIS CARD

2 MAKE A FIRM MARK LENGTHWISE WITHIN THE MARKING OVAL. USE A MACHINE SCORING PENCIL.

EXAMPLE: ⊂A⊃

3. OVERLAPPING OR UNNECESSARY MARKS CAUSE ERRORS

4. SIGN OR STAMP SIGNATURE

INSTRUCTOR'S SIGNATURE

DO NOT MARK THESE COLUMNS

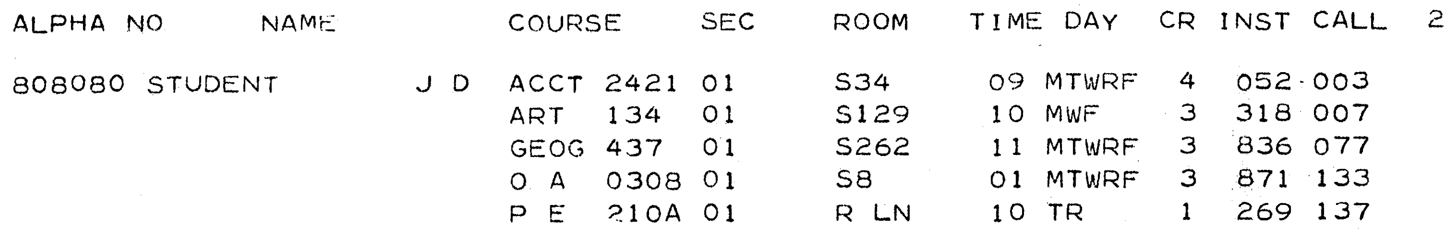
⊂0⊃⊂0⊃⊂0⊃⊂0⊃
⊂1⊃⊂1⊃⊂1⊃⊂1⊃
⊂2⊃⊂2⊃⊂2⊃⊂2⊃
⊂3⊃⊂3⊃⊂3⊃⊂3⊃
⊂4⊃⊂4⊃⊂4⊃⊂4⊃
⊂5⊃⊂5⊃⊂5⊃⊂5⊃
⊂6⊃⊂6⊃⊂6⊃⊂6⊃
⊂7⊃⊂7⊃⊂7⊃⊂7⊃
⊂8⊃⊂8⊃⊂8⊃⊂8⊃
⊂9⊃⊂9⊃⊂9⊃⊂9⊃

WCCO-REG-003

683028 BSC

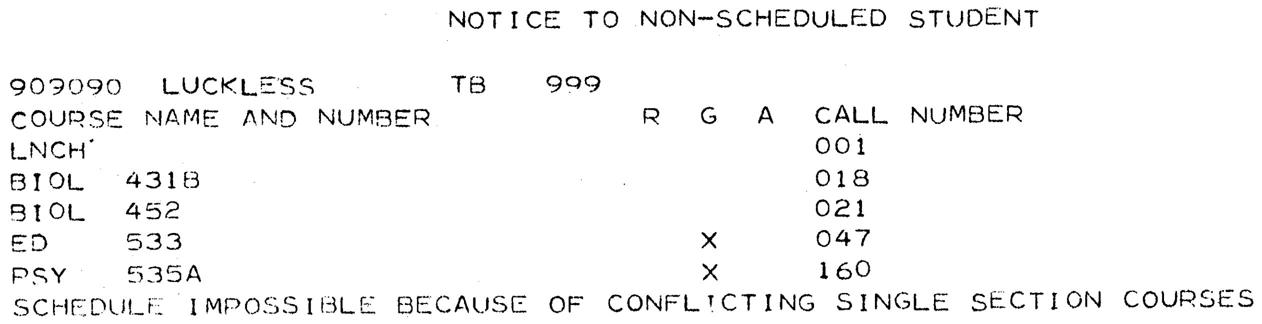
| 1 2 3 4 5 6 7 | 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | 22 | 23 | 24 25 26 | 27 28 | 29 30 | 31 32 33 34 | 35 36 37 38 | 39 40 | 41 42 43 | 44 45 46 47 | 48 | 49 | 50 51 | 52 53 54 55 56 | 57 | 58 | 59 | 60 61 62 | 63 64 65 66 67 68 69 | 70 71 | 72 | 73 | 74 | 75 76 77 78 | 79 | 80 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NUMBER | LAST NAME | F | M | ADV. | M-1 | M-2 | DEPT. | COURSE | SECT. | CALL | ROOM | MI | G | TIME | DAYS | CR | Q | Y | INSTR. | | ABS | CC | CC | NC | REG | | CC |

FIGURE 6



```
ALPHA NO     NAME              COURSE     SEC    ROOM    TIME DAY    CR  INST CALL   2

808080 STUDENT           J D  ACCT 2421  01     S34       09 MTWRF   4   052 003
                              ART  134   01     S129      10 MWF     3   318 007
                              GEOG 437   01     S262      11 MTWRF   3   836 077
                              O A  0308  01     S8        01 MTWRF   3   871 133
                              P E  210A  01     R LN      10 TR      1   269 137
```

FIGURE 7



```
                         NOTICE TO NON-SCHEDULED STUDENT

909090   LUCKLESS          TB    999
COURSE NAME AND NUMBER                 R   G   A    CALL NUMBER
LNCH'                                               001
BIOL   431B                                         018
BIOL   452                                          021
ED     533                                 X        047
PSY    535A                                X        160
SCHEDULE IMPOSSIBLE BECAUSE OF CONFLICTING SINGLE SECTION COURSES
```

Presented to Eastern-
Midwestern Users Group
Conference, New York
City, October 7, 1965

# TEST GENERATION PROGRAM

by
Val Tareski
and
Donald Peterson
North Dakota State University
Fargo, North Dakota

## ABSTRACT

A significant percentage of the computer time used at North Dakota State University is for undergraduate educational use. To aid the instructors in preparing tests for a beginning computer course and to obtain greater consistency between tests given to the various sections, a Fortran Program was written to randomly select questions from a master set of questions. Although the examples shown are for a computer programming course, it is obvious that the program is applicable for any course where multiple choice and true-false question tests are suitable. Some of the techniques utilized will find application in many areas of programming.

## Introduction

The original idea of a test generation program came from several sources. North Dakota State University has successfully used mark sense cards and computer evaluation of test results[1] for about three years. Noting the success in this area, one of the computer programming instructors tried using multiple-choice tests with two sections of the class. He had considerable success using mark sense cards and computer evaluation, so more of the instructors teaching this course decided to try it. They expanded the idea to the point where each instructor would write a given number of questions on index cards, all cards would be filed together, and each person could draw out a given number of questions from this file when writing a test. To eliminate typing errors in the process of copying the questions from the index cards to the test sheet, it was decided to have the questions punched on IBM cards and then a stencil or ditto could be cut directly from the selected cards using a 407.

Concurrently another staff member at N.D.S.U. was experimenting with this same type of approach with a course in tests

[1] This evaluation has been accomplished by using a modified version of the Northeastern University Mark Sense Testing Scoring Program written by Robert M. O'Brian (Users Group Library classification is 13.0.003).

and measurements.[2] His work also involved manually pulling a set of questions and cutting a stencil on a 407.

A short program was then written to generate a list of random numbers enabling non-professional personnel to pull the desired cards for the test. The logical culmination of the procedure was to let the computer generate the test directly from the master test file. This is the program that is discussed on the following pages.

## Program Outline

As with any program one big problem is the selection of a suitably versatile input data format. It was desired to have a coding that would allow easy editing and changing of questions, a maximum amount of space for the question material, and still be simple. Allowing a maximum of 999 questions tied up three card columns, so to minimize further excessive card usage one column was allotted to each of the following: the answer, the card number, and the figure numbers. Since the Northeastern program only allows a single answer per question, using a single column for the answer is not a restriction. Using the card number as a supplemental code to the question number, it becomes possible to have ten cards per question (thus allowing for a possible 740 character question). Those few questions that require more than ten cards can be rewritten to make use of a figure. Using only one card column to represent a figure would seem to be a serious limitation; however, if all symbols are used except a record mark, a blank, and an asterisk, it is possible to code for a maximum of 47 figures.

Using this approach the following coding was decided upon.

Tests Questions:

CC 1: Answer of that question, i.e. A to E or T, F. This is usually found only on the last card of the question.

CC 2-4: Question number. This is punched on all the cards of the question.

CC 5: The card number of that question.

[2] The results of his work was presented as a paper "Some Practical Uses of Data Processing in Testing and Counseling" by Q. Stodola, D. Peterson, and M. Holoien to the National Counsel on Measurement in Education on February 11, 1965.

CC 6: Refers to a figure if applicable. If all symbols except ‡, *, and blank are used, a maximum of 47 figures could be accommodated.

CC 7-80: Actual questions (any information).

The figures are grouped together and placed after the question deck. They are coded as follows:

CC 1: Figure symbol (same symbol on all cards of the set). This is the cross reference to the character punched in CC 6 of the applicable cards in the question deck.

CC 2-3: On the first card in the set, the number of cards in the set.

CC 4-5: Card number of the figure. The number on the last card is the same as CC 2-3 of the first card.

CC 6: Asterisk (*), identifies the card as part of a figure.

When this program was used at N.D.S.U. in conjunction with a programming class the figures ranged from flow charts to segments of programs. By making the figures more complete than necessary it was possible to ask several different questions referencing the same figure. Using this technique, it has been found that the maximum of 47 figures is not a limitation.

The pilot cards which precede the actual question deck contain such information as:

a. The number of multiple-choice and/or true-false questions desired.

b. The minimum and maximum question number from which the selection is to be made.

c. The total quantity of the T-F questions in the master file and their numbers.

The program itself (compiled in Fortran II on a 40K, 1620) follows five basic steps to generate each different test:

a. Generation of a random number which will serve as the base from which future random numbers are obtained.

b. Reading of pilot cards which list the range of test question numbers to be considered and the size of test to be generated.

c. Creation of table of random numbers within the group specified.

d. Reading of master set of test questions and table search to determine which questions should be reproduced.

e. Typed listing of the answers.

The random number subroutine used with this program is Program No. 7.0.022 modified for Fortran II. The basic function of the random subroutine is to generate a floating point number between zero and one. The generated number is then multiplied by 1000 (if there are more than 100 questions in the master deck). If the resultant number is outside the range of the test question range as specified on the pilot cards, it is rejected and a new one is generated. This process is continued until the desired quantity of question numbers is stored in the table.

Upon completion of the question number table, the question deck is read in a card at a time, and those cards with question numbers matching the numbers stored in the table are reproduced. During this time the figure codes encountered in the reproduced questions are stored in table form. This table is searched when the master file of figures is read, and the appropriate figures are reproduced at the end of the test.

If a program of this type is to fully accomplish its purpose, there should be minimum amount of manual intervention between the time it is desired to write the test and the actual time the test is passed out to the student. This is accomplished by an editing feature in the program. Two cards precede the pilot cards which specify the size of the test desired. The first card is reproduced as the first card of the test giving course number, date, etc. and the second card is reproduced as the first line on succeeding pages of the test. The questions are renumbered in sequence for the output and all the pages are properly numbered so the output can be listed directly on continuous form stencil or ditto with a 407 Accounting Machine. The panel used in the 407 is basically an 80-80 listing with these modifications.

a. CC 1 is not listed.

b. One (1) in CC 1 extra space before printing card.

c. Two (2) in CC 1 skip to new sheet before printing card.

The test, which is now in final form, can be run off directly on a duplicating machine. While the computer is punching out the selected test questions the console typewriter lists the original test question number, the new test question number, and the answer. The student normally records his answers directly on mark sense cards which are graded and analyzed on the computer. This completes the automated testing and grading procedure.

In this program the question deck coding is not fully utilized, but provisions have been made within the coding structure to write a second program to check for various sequencing and format errors that might occur if the master deck was handled or modified frequently. It was the original intent when writing this program to include this checking feature, but because of limited memory space it had to be left out.

It was quickly realized that an instructor could not adequately cover the necessary topics in a test if he limited his tests to multiple-choice and true-false selections. To broaden the range of the tests, supplementary questions requiring the writing of programs or segments of programs were added by the instructor. These resulting tests have been used successfully at N.D.S.U. for two years by as many as four different instructors teaching up to seven sections of our introductory programming course in one quarter.

Some obvious advantages of such a system is a greater consistency of subject matter covered in the tests and the ease in creating the tests. In all fairness, one must point out the disadvantages of this system. The type and the range of test questions must be constantly reviewed and updated as the emphasis on different topics is changed. A reasonable number of questions should be added periodically to stay ahead of the students who undoubtedly have a better file of test questions than the instructor. Although care is taken to prevent old copies of tests from getting into student's files, one would be very naive in thinking that this did not occur to a limited degree. The only way of minimizing this problem is to build up a substantial quantity of test questions for the master deck.

In summary, the major limitation of any multiple-choice testing procedure is the construction of clear, concise questions that can not be interpreted as having a dual meaning. The choice of answers must also include logical responses for the student that does not have a complete grasp of the subject matter involved. This is necessary in any test construction.

This program offers a convenient tool for the instructor to free him of some of the paper work involved in making up tests. An important by-product of this procedure evolves when several instructors are teaching the same course. By creating a pool of questions the instructors will find that a better quality test will result. With several instructors drawing from the same set of questions, the tests generated for different sections of the same course are apt to be more consistent in both difficulty and subject matter.

## SAMPLE TEST

### INPUT

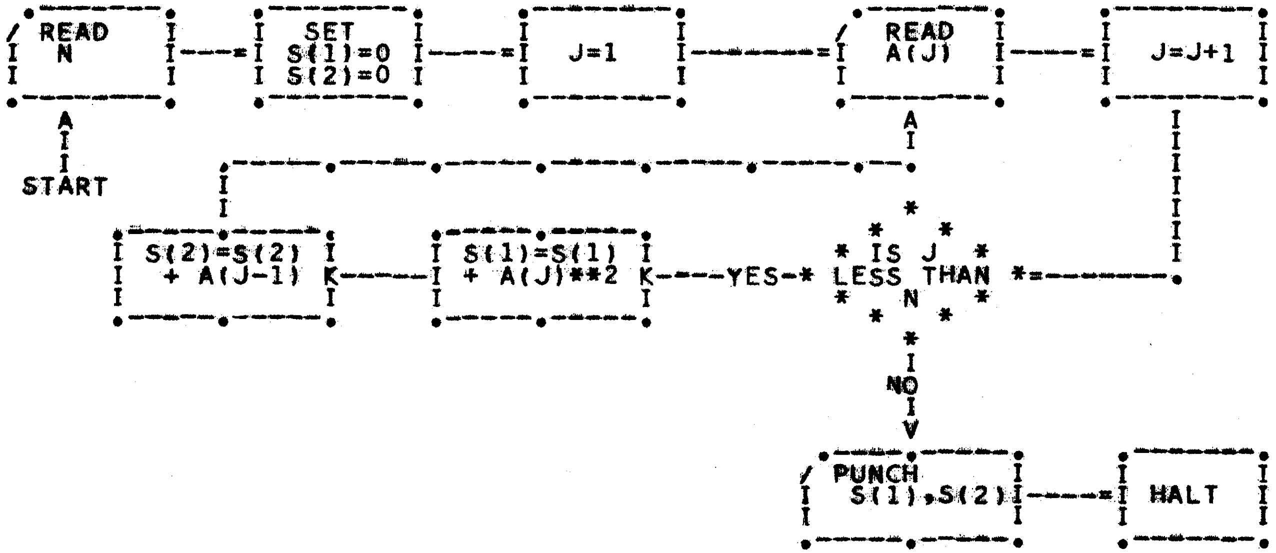
```
NOTE..... THE COMPLETE MASTER FILE WOULD BE READ AT THIS POINT.  FOR CONVENIENCE
          ONLY THE RANDOMLY SELECTED QUESTIONS ARE REPRODUCED.

   7714ASSUMING THAT THE PROGRAM IN THE FLOW CHART OF FIGURE 4 RUNS AND IF N=10,
   772 A(1)=1,A(2)=2,.........,A(N)=N.
   773 WHEN THE PROGRAM HALTS, THE VALUE OF S(2) WILL BE
A0774 A. 45   B. 55   C. 36   D. CANNOT BE DETERMINED

F08014REFER TO THE FLOW CHART OF FIGURE 4.  N CARDS WILL BE PUNCHED.

T0891 THE MEMORY OF THE IBM 1620 IS MADE UP OF ARRAYS OF FERRITE CORES.

NOTE..... SAMPLE FIGURES WHICH WOULD NORMALLY APPEAR AT THIS POINT HAVE BEEN LEFT
          OUT TO CONSERVE SPACE.
```

..........................................................................................

### OUTPUT AS LISTED ON THE 407

```
      SAMPLE TEST          COMPUTER SCIENCE NNN      OCTOBER 1965

            MULTIPLE-CHOICE.       MARK ONLY ONE ANSWER PER QUESTION.

  1. ASSUMING THAT THE PROGRAM IN THE FLOW CHART OF FIGURE 4 RUNS AND IF N=10,
     A(1)=1,A(2)=2,.........,A(N)=N.
     WHEN THE PROGRAM HALTS, THE VALUE OF S(2) WILL BE
     A. 45   B. 55   C. 36   D. CANNOT BE DETERMINED

               TRUE-FALSE.          MARK A FOR TRUE, B FOR FALSE.

  2. REFER TO THE FLOW CHART OF FIGURE 4.  N CARDS WILL BE PUNCHED.

  3. THE MEMORY OF THE IBM 1620 IS MADE UP OF ARRAYS OF FERRITE CORES.
```

FIGURE 4



### TYPEWRITER OUTPUT

```
1 ... A  -  77
2 ... F  -  80
3 ... T  -  79
```

```
C     TEST GENERATION
      DIMENSION KFIG(35),NBR(999),QUES(19),NBTF(999),JTF(999),H(17)
C     INITIAL GENERATION OF A RANDOM NUMBER
 1010 PRINT 5000
 5000 FORMAT(/22HSET SW 1 OFF TO START.)
      PAUSE
      IF(SENSE SWITCH 1)1010,1020
 1020 PRINT 5010
 5010 FORMAT(/39HTO SELECT A RANDOM NUMBER, SET SW 1 ON.)
      KNT=0
 1030 RN=RAND(0.51)
      KNT=KNT+1
      IF(KNT-500)1035,1020,1020
 1035 IF(SENSE SWITCH 1)1038,1030
 1038 PRINT 5015,RN
 5015 FORMAT(9HRANDOM = ,F10.8/)
C     HEADER CARD FOR FIRST PAGE OF THE TEST
      READ 50
 50   FORMAT(80H1234567891         20         30         40         50
     1   60         70         8)
C     HEADER CARD FOR ALL FOLLOWING PAGES OF THE TEST
      READ 940,(H(I),I=1,17)
 940  FORMAT(12X,17A4)
      PRINT 50
      PUNCH 50
C     INITIALIZATION OF CONSTANTS
      KOUNT = 1
      PROBN=1.
      IPAGE=2
 104  IF(SENSE SWITCH 9)105,105
 105  NF=0
      MFIGS=0
      IP=0
      NEWTF=0
      ITOTF=0
      ITOTM=0
      LTF=0
C     NM IS THE NUMBER OF MULTIPLE-CHOICE QUESTIONS DESIRED ON THE TEST,
C     NTF IS THE NUMBER OF TRUE-FALSE QUESTIONS DESIRED ON THE TEST,
C     NS IS THE FIRST NUMBER IN THE MASTER QUESTION FILE FROM WHICH
C     QUESTIONS ARE TO BE SELECTED,
C     ITEMS IS THE LAST NUMBER FROM WHICH QUESTIONS ARE TO BE SELECTED,
C     N IS THE TOTAL NUMBER OF TRUE-FALSE QUESTIONS IN THE MASTER FILE
      READ 5025,NM,NTF,NS,ITEMS,N
 5025 FORMAT(2I5/2I5/I5)
      IF(N)9000,1080,1055
C     JTF(K) ARE THE ACTUAL NUMBERS OF THE TRUE-FALSE QUESTIONS IN THE
C     MASTER FILE OF QUESTIONS
 1055 READ 5030,(JTF(K),K=1,N)
 5030 FORMAT(16I5)
C     ERROR CHECKING
 1060 DO 1070 LL=1,N
       IF(JTF(LL)-ITEMS)1064,1064,1070
 1064 IF(JTF(LL)-NS)1070,1066,1066
 1066 LTF=LTF+1
 1070 CONTINUE
 1080 IF(NTF-LTF)1085,9012,9010
 1085 IF(ITEMS-LTF-NM-NS+1)9015,9017,1100
C     GENERATION OF TABLE OF RANDOM QUESTION NUMBERS
 1100 SP=RAND(RN)
      IF(ITEMS-99)1110,1110,1250
 1110 ISP=SP*100.
 1120 IF(ISP-ITEMS)1125,1125,1100
 1125  IF(ISP)1100,1100,1130
 1130 IF(ISP-NS)1100,1140,1140
 1140 IF(ITOTM)1160,1160,1145
 1145 DO 1150 II=1,ITOTM
       IF(NBR(II)-ISP)1150,1100,1150
 1150 CONTINUE
 1160 IF(ITOTF)1180,1180,1165
 1165 DO.1170 JJ=1,ITOTF
      IF(NBTF(JJ)-ISP)1170,1100,1170
 1170 CONTINUE
 1180 IF(N)1200,1200,1185
 1185 DO 1190 NI=1,N
       IF(ISP-JTF(NI))1190,1220,1190
 1190 CONTINUE
 1200 IF(ITOTM-NM)1205,1210,1210
 1205 ITOTM=ITOTM+1
       NBR(ITOTM)=ISP
 1208 IF(ITOTM-NM)1100,1210,1210
 1210 IF(ITOTF-NTF)1100,1,1
 1220 IF(ITOTF-NTF)1225,1208,1208
 1225 ITOTF=ITOTF+1
       NBTF(ITOTF)=ISP
      GO TO 1208
 1250 ISP=SP*1000.
       GO TO 1120
C     END OF TABLE GENERATION
 1    ID1=0
      ID2=0
C     READING OF MASTER QUESTION FILE
      IF (SENSE SWITCH 9)  4010, 4010
 4010 IF(KOUNT-44)106,106,108
 108  PUNCH 6,IPAGE,(H(I),I=1,17)
      IPAGE=IPAGE+1
      KOUNT=1
 106  PUNCH 924
 924  FORMAT(/1H1,12X16HMULTIPLE-CHOICE.6X24HMARK ONLY ONE ANSWER PER,
     110H QUESTION.)
      KOUNT=KOUNT + 3
 30   READ 3,IANS,IDENT,IFIG,(QUES(I),I=1,19)
      IF(IDENT-0)90,91,90
 3    FORMAT(A1,I3,1XA1,18A4,A2)
 90   IF(ID1-IDENT)31,30,31
 31   IF(ID2-IDENT)32,38,32
C     SEARCH TABLE TO SEE IF QUESTION NUMBER COMPARES
 32   IF(NTF)4100,4100,3200
 3200 DO 40 K=1,NTF
      IF(IDENT-NBTF(K))40,210,40
 40   CONTINUE
 4100 IF(NM)4400,4400,4200
 4200 DO 44 I=1,NM
      IF(IDENT-NBR(I))44,45,44
 44   CONTINUE
 4400 ID1=IDENT
      GO TO 30
 210  IF(NEWTF)212,212,45
 212  IF(KOUNT-44)214,214,218
 218  PUNCH 6,IPAGE,(H(I),I=1,17)
      IPAGE=IPAGE+1
      KOUNT=1
```

```
  214 PUNCH 930
  930 FORMAT(/1H1,16X11HTRUE-FALSE.7X29HMARK A FOR TRUE, B FOR FALSE.)
      KOUNT=KOUNT+3
      NEWTF=1
   45 ID2=IDENT
C     PUNCH QUESTION JUST READ
      IF(KOUNT-46)46,46,48
   48 PUNCH 6,IPAGE,(H(I),I=1,17)
      IPAGE=IPAGE+1
      KOUNT=1
   46 PUNCH 918,PROBN,(QUES(I),I=1,19)
      PROBN=PROBN+1.
  918 FORMAT(1H1,F4.0,1X,18A4,A2)
      KOUNT=KOUNT+2
      IF(IFIG)110,33,110
  110 IF(IFIG-70000)120,33,120
C     PLACE FIGURE SYMBOL IN FIGURE TABLE
  120 IF(NF)130,130,122
  122 DO 125 NFN=1,NF
      IF(IFIG-KFIG(NFN))125,33,125
  125 CONTINUE
  130 NF=NF+1
      KFIG(NF)=IFIG
   38 PUNCH 922,(QUES(I),I=1,19)
  922 FORMAT(6X,18A4,A2)
      KOUNT=KOUNT+1
   33 IF(IANS-0)34,30,34
   34 IF(IANS-70000)36,30,36
C     TYPING ANSWER ON CONSOLE TYPEWRITER
   36 AN=PROBN-1.
      PRINT 920,AN,IANS,IDENT
  920 FORMAT(F4.0,2H..,A1,2X1H-,2X,I3)
      GO TO 30
C     READING OF FIGURES
  135 READ 910,MFIG,ICC,(QUES(I),I=1,19)
  910 FORMAT(A1,I2,3X,18A4,A2)
      IF(MFIGS-MFIG)140,142,140
  142 IF(IP)170,91,170
  140 IF(NF)152,152,141
C     SEARCH FIGURE TABLE
  141 DO 150 NK=1,NF
      IF(MFIG-KFIG(NK))150,160,150
  150 CONTINUE
  152 MFIGS=MFIG
      IP=0
   91 IF(SENSE SWITCH 9)200,135
C     PUNCH FIGURE JUST READ
  160 IF(KOUNT+ICC-55)162,175,175
  175 PUNCH 6,IPAGE,(H(I),I=1,17)
  6   FORMAT(1H2,4HPAGE,I3,4X,17A4)
      IPAGE=IPAGE+1
      KOUNT=1
  162 PUNCH 915,(QUES(I),I=1,19)
  915 FORMAT(/1H1,5X,18A4,A2)
      KOUNT=KOUNT+3
      IP=1
      MFIGS=MFIG
      GO TO 91
  170 PUNCH 922,(QUES(I),I=1,19)
      KOUNT=KOUNT+1
      GO TO 91
```

```
 200  PRINT 926
 926  FORMAT(/47HSW 2 OFF FOR NEW TEST, ON TO CONTINUE OLD TEST.)
       PAUSE
      IF(SENSE SWITCH 2)104,1010
C     ERROR MESSAGES
 9000 PRINT 8600
 8600 FORMAT(/23HCANNOT HAVE NEGATIVE N.)
 9006 PAUSE
      GO TO 1010
 9012 PRINT 8680
 8680 FORMAT(/48HREQUESTED NO. OF TRUE-FALSE QUESTIONS EQUALS NO.,
     111H AVAILABLE.)
       GO TO 1085
 9010 PRINT 8660
 8660 FORMAT(/52HIMPOSSIBLE TO REQUEST MORE TRUE-FALSE QUESTIONS THAN,
     115H ARE AVAILABLE.)
      GO TO 9006
 9015 PRINT 8670
 8670 FORMAT(/52HIMPOSSIBLE TO REQUEST MORE MULTIPLE-CHOICE QUESTIONS,
     120H THAN ARE AVAILABLE.)
      GO TO 9006
 9017 PRINT 8690
      GO TO 1100
 8690 FORMAT(/53HREQUESTED NO. OF MULTIPLE-CHOICE QUESTIONS EQUALS NO.,
     111H AVAILABLE.)
      END
```

Computer Center
RANDOLPH-MACON COLLEGE
Ashland, Virginia 23005

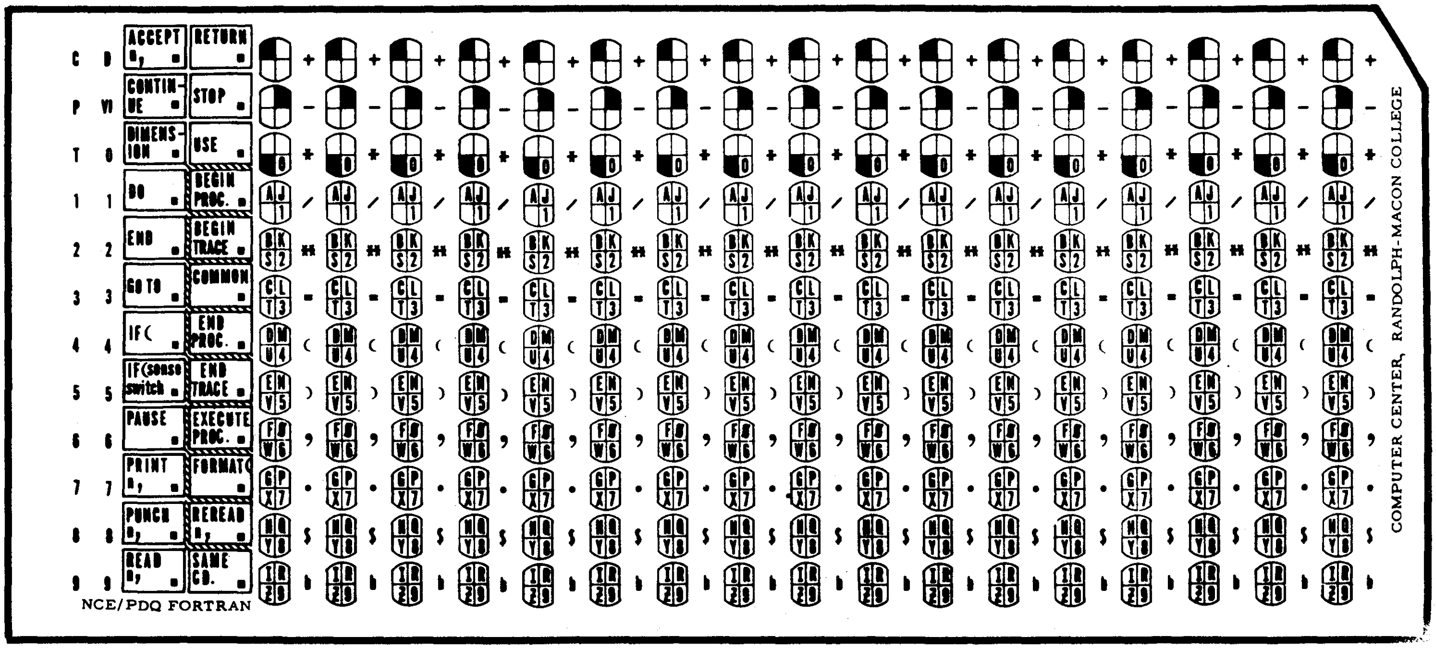
# PORT-A-PUNCH FORTRAN SOURCE AND DATA CARDS

Richard E. Grove

A paper given before the 1620 Users Group Joint Eastern-Midwestern Meeting, Americana Hotel, New York, October 6, 7, 8, 1965.

The Randolph-Macon Port-a-punch System (RAMPUS) allows students to "punch" computer program source and data cards without having access to a keypunch. The complete RAMPUS includes the capability of generating source cards for several programming languages and data cards of several types. This paper is restricted to those features related to FORTRAN source and data cards.

RAMPUS uses specially formated, pre-perforated cards--IBM port-a-punch cards--which may be punched using a simple stylus. These cards are accepted by the RAMPUS conversion computer program and source or data cards are output in standard FORTRAN format. The RAMPUS FORTRAN card is shown in the figure below. It has been designed for use with NCE Load and Go FORTRAN (2.0.029), PDQ FORTRAN (2.0.031) and other compatible FORTRAN processors including 1620 FORTRAN/Format.



The numeric digits in the first two columns are used for statement numbers. Note that the RAMPUS restricts statement number to at most two digits but this has proved to be adequate for student programs. The entries C, P, T, D and VI have special uses as follows:

**C** If a C is punched in the first column, the card is considered to be a FORTRAN comment card, a C appears in cc 1 of the output, and the message contained in the body of the RAMPUS card starts in cc 7 of the output card.

**T**
**P** These are used only in NCE FORTRAN to take advantage of the limited capability for alphameric output. If punched, a P or T is placed in cc 1 of the output and the message contained in the body of the RAMPUS card starts in cc 2 of the output card.

**D** This is used to indicate that the card contains data. The content of the body of the RAMPUS card is placed in cc 1 and following card columns of the output card.

**VI** Punching this entry will cause the number 1 (one) to be punched in cc 6 and the content of the body of the RAMPUS card will begin in cc 7 of the output card. This is used for continuation cards in PDQ for input/output lists and format statements.

Consider now the two columns of rectangles which contain reserved FORTRAN words. Punching of the hole indicated by the small solid square symbol in each of the several FORTRAN word rectangles has the following effect in the output FORTRAN source card produced by the RAMPUS conversion program:

**ACCEPT n, ■** produces ACCEPT, which must be followed by the list of input variables. The n is ignored in NCE FORTRAN. For PDQ, if the next symbol or the next two symbols are numeric, they are interpreted as format number and are placed to the left of the comma.

**CONTIN- UE ■** produces CONTINUE This is a complete FORTRAN statement.

**DIMENS- ION ■** produces the word DIMENSION which must be followed by the list of subscripted variables with the array size indicated.

DO ■ produces the word DO which must be followed by the statement number giving the range of the DO, a fixed point variable name, an equal sign, and the indexing parameters.

END ■ produces the word END which is a complete FORTRAN statement.

GO TO ■ produces the words GO TO which must be followed by the statement number of the next command to be executed. This may also be used for the computed GO TO.

IF( ■ produces the entry IF ( which must be followed by an arithmetic expression, a close parenthesis and three statement numbers separated by commas.

IF(sense switch ■ produces the phrase IF (SENSE SWITCH and must be followed by the switch number (1, 2, 3, 4 or 9), a close parenthesis, and two statement numbers separated by a comma.

PAUSE ■ produces the word PAUSE which is a complete FORTRAN statement.

PRINT n, ■ produces the entry PRINT, which must be followed by the list of output variables. The n is ignored in NCE FORTRAN. For PDQ, if the next symbol or the next two symbols are numeric, they are interpreted as format numbers and are placed to the left of the comma.

PUNCH n, ■ produces the entry PUNCH, which must be followed by the list of output variables. The n is ignored in NCE FORTRAN. For PDQ, if the next symbol or the next two symbols are numeric, they are interpreted as format numbers and are placed to the left of the comma.

READ n, ■ produces the entry READ, which must be followed by the list of input variables. The n is ignored in NCE FORTRAN. For PDQ, if the next symbol or the next two symbols are numeric, they are interpreted as format numbers and are placed to the left of the comma.

RETURN ■

produces the word RETURN which is a complete statement in NCE FORTRAN. In PDQ it must be followed by the number of the procedure from which control is passed.

STOP ■

produces the word STOP which is a complete NCE FORTRAN statement. In PDQ it may be followed by a fixed-point constant for identification purposes.

USE ■

produces the word USE which is used only in NCE FORTRAN. This must be followed by the statement number of the first command of the subprogram.

BEGIN PROC. ■

produces the words BEGIN PROCEDURE which is used only in PDQ FORTRAN and must be followed by a fixed-point constant which identifies the subprogram.

BEGIN TRACE ■

produces the words BEGIN TRACE which is used only in PDQ FORTRAN and is a complete command.

COMMON ■

produces the word COMMON which is used in PDQ FORTRAN and must be followed by the list of variables so declared.

END PROC. ■

produces the words END PROCEDURE which is used only in PDQ FORTRAN and must be followed by a fixed-point constant identifying the subprogram.

END TRACE ■

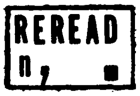
produces the words END TRACE which is used only in PDQ FORTRAN and is a complete command.

EXECUTE PROC. ■

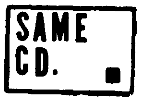
produces the words EXECUTE PROCEDURE which is used only in PDQ FORTRAN and must be followed by a fixed-point number identifying the subprogram to which control is to be passed.

FORMAT( ■

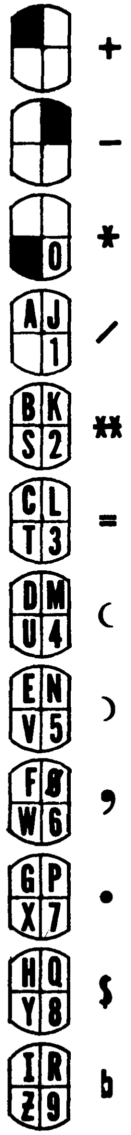
produces the entry FORMAT ( which is used in PDQ FORTRAN. This must be followed by the format specifications and a close parenthesis.



produces REREAD, which is used only in PDQ FORTRAN. The first symbol or the first two symbols in the body of the RAMPUS card must be numeric. These are considered to be format numbers and will be inserted to the left of the comma by the RAMPUS conversion program.



This punch produces no information on the output card. It is used to allow the content of several RAMPUS cards to be punched on the same FORTRAN source card. The standard format of the FORTRAN source card requires that the statement begin in cc 7 and not extend beyond cc 72. This is more information than can be contained on one RAMPUS CARD. Information for a single FORTRAN statement may be continued onto a second RAMPUS card if the SAME CD. box is punched on the second (and following) RAMPUS cards. In this case the content of the subsequent card(s) is placed immediately to the right of the content of the preceeding RAMPUS card on the output FORTRAN source card. There is no restriction on the number of successive SAME CD. cards except, of course, the source statement may not extend beyond cc 72. The RAMPUS conversion program does not check for violation of this restriction.



It will be noted that the major portion of the RAMPUS card is made of alternating columns of two types as shown on the left. The columns which have the oval boxes will be known as alphameric columns and the other type will be known as special character columns.

The alphameric columns may contain either one punch to indicate numeric digits or two punches to indicate alphabetic characters. To punch numeric digits, it is necessary only to punch the box which contains the desired digit in the lower right quadrant.

Two punches are required to indicate an alphabetic character. A special quadrant coding scheme has been devised which obviates the necessity of knowing the Hollerith card code. First, find the box which contains the desired character and punch out the perforated hole in this box. Next, note the quadrant (upper right, upper left, or lower left) in which the alphabetic character was found. Go now to the top three boxes in the same column and punch the box for which the quadrant noted above is solidly colored. Thus, an A would be indicated by punches in the top box and in the 4th box from the top. A Y would be indicated by punches in the 3rd box from the top and the box next to the bottom.

In the special character columns, a single-punch will cause the indicated symbol to be produced in the FORTRAN source card by the RAMPUS conversion program. The only exception to this statement is the bottom entry in the special character column, b. A punch in this position will cause a blank card column to be produced by the RAMPUS conversion program.

For both alphameric and special character columns, if no punch is made, the RAMPUS conversion program ignores the column and no entry is made in the output FORTRAN source or data card. To obtain a blank card column in the final source card, the b symbol must be punched in the RAMPUS card.

No more than one punch may be made in any of the special character columns. In the alphameric columns, a single punch is used to indicate a numeric digit and two punches, associated with the quadrant coding scheme, are used to indicate alphabetic characters. Any other combination of punches in a given column of the RAMPUS card will yield an erroneous code and may cause a READ CHECK error condition.

The RAMPUS conversion program for FORTRAN source and data cards has been designed to occupy that portion of core devoted to the pseudo-instructions and symbol table in NCE FORTRAN. Because of this feature, the NCE FORTRAN processor may exist unaltered in core before, during, and after the RAMPUS conversion program is loaded and executed. Execution of the first NCE FORTRAN program will overlay the RAMPUS conversion program and it must be loaded prior to each batch of RAMPUS conversions.

The IBM port-a-punch cards were printed so that the punched columns correspond to the even card columns of a standard card. Since this FORTRAN source and data card conversion is only part of a larger system, RAMPUS FORTRAN source cards are pre-punched with an F in (true) card column 1. The conversion program described here will convert any input card containing an F in cc 1 into standard FORTRAN source card format. Any other punch in cc 1 will cause an image of the input card to be punched.

Because of the experimental nature of this project when it was initiated, the RAMPUS cards were designed at the Randolph-Macon College Computer Center and printed locally on a Multilith 1250 offset duplicator. Few would care to do this and suitable cards can be obtained from IBM for an initial cost of $45.00 for the electroplate and a set-up charge of $35.00 on each order (no matter what size) plus the cost of cards at $2.52 per thousand. While card costs are approximately three times the cost of standard cards, the only reasonable alternative to port-a-punch cards is the use of mark-sense cards which require a monthly rental of about $155.00 for an IBM 514 reproducing punch with the mark-sense special feature.

For any who may be fearful, there has not yet been a single card read failure on the 1622 while using port-a-punch cards at the Randolph-Macon College Computer Center.

The standard IBM port-a-punch holder and stylus may be used with this system. This is, however, slow, awkward, and expensive. A better arrangement is to use a simple stylus and a small piece (approximately 4" x 8") of carpeting in the form of a burlap-like surface backed by 1/8" of foam rubber. This is available in many department stores under the trade name "Tex-a-weave".

Best separation of the chip from the card is obtained if the cushion is used cloth-side up. A convenient stylus is made by pushing a straight pin into the eraser of an ordinary wood pencil and clipping the head to leave about 1/4" of shank extending from the eraser. "Commerce" straight pins available from stationery suppliers have larger shanks and are more suitable than those used for sewing. The cost of stylus and foam pad is less than ten cents.

The RAMPUS conversion program has been written in 1620/1710 SPS for a model 1 1620 with 20 K and requires indirect addressing. A copy of the source program and the condensed object deck may be obtained from the author.

# NUMERICAL INTEGRATION USING GAUSS'S QUADRATURE FORMULA

RICHARD D. ROSS
COMPUTER CENTER
UNIVERSITY OF MISSISSIPPI

The problem of numerical integration has always been an intriguing one for mathematicians and scientists. Most numerical methods of integration recognize that the base points have to be equally spaced. This limitation is not imposed in the Gauss's quadrature formula.

A higher degree of accuracy of tabulation is now required as a result of scientific advances and especially of the increasing use of automatic computers.

Gauss's quadrature formula has been developed previously by the author with much emphasis placed upon the calculation of the roots and weight coefficients. The development of Gauss's quadrature formula is not given in this paper but can be found in "Gauss's Quadrature Formula" by Richard D. Ross as submitted to the faculty of the University of Mississippi Mathematics Department as a Masters' thesis.

All numerical integration formulas have the same general form which is

$$\int_a^b f(x)dx = \sum_{i=0}^{n} w_i f(x_i)$$

where n+1 weight factors, $w_i$, and the n+1 sample values, $f(x_i)$, are to be calculated.

By using Newton's Formula and the Interpolating Polynomial of Lagrange and also using Legendre's Orthogonial Polynomial, we are able to establish a bound for the error of Gauss's formula and are also able to calculate the root, $x_i$, and the weight coefficient, $w_i$, of Gauss's formula. The roots and weight coefficient for N = 1 to N = 18 are given in Appendix 1.

When calculating the roots and weight coefficients the limits (a,b) of the above formula are taken to be (-1, +1).

Extreme care has been taken in the calculations of the values of the roots and weight coefficients of Gauss's quadrature formula. To show the accuracy obtained a few examples will be taken and these examples will be compared to the trapezoidal rule, Simpson's rule, and Weddle's rule to show the relative accuracy of the four quadrature formulas. First consider

$$\int_0^6 (3x+1)dx \qquad (1)$$

using the six-point formula

$$
\begin{array}{lll}
x_0 = 0 & & f(x_0) = 1 \\
x_1 = 1 & & f(x_1) = 4 \\
x_2 = 2 & & f(x_2) = 7 \\
x_3 = 3 & & f(x_3) = 10 \\
x_4 = 4 & & f(x_4) = 13 \\
x_5 = 5 & & f(x_5) = 16 \\
x_6 = 6 & & f(x_6) = 19
\end{array}
$$

we obtain the following results, where N = n,

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 60.00000000000000000000000 |
| Simpson rule | N = 6 | 60.00000000000000000000000 |
| Weddle rule | N = 6 | 60.00000000000000000000000 |
| Gauss formula | N = 6 | 60.00000000000000000000000 |
| TRUE VALUE | | 60.00000000000000000000000 |

of which all four are completely accurate, which is to be expected.

In the above integration using Gauss's formula a transformation of the values of $x_i$,i=0,1,...,6 using

$$x_i = \frac{z_i\ (b-a)+a+b}{2}$$

where $z_i$ is the value of the roots of Gauss's formula for N = 6, and the integration of Equation 1 was calculated using Equation 2 which states that

$$\int_{-1}^{1} f(z)dz = \sum_{i=0}^{n} w_i f(z_i) \tag{2}$$

but for limits 0 to 6 the integral of Equation 2 will have the form

$$\int_{a}^{b} f(x)dx = \frac{b-a}{2} \sum_{i=0}^{n} w_i\ f(x_i) \tag{3}$$

Now consider the following integration with the same limits and the same number of subdivisions as Equation 1.

$$\int_0^6 (x^2 + 2x + 1)dx \tag{4}$$

and we obtain the following results

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 115.0000000000000000000000 |
| Simpson rule | N = 6 | 114.0000000000000000000000 |
| Weddle rule | N = 6 | 114.0000000000000000000000 |
| Gauss formula | N = 6 | 114.0000000000000000000000 |
| TRUE VALUE | | 114.0000000000000000000000 |

We see that the trapezoidal rule is not exactly accurate because the trapezoidal rule will only integrate correctly a straight line formula.

Consider the following examples and their results:

For

$$\int_0^6 (x^4 + 9.3x^3 + 7.2x^2 + 5.0x + 1.0)dx \tag{5}$$

we obtain

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 5345.500000000000000000000 |
| Simpson rule | N = 6 | 5183.600000000000000000000 |
| Weddle rule | N = 6 | 5182.800000000000000000000 |
| Gauss formula | N = 6 | 5182.800000000000000000000 |
| TRUE VALUE | | 5182.800000000000000000000 |

For

$$\int_0^6 (x^6 +5x^5 + 2x + 1)dx \tag{6}$$

we get

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 85450.00000000000000000000 |
| Simpson rule | N = 6 | 79114.00000000000000000000 |
| Weddle rule | N = 6 | 78918.00000000000000000000 |
| Gauss formula | N = 6 | 78912.85714285714285714286 |
| TRUE VALUE | | 78912.85714285714285714286 |

For

$$\int_0^6 (x^7 + 2x^6 + 3x^2 + 1)dx \tag{7}$$

we get

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 324704.0000000000000000000 |
| Simpson rule | N = 6 | 291890.0000000000000000000 |
| Weddle rule | N = 6 | 290274.0000000000000000000 |
| Gauss formula | N = 6 | 290155.7142857142857142857 |
| TRUE VALUE | | 290155.7142857142857142857 |

From Equation 5 we see that any equation of degree greater than 3, Simpson or the trapezoidal rule will not obtain the correct results. From Equation 6 we see that any equation of degree 6 or greater, Weddle's rule will not integrate correctly. By taking some subdivisions Equation 6 may be integrated more accurately by Weddle's rule; and also Simpson's rule and the trapezoidal rule are more accurate, which is to be expected. Taking 12 subintervals and integrating, we get

| | | |
|---|---|---|
| Trapezoidal rule | N = 12 | 80556.67187500000000000000 |
| Simpson rule | N = 12 | 78925.56250000000000000000 |
| Weddle rule | N = 12 | 78912.93750000000000000000 |
| Gauss formula | N = 12 | 78912.85714285714285714286 |
| TRUE VALUE | | 78912.85714285714285714286 |

In all cases only Gauss's rule gave the exact result because for n = 5, Gauss's formula will give the exact result if the integrand is an equation of degree 2n+1=11 or less, which in all instances was true, except in the example above where 2n+1=25.

Integrate Equation 5 using Gauss's formula for n=2 and we get

| | | |
|---|---|---|
| Gauss formula | N = 2 | 5182.800000000000000000000 |
| TRUE VALUE | | 5182.800000000000000000000 |

which is an exact result since 3.4.5 is indeed an equation of degree 2n+1=5 or less.

At this point there has been no restriction on the number of points to be used or the number of subdivisions, nor is there a necessity for equally spaced base points when using Gauss's formula.

Suppose that we wish to calculate the value of the integral

$$\pi = \int_0^1 \left(\frac{4}{1+x^2}\right) dx \qquad (8)$$

we obtain the following results for six subintervals (n=6),

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 3.136963066471263192574668 |
| Simpson rule | N = 6 | 3.141591780936043231125198 |
| Weddle rule | N = 6 | 3.141598445860740942708156 |
| Gauss formula | N = 6 | 3.141592656253749547166988 |
| TRUE VALUE | | 3.141592653589793238462643 |

We see that Weddle's rule is accurate to six decimal places and Gauss's formula has nine place accuracy. Now suppose we integrate Equation 8 using twelve equal subintervals and the results are shown below.

| | | |
|---|---|---|
| Trapezoidal rule | N = 12 | 3.140435246846850651454428 |
| Simpson rule | N = 12 | 3.141592640305379804414347 |
| Weddle rule | N = 12 | 3.141592677446341591048605 |
| Gauss formula | N = 12 | 3.141592653589793253511255 |
| TRUE VALUE | | 3.141592653589793238462643 |

Weddle's rule now has eight place accuracy but Gauss's formula has seventeen place accuracy. We are able at this time to see the real power in using Gauss's rule. Although the algebraic operations performed are very tedious and lengthy and are not recommended for tabulation by hand calculators, the results are extremely accurate.

To see the accuracy of Gauss's formula, let us integrate Equation 9 for n=1 to n=18 using Gauss's formula. We can see from the results given that as n increases the number of accurate decimal points is almost a linear function of n. For n=18 Gauss's formula integrates 3.4.9 accurately to twenty-five decimal places. The significance of the results shown is that it will take a polynomial of degree 2n+1=37 to represent the integral of a harmless looking equation as $\frac{4}{1+x^2}$ between the limits of 0 to 1.

$$\int_0^1 \left(\frac{4}{1+x^2}\right) dx = \frac{1-0}{2} \sum_{i=0}^{n} w_i\, f(x_i) \text{ for } n=1,2,\ldots,18,19. \tag{9}$$

| | | |
|---|---|---|
| Gauss formula | N = 1 | 3.147540983606557377049180 |
| Gauss formula | N = 2 | 3.141068139963167587476977 |
| Gauss formula | N = 3 | 3.141611905245805388072172 |
| Gauss formula | N = 4 | 3.141592639884752643813640 |
| Gauss formula | N = 5 | 3.141592611187586584212313 |
| Gauss formula | N = 6 | 3.141592656253749547166988 |
| Gauss formula | N = 7 | 3.141592653519118378361451 |
| Gauss formula | N = 8 | 4.141592653588243839589258 |
| Gauss formula | N = 9 | 3.141592653590046288835685 |
| Gauss formula | N = 10 | 3.141592653589781014051290 |
| Gauss formula | N = 11 | 3.141592653589793433433810 |
| Gauss formula | N = 12 | 3.141592653589793253511255 |
| Gauss formula | N = 13 | 3.141592653589793237099615 |
| Gauss formula | N = 14 | 3.141592653589793238513590 |
| Gauss formula | N = 15 | 3.141592653589792328462513 |
| Gauss formula | N = 16 | 3.141592653589793238462539 |
| Gauss formula | N = 17 | 3.141592653589793238462650 |
| Gauss formula | N = 18 | 3.141592653589793238462643 |
| TRUE VALUE | | 3.141592653589793238462643 |

Now let us consider another simple looking function such as

$$\int_{2}^{3.2} \ln(x)dx \qquad (10)$$

and get the following results for n=6 and n=12 for the trapezoidal rule, Simpson's rule, Weddle's rule, and Gauss's formula and also the value of Gauss's formula for n=7,8,9,10,11.

| | | |
|---|---|---|
| Trapezoidal rule | N = 6 | 1.8276551386820338362966800 |
| Simpson rule | N = 6 | 1.8278472579504855325586660 |
| Weddle rule | N = 6 | 1.8278474073079646336020780 |
| Gauss formula | N = 6 | 1.8278474085748222166539280 |
| | | |
| Gauss formula | N = 7 | 1.8278474085748222131990040 |
| Gauss formula | N = 8 | 1.8278474085748222131859860 |
| Gauss formula | N = 9 | 1.8278474085748222131859360 |
| Gauss formula | N = 10 | 1.8278474085748222131859360 |
| Gauss formula | N = 11 | 1.8278474085748222131859360 |
| | | |
| Trapezoidal rule | N = 12 | 1.8277993340159096030782750 |
| Simpson rule | N = 12 | 1.8278473991272015253388060 |
| Weddle rule | N = 12 | 1.8278474085545876966405770 |
| Gauss formula | N = 12 | 1.8278474085748222131859360 |
| TRUE VALUE | | 1.8278474085748222131859360 |

Again from the results shown above we are able to see that Gauss's formula is superior to the other three in question. Using Gauss's formula for n=9 we obtain an exact solution to Equation 10.

Consider the given equation

$$\int_{2.7}^{3.159} \frac{\left(\sin(x)e^{(\cos(x))}\right) \ln\left(\frac{x^{2.7}}{3.02}\right)}{\sinh(x^{2} -2.3x+1.73)(x^{3.97})} dx \qquad (11)$$

It can be seen that this is not a readily integrable equation. To integrate Equation 11 to any degree of accuracy using Weddle's rule would require the calculation of many base points. The results using Gauss's equation for n=1 to n=13 is shown below.

| | | |
|---|---|---|
| Gauss formula | N = 1 | 0.0000806183130952804160873674 8 |
| Gauss formula | N = 2 | 0.0000815327951784113771577337 0 |
| Gauss formula | N = 3 | 0.0000815380322342610783600251 2 |

| | | |
|---|---|---|
| Gauss formula | N = 4 | 0.00008153802146357601426297492 |
| Gauss formula | N = 5 | 0.00008153802131849522323566839 |
| Gauss formula | N = 6 | 0.00008153802131731542693985524 |
| Gauss formula | N = 7 | 0.00008153802131730857215620960 |
| Gauss formula | N = 8 | 0.00008153802131730856547759855 |
| Gauss formula | N = 9 | 0.00008153802131730856543469721 |
| Gauss formula | N = 10 | 0.00008153802131730856543465165 |
| Gauss formula | N = 11 | 0.00008153802131730856543465460 |
| Gauss formula | N = 12 | 0.00008153802131730856543465459 |
| Gauss formula | N = 13 | 0.00008153802131730856543465459 |

For n=12 an exact integral of 11 is shown accurate to the last decimal place.

## APPLICATION AND ILLUSTRATION OF GAUSS'S FORMULA

The following example will show how Gauss's formula may be used other than obtaining the numerical result of integration.

Suppose that we are given a complicated function and an upper and lower bound for the use of the function. But we wish to substitute for this involved function a simpler function that would be much easier to handle algebraically and will give the desired results to a specified degree of accuracy. There are many methods of arriving at a simpler function, but by using Gauss's formula and integrating the function between the bounds given using the value of n=i (where n is the number of base points minus one) for i=1,2,...m, m 18, we are able to determine by using Gauss's formula the value of m that will give the desired results which will mean that we will be able to replace the given function over the given range by a polynomial of degree 2m+1. For example, if we take Equation 11 and the results given for the integration of this equation, we are able to see that for n=3 we obtain a result that is accurate to six decimal places. Therefore, in this case, m=3 and 2m+1 = 7 which means that we should be able to replace the function given by a 7th degree polynomial and obtain the same results accurate to six decimal places. Suppose that we replace the integral 11 by the function

$$a_0 + a_1x + a_2x^2 + \ldots + a_7x^7 \quad . \tag{12}$$

By taking seven subintervals between the limits 2.7 and 3.159 and substituting these values in Equation 11 and by using the form of Equation 12 we are able to set up eight simultaneous equations with eight unknowns and are able to solve for the values of $a_0, a_1, \ldots a_7$, as given below. All results are shown accurate to twenty-five places.

|  |  |  |
|---|---|---|
| A(0) | = | 5.7060916151871860216008 24 |
| A(1) | = | -11.0639393747257281781 0977 |
| A(2) | = | 9.2198657838926481854 81282 |
| A(3) | = | -4.2781233647824118649 35882 |
| A(4) | = | 1.1929808907312123537 30572 |
| A(5) | = | -0.1997677036537830939 234936 |
| A(6) | = | 0.0185825173212763006 0092552 |
| A(7) | = | -0.000739943109486893173 7066830 |

Using these values calculated and integrating the function Equation 12 between the given limits we obtain for the value of the integral 0.00008153801358456494024586480. This result is accurate to six decimal places as originally stated. Thus, the very complex integral 11 is now reduced to a 7th degree equation and we are now able to manipulate algebraically with ease the transformed equation.

# APPENDIX I

| Roots ($x_i$) + and - | N = | Weight Coefficients ($w_i$) |
|---|---|---|
| 0.57735 02691 89625 76450 91488 | 1 | 1.00000 00000 00000 00000 00000 |
| 0.00000 00000 00000 00000 00000 | 2 | 0.88888 88888 88888 88888 88889 |
| 0.77459 66692 41483 37703 58530 | | 0.55555 55555 55555 55555 55556 |
| 0.33998 10435 84856 26480 26657 | 3 | 0.65214 51548 62546 14262 69360 |
| 0.86113 63115 94052 57522 39464 | | 0.34785 48451 37453 85737 30639 |
| 0.00000 00000 00000 00000 00000 | 4 | 0.56888 88888 88888 88888 88888 |
| 0.53846 93101 05683 09103 63144 | | 0.47862 86704 99366 46804 12915 |
| 0.90617 98459 38663 99279 76268 | | 0.23692 68850 56189 08751 42640 |
| 0.23861 91860 83196 90863 05017 | 5 | 0.46791 39345 72691 04738 98703 |
| 0.66120 93864 66264 51366 13995 | | 0.36076 15730 48138 60756 98335 |
| 0.93246 95142 03152 02781 23015 | | 0.17132 44923 79170 34504 02961 |
| 0.00000 00000 00000 00000 00000 | 6 | 0.41795 91836 73469 38775 51020 |
| 0.40584 51513 77397 16690 66064 | | 0.38183 00505 05118 94495 03697 |
| 0.74153 11855 99394 43986 38647 | | 0.27970 53914 89276 66790 14677 |
| 0.94910 79123 42758 52452 61896 | | 0.12948 49661 68869 69327 06114 |
| 0.18343 46424 95649 80493 94761 | 7 | 0.36268 37833 78361 98296 51504 |
| 0.52553 24099 16328 98581 77390 | | 0.31370 66458 77887 28733 79622 |
| 0.79666 64774 13626 73959 15539 | | 0.22238 10344 53374 47054 43559 |
| 0.96028 98564 97536 23168 35608 | | 0.10122 85362 90376 25915 25313 |
| 0.00000 00000 00000 00000 00000 | 8 | 0.33023 93550 01259 76316 45250 |
| 0.32425 34234 03808 92903 85380 | | 0.31234 70770 40002 84006 86304 |
| 0.61337 14327 00590 39730 87020 | | 0.26061 06964 02935 46231 87428 |
| 0.83603 11073 26635 79429 94297 | | 0.18064 81606 94857 40405 84720 |
| 0.96816 02395 07626 08983 55762 | | 0.08127 43883 61574 41197 18921 |
| 0.14887 43389 81631 21088 48260 | 9 | 0.29552 42247 14752 87017 38929 |
| 0.43339 53941 29247 19079 92659 | | 0.26926 67193 09996 35509 12269 |
| 0.67940 95682 99024 40623 43273 | | 0.21908 63625 15982 04399 55349 |
| 0.86506 33666 88984 51073 20966 | | 0.14945 13491 50580 59314 57763 |
| 0.97390 65285 17171 72007 79640 | | 0.06667 13443 08688 13759 35688 |
| 0.00000 00000 00000 00000 00000 | 10 | 0.27292 50867 77900 63071 44835 |
| 0.26954 31559 52344 97233 15319 | | 0.26280 45445 10246 66218 06888 |
| 0.51909 61292 06811 81592 57256 | | 0.23319 37645 91990 47991 85237 |
| 0.73015 20055 74049 32409 34162 | | 0.18629 02109 27734 25142 60976 |
| 0.88706 25997 68095 29907 51577 | | 0.12558 03694 64904 62463 46943 |
| 0.97822 86581 46056 99280 39380 | | 0.05566 85671 16173 66648 27537 |

```
                                   N =  11
0.12523 34085 11468 91547 24413  0.24914 70458 13402 78500 05624
0.36783 14989 98180 19375 26915  0.23349 25365 38354 80876 08498
0.58731 79542 86617 44729 67024  0.20316 74267 23065 92174 90644
0.76990 26741 94304 68703 68938  0.16007 83285 43346 22633 46525
0.90411 72563 70474 85667 84658  0.10693 93259 95318 43096 02547
0.98156 06342 46719 25069 05490  0.04717 53363 86511 82719 46159

                                   N =  12
0.00000 00000 00000 00000 00000  0.23255 15532 30873 91019 45896
0.23045 83159 55134 79406 55281  0.22628 31802 62897 23841 20900
0.44849 27510 36446 85287 79128  0.20781 60475 36888 50231 25233
0.64234 93394 40340 22064 39846  0.17814 59807 61945 73828 00466
0.80157 80907 33309 91279 42064  0.13887 35102 19787 23846 36018
0.91759 83992 22977 96520 65478  0.09212 14998 37728 44791 44217
0.98418 30547 18588 14947 28294  0.04048 40047 65315 87952 00215

                                   N =  13
0.10805 49487 07343 66206 62446  0.21526 38534 63157 79019 58764
0.31911 23689 27889 76043 56718  0.20519 84637 21295 60396 59240
0.51524 86363 58154 09196 52907  0.18553 83974 77937 81374 17165
0.68729 29048 11685 47014 80198  0.15720 31671 58193 53456 96019
0.82720 13150 69764 99318 97947  0.12151 85706 87903 18468 94148
0.92843 48836 63573 51733 63911  0.08015 80871 59760 20980 56332
0.98628 38086 96812 33884 15972  0.03511 94603 31751 86303 18328

                                   N =  14
0.00000 00000 00000 00000 00000  0.20257 82419 25561 27288 06195
0.20119 40939 97434 52230 06283  0.19843 14853 27111 57645 61189
0.39415 13470 77563 36989 72073  0.18616 10000 15562 21102 68000
0.57097 21726 08538 84753 72267  0.16626 92058 16993 93355 32012
0.72441 77313 60170 04741 61860  0.13957 06779 26154 31444 78045
0.84820 65834 10427 21620 06483  0.10715 92204 67171 93501 18697
0.93727 33924 00705 90430 77589  0.07036 60474 88108 12470 92673
0.98799 25180 20485 42848 95657  0.03075 32419 96117 26835 46284

                                   N =  15
0.09501 25098 37637 44018 53193  0.18945 06104 55068 49628 53967
0.28160 35507 79258 91323 04605  0.18260 34150 44923 58886 67636
0.45801 67776 57227 38634 24194  0.16915 65193 95002 53818 93120
0.61787 62444 02643 74844 66717  0.14959 59888 16576 73208 15017
0.75540 44083 55003 03389 51011  0.12462 89712 55533 87205 24762
0.86563 12023 87831 74388 04678  0.09515 85116 82492 78480 99251
0.94457 50230 73232 57607 79884  0.06225 35239 38647 89286 28438
0.98940 09349 91649 93259 61541  0.02715 24594 11754 09485 17805
```

```
                                    N =  16
0.00000 00000 00000 00000 00000  0.17944 64703 56206 52545 82608
0.17848 41814 95847 85585 06774  0.17656 27053 66992 64632 52754
0.35123 17634 53876 31529 71855  0.16800 41021 56450 04450 99669
0.51269 05370 86476 96788 62465  0.15404 57610 76810 28808 14344
0.65767 11592 16690 76585 03022  0.13513 63684 68525 47328 63179
0.78151 40038 96801 40692 52300  0.11188 38471 93403 97109 47897
0.88023 91537 26985 90212 29556  0.08503 61483 17179 18088 35345
0.95067 55217 68767 76122 27169  0.05545 95293 73987 20112 94405
0.99057 54753 14417 33567 54340  0.02414 83028 68547 93196 01099

                                    N =  17
0.08477 50130 41735 30124 22618  0.16914 23829 63143 59184 06563
0.25188 62256 91505 50958 89728  0.16427 64837 45832 72298 60540
0.41175 11614 62842 64603 59317  0.15468 46751 26265 24492 54174
0.55977 08310 73947 53460 78715  0.14064 29146 70650 65120 47319
0.69168 70430 60353 20787 48910  0.12255 52067 11478 46018 45183
0.80370 49589 72523 11568 24174  0.10094 20441 06287 16556 28146
0.89260 24664 97555 73920 60605  0.07642 57302 54889 05652 91291
0.95582 39495 71397 75518 11959  0.04971 45488 94969 79645 33352
0.99156 51684 20930 94673 00160  0.02161 60135 26483 31031 33426

                                    N =  18
0.00000 00000 00000 00000 00000  0.16105 44498 48783 69597 91721
0.16035 86456 40225 37586 80961  0.15896 88433 93954 34764 99483
0.31656 40999 63629 83199 01173  0.15276 60420 65859 66677 88624
0.46457 07413 75960 94571 72671  0.14260 67021 73606 61177 57403
0.60054 53046 61681 02346 96381  0.12875 39625 39336 22767 55203
0.72096 61773 35229 37861 70958  0.11156 66455 47333 99471 60204
0.82271 46565 37142 82497 89224  0.09149 00216 22449 99946 44644
0.90315 59036 14817 90164 26609  0.06904 45427 37641 22658 07067
0.96020 81521 34830 03085 27788  0.04481 42267 65699 60033 28389
0.99240 68438 43584 40318 90176  0.01946 17882 29726 47703 63118
```

# SERIES AND THE 1620

Written at Newark College of Engineering by:

Peter Byeff
Jerry Kleinbaum
Mark Sciora

# PREFACE

This paper, *Series and the 1620*, was written during the summer while the authors, high school students, were working at Newark College of Engineering on a grant from the National Science Foundation. At this time, we wish to thank the following, without whom this project would have been impossible: the National Science Foundation, the staff of Newark College of Engineering, Drs. Frederick Lehman and Phyllis Fox, Victor Miller and Joel Shwimer.

Peter Byeff
Jerry Kleinbaum
Mark Sciora

## Series and the 1620

The object of our work was to program the IBM 1620 computer to analyze number sequences. After comparing the elements of a given sequence, the computer determines if there is a relationship between the numbers. If a relationship is detected, succeeding terms in the series are calculated, printed, and/or punched on cards, depending upon what the operator desires.

For our purposes, it was necessary to define certain terms. We defined a sequence of numbers that has a definite, logical relationship between the elements of the sequence to be a series. A sequence, however, we defined as any collection of numbers.

Because of the largely algebraic nature of our work, our program very quickly became quite large. This necessitated our use of the 1620 model II computer coordinated with the 1311 disk storage unit. For this purpose, we divided the program into thirteen subprograms and one main program; each of the fourteen components was stored separately on the 1311 disk storage unit. Each component is separately called up from storage on the disk to the computer to ascertain whether the sequence in question is the series which the component has been programmed to recognize. If it is, then the elements of that series, if the operator so desires, are calculated up to the capacity of the machine. If it is not, then, through use of the Call Link statement, the next component is called for execution.

Probably the most challenging part of our project was programming the computer to recognize a series in which the increments form a repeating pattern. This proved difficult not only in programming the computer to recognize this type of series, but also in calculating its elements indefinitely. The following is an excerpt from the subprogram in which we solved this problem:

```
44 I=2
   J=1
45 IF(X(J)-X(I))46,48,46
46 IF(I-(IT-1))47,80,80
47 I=I+1
   GO TO 45
48 IK=I-1
49 IF(X(J+1)-X(I+1))47,50,47
50 IF(I+1-(IT-1))51,52,80
51 J=J+1
   I=I+1
   GO TO 49
52 NB=IT
53 JK=NB-IK
   IF(JK-IK)55,55,54
54 NB=JK
   GO TO 53
55 QQ=DATA(IT)+X(JK)
   TYPE 901
901 FORMAT(/12HSERIES NO. 2)
   DO 56 IA=1,IT
56 PRINT 500, DATA(IA)
500 FORMAT(/E14.8)
   PRINT 501,QQ
501 FORMAT(/E14.8)
   IF(SENSE SWITCH 3)800,59
800 CALL LINK (MAIN)
59 IF(JK-IK)70,61,80
70 QQ=QQ+X(JK+1)
   PUNCH 303, QQ
303 FORMAT(E14.8)
62 JK=JK+1
   GO TO 59
61 JK=1
   QQ=QQ+X(JK)
   PUNCH 304, QQ
304 FORMAT(E14.8)
   GO TO 59
80 CALL LINK(E5)
   END
```

In this excerpt, X represents the differences of the data which has been read in. DATA represents the elements which have been fed in. IT is a subscript of DATA and represents the relative position of all the data which has been read in as well as the last number which has been read in and the total number of elements in the sequence. The program works in the following manner. First, it is necessary to determine which, if any, of the increments matches the first increment indentically. This is accomplished by steps 44-47. If one is found which matches it, the program proceeds to determine whether the corresponding differences which succeed the first two are exactly alike. This is accomplished in steps 48-51. If this is found to be the case, all that remains to be done is to calculate the elements of the series indefinitely. The rest of the subprogram is devoted to this.

When it is known which increment is the same as the first, it is also known how many increments there are in the pattern. This number is then subtracted from the total number of terms in the series until the difference (remainder) is equal or less than the number of increments in the pattern. The remainder represents the increment in the pattern which must be added to the last element of the series to produce the next term. This can be seen in steps 52-end.

In the following pages is representative output for the various kinds of series that our program can solve for with a short explanation accompanying each one.

```
MAIN    07500 02714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
05RS

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      07500 01990  LOADED

 THIS IS AN ARITHMETIC PROGRESSION

INPUT DATA CHECK

 .23573000E+05

 .24671000E+05

 .25769000E+05

 .26867000E+05

 .27965000E+05

NEXT TERM

 .29063000E+05

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

Representative output for our program which determines whether the sequence in question is an arithmetic progression.

```
MAIN   0̄7500 02714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
04RS

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1     0̄7500 0̄1990  LOADED
B2     0̄7500 0̄1986  LOADED

 THIS IS A GEOMETRIC PROGRESSION

INPUT DATA CHECK

 .83000000E+02

 .65570000E+04

 .51800300E+06

 .40922237E+08

NEXT TERM

 .32328567E+10

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

Representative output for our program which determines whether the sequence in question is a geometric progression.

```
MAIN    0̄7500 0̄2714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
05R/S

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      0̄7500 0̄1990  LOADED
B2      0̄7500 0̄1986  LOADED
C3      0̄7500 0̄2150  LOADED

SERIES NO. 1

INPUT DATA CHECK

 .40723240E+07

 .40763610E+07

 .40804000E+07

 .40844410E+07

 .40884840E+07

NEXT TERM

 .40925290E+07

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

This is representative output from the part of our program which is designed to recognize a series in which the increments are in an arithmetic progression.

```
MAIN    0̄7500 0̄2714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
08RS

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      0̄7500 0̄1990  LOADED
B2      0̄7500 0̄1986  LOADED
C3      0̄7500 0̄2150  LOADED
D4      0̄7500 0̄2926  LOADED

SERIES NO. 2

INPUT DATA CHECK

 .58720000E+04

 .60440000E+04

 .86700000E+04

 .86870000E+04

 .88590000E+04

 .11485000E+05

 .11502000E+05

 .11674000E+05

NEXT TERM

 .14300000E+05

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

Series No. 2 is the type of series in which the increments form a repeating pattern

```
MAIN    0̄7500 0̄2714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
05RS

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      0̄7500 0̄1990  LOADED
B2      0̄7500 0̄1986  LOADED
C3      0̄7500 0̄2150  LOADED
D4      0̄7500 0̄2926  LOADED
E5      0̄7500 0̄2150  LOADED

SERIES NO. 3

INPUT DATA CHECK

 .87000000E+02

 .60900000E+03

 .60900000E+04

 .79170000E+05

 .12667200E+07

NEXT TERM

 .24067680E+08

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

In this type of series,
the factors form an
arithmetic progression.

```
MAIN    07500 02714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
08R/S

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      07500 01990  LOADED
B2      07500 01986  LOADED
C3      07500 02150  LOADED
D4      07500 02926  LOADED
E5      07500 02150  LOADED
F6      07500 02584  LOADED

SERIES NO. 4

INPUT DATA CHECK

 .53000000E+02

 .31800000E+03

 .41340000E+04

 .28938000E+05

 .17362800E+06

 .22571640E+07

 .15800148E+08

 .94800888E+08

NEXT TERM

 .12324115E+10

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

This series is much like Series No.2 except that in this case it is the factors that form a repeating pattern.

```
MAIN    0̄7500 0̄2714  LOADED

ENTER NUMBER OF TERMS IN SERIES, PUSH RS
07RS

ENTER ALL TERMS BY CARD, 1 TO A CARD, FLOATING PT
A1      0̄7500 0̄1990  LOADED
B2      0̄7500 0̄1986  LOADED
C3      0̄7500 0̄2150  LOADED
D4      0̄7500 0̄2926  LOADED
E5      0̄7500 0̄2150  LOADED
F6      0̄7500 0̄2584  LOADED
G7      0̄7500 0̄5130  LOADED
H8      07500 05050  LOADED

SERIES NO. 5

INPUT DATA CHECK

  .17830000E+04

  .19390000E+04

  .18400000E+04

  .20060000E+04

  .19070000E+04

  .20830000E+04

  .19840000E+04

NEXT TERM

  .21700000E+04

SW 3 ON TO ENTER NEW DATA

SW 3 OFF FOR NEXT 10 TERMS ON CARDS

SET SW 3,PUSH START
```

Series No.5 is a series in which the increments form an arithmetic progression which is separated by a constant every other term.

Note: The programs which identify Series No.6-8 are at present being rewritten and for that reason no representative data for those programs is ready for inclusion at this time.

# A COMPUTER SURVEY OF PROFESSIONAL SALARIES

1620 USERS GROUP MEETING

OCTOBER 7, 1965

AMERICANA HOTEL, NEW YORK CITY

William J. Abnett
Sun Oil Company
Research and Engineering Department
Philadelphia, Pennsylvania

# A COMPUTER SURVEY OF PROFESSIONAL SALARIES

## ABSTRACT

A Computer method of assembling, tabulating, and graphically presenting salary data is discussed.

The salary data of professional employees within a single organization or among several organizations is collected and assembled.

Within logical groupings of employees certain percentiles and averages are calculated for each year starting with the oldest employee and continuing until the most recent.

A mathematical curve of the form $y = A + Bx = Cx^2 + Dx^3$ is statistically fitted using the least squares technique for each percentile for each group of employees.

Letting the "y" axis represent dollars and the "x" axis, the year since the employee's first degree, the various percentiles are machine plotted for each group.

The tabulated data consists of the following for each year since the first degree:

1. Actual Salaries by percentiles
2. Least Squares Salaries by percentiles
3. Average Salary

# A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETING
OCTOBER 7, 1965

## Introduction

In this paper are discussed computer techniques of assembling, calculating percentiles, tabulating and graphically presenting professional employee salary data using an IBM 1620 Model II computer and an IBM 1627 Model I Plotter. The techniques have resulted from our experiences in executing both multi-company and our own internal Research and Engineering Department salary surveys.

A salary survey is of general interest to management, employees, economists, cost analysts, salary administrators, trade and professional associations and others. Managers may use a salary survey, along with other tools as a guide in hiring employees, compensating employees, and determining whether or not base salaries are competitive.

The discussion which follows is generally limited to the treating, handling, and obtaining of results. We will comment only briefly on the analysis of the end product.

## Professional Salaries

Although at this time we are interested in professional salaries the techniques which follow may, with slight modifications, be used for skilled labor, semi-professionals and management personnel.

A continuing discussion and disagreement exists as to the definition of a professional employee. For our purposes we shall assume that professional means an employee possessing a baccalaureate degree, recognizing that some non-degreed people are professional in the widest meaning of the word.

In any salary survey the population must be clearly defined. The population, for the purpose of this paper includes Research and Engineering technical professional employees - engineers, chemists, mathematicians, etc. - up to some predetermined level of supervision.

## The Variables

There are at least two choices for the unit of compensation:

1. Yearly salary

2. Monthly salary

As a side note some professional salary surveys report both base salary and total professional income.[1]

We have used the monthly salary and have limited this figure to the nearest whole dollar. From a computer standpoint a yearly salary base would require larger fields to handle the correspondingly larger data ranges. Monthly salary then is the dependent variable and, ultimately, will be plotted along the Y-axis.

For our independent variable, which we will later plot along the X-axis, we wish to show a time unit and we might choose:

1. Years of experience

2. Chronological age

3. Years since degree

If the third item is taken and modified slightly to Years Since First Degree the analyst will then be able to compare those with Master and/or Doctor degrees with those holding singular Bachelor degrees.

## Raw Data

Now having defined the variables, the raw data must be collected. At this point, it may be well to emphasize that the data are extremely confidential and should be treated accordingly. Individuals are known only by year of graduation and type of final degree. The data may be available directly from the company personnel records or, in the case of a survey including other companies, it may be necessary to prepare a questionnaire. For our purposes, all that is necessary for each individual in addition to his organization[2] identification is:

1. Year of first degree

[1] See for example: Business Economist's Salary Survey
(National Association of Business Economists, 1964)

[2] The organization may be a company, division, department or any other group.

2. Salary, dollars per month

3. Highest degree attained[3]

It may be somewhat obvious by this time that the raw data may be sorted for further processing using, but not limited to, any of the following parameters:

1. Company and/or group

2. Effective year of data; that is, this year and last year.

3. Degree

4. Professional discipline

Different colors may be used for the internally punched cards representing each group of raw data, making it easy to spot any cards out of order.

## Moving Averages

In developing various salary surveys it sometimes becomes necessary to handle small quantities of data and/or data for which the distribution - on a year by year basis - is something less than desireable.

The familiar moving average statistical technique[1] which, as you may recall is a statistical method generally used for smoothing seasonal data, may be used in these cases of limited data.

In applying the moving average technique a span of years or increments must be determined. If the data set is sufficiently large and all years represented, then the span of years may be set equal to unity and the survey executed without resorting to this technique. Before the computer was available hand calculations had been made to determine an optimum span. These lengthly hand calculations were abandoned after the trial reached a span of 5 years.

With the computer we were able, in a small fraction of the time

[3] See Figure 2

[4] See, for example, Chapter 11, Page 322, "The Analysis of Time Series, Measurement of Seasonal Fluctuations, Moving Averages." Introduction to Statistics, Frederick C. Mills, Henry Holt and Company, 1956.

required manually to enter the increment as a variable and try several sets of data each time increasing the increment from 2 to 8 years. From observations of the plotted data there appeared to be no reason not to use the originally determined 5 year increment.

Percentile Finder

From the raw data the user selects the data cards according to the group he desires and proceeds through a series of sub-programs, (See Figure 1) the first of which we term the Percentile Finder.

This sub-program reads the raw data representing the salaries of individuals, the starting year, the increment in years and the last year.

The program begins at the starting year, adds to this year the increment in years and selects from the raw data all data points (records) that fall within this range of years and places these into temporary storage.[5]

Referring to Figure 4 the data are sorted into ascending order of salaries starting with the first block in the temporary storage area. This is compared with the data in the second block and if the first is lower than the second, the first is then compared with the third, fourth, etc. until the entire used area of the temporary storage matrix is exhausted.

If the second value is lower than the first, the first and second values are interchanged so that the second value (being lower) is now in the place of the first and the comparisons above are continued. When the first block has been compared with the entire used matrix it has the lowest value in the used dimension area. The program then advances to the second block and repeats the entire process, switching higher and lower values as necessary. The program continues with the remaining blocks until the used area is completed. After the sorting the program then does one of the following:

1. More than 14 observations:

   a. Determines the 10, 25, 50, 75 and 90 percentile salaries. This program is flexible so that the 5, 25, 50, 75 and 95 percentiles could be used.

   b. Determines the average salary.

[5] See Figure 3.

2. Less than 15 but more than 4 observations:

   a. Determines the 25, 50, 75 salary percentiles.

   b. Determines the average salary.

3. Less than five observations:

   a. Determines the average salary.

The starting year is automatically increased by a single year, the pertinent data internally selected, sorted, and calculated. This entire procedure is repeated until the upper year of the time interval to be processed exceeds the last year to be processed.

## Sorting Percentile Finder Output

It is desireable to maintain the percentile finder output data deck intact inasmuch as it is used as input for later subprograms. To obtain input for the cubic equation fitting program, which follows another short program was written which reads the percentile finder output and if the input is part of the 10, 25, 50, 75, 90 percentile of the group being run a new card is punched. These newly punched cards for each group are then machine sorted into 10, 25, 50, 75, 90 percentiles for input to the cubic equation fitting program.

## Cubic Equation Fitting

The cubic equation fitting subprogram reads the sorted individual percentile output from the percentile finder and determines the cubic equation of the form below which best fits the data using the Gauss-Jordan[6] reduction scheme.

$$Y = K + A + Bx^2 + Cx^3$$

For each moving average group of years, X, the program calculates

[6] This is a matrix solving system discussed in several mathematics books; for instance, on Page 165, Linear Algebra by G. Hadley of the Addison Wesley Publishing Co., Reading, Mass., 1961. Perhaps one of the clearest may be found as a complete example starting on Page 59 of the book, Linear Programming and Theory of Games (paperback), A. M. Glicksman, John Wiley & Sons, Inc., 1963.

Y and reports the calculated observed Y values. Finally, the sum of squares of the deviations is given and the number of weighted observations (N).

The user selects a minimum number of observation points for a given year. If this minimum is not reached that year is disregarded. Any year having more than the minimum number of observation points is automatically weighted N times where: $N = \frac{\text{Number of observation points}}{\text{Minimum number of points}}$ and N is truncated to an integer.

## Tabulated Data

Two tables may be printed from the calculated data, one of which is a typical company survey shown on Table I.

This shows for each year since the Bachelor Degree, the number of people in the year,[7] the 10, 25, 50, 75, and 90 percentiles and the average salary.[8]

The second table, which we call the Survey List, is shown on Table II and for each group and for each year compares the actual percentiles and the least squares percentiles. This provides the analyst with an opportunity to note the deviations of the actual data from the least squares data for each year.

## The Plot

After the raw data have been sorted, merged, percentiles calculated, and equations developed to represent these percentiles, the results may be presented graphically using the computer and plotter.

It may be helpful to refer to the attached Figure 5 as we go through the following discussion.

Already the x-axis, years since the Bachelor Degree, has been decided upon, as has the y-axis, monthly salary in dollars.

[7] In a moving average survey and a span of one year this is the actual number of people in the year. If the span is larger than one year, the number of persons includes people who are in more than one calendar year.

[8] If the distribution is symmetrical, the 50th percentile will equal the average salary. Most data sets (years) are skewed; consequently, the 50th percentile frequently differs from the average salary.

The x-axis should extend from 0 years up to, say, 45 years. With the salaries of today we could place lower and upper limits. Inasmuch as new college technical recruits usually receive in excess of $600 a month, $500 seems a good lower limit. It is possible to have data points below this arbitrarily selected limit. For instance, a lady BS chemist returns to work in the technical library as a literature searcher after an absence of several years and is hired at, say, $400 a month.

For the upper limit $2000 a month will cover nearly all technical salaries. However the bright middle-aged chemist who discovers a new product that becomes the chief source of revenue for the chemical company may be compensated beyond that level.

Sometimes individual data points are encountered which fall outside the selected range of $500 to $2000. Obviously these points should be bypassed and not plotted.

Before plotting the percentiles we should bear in mind that the percentiles will be a pure mathematical fit. Typically, curves are steep in the early years, ascend less steeply, reach a plateau, and, in some cases, fall off slightly in the higher years. We have found that if the percentile curves start from year 0 the respective percentiles may be switched; that is, at year 1 or 2 the 10th percentile may be greater than the 90th percentile. This is difficult to explain to the personnel man who usually is not mathematically inclined. For this reason we start plotting the data at about the 5th or 6th year. Generally, we do not carry the data much beyond the 25th year.

We have found that not all years will have sufficient data for the 10th and 90th percentiles. Further the true mathematical curve for these percentiles, when plotted, frequently goes off the page. To avoid this we stop the plots of the 10th and 90th percentiles far short of the other percentiles.

Using this same background grid we have the option within the program of superimposing another set of percentiles which might represent another industry or as indicated on our title note a previous year.

The two sets of percentile curves may be used to compare salary schedules between companies, industries, or the increase of salaries from year to year. Another option within this program is to plot on the same grid the individual data points; that is, the individual salaries.

This serves several useful purposes. First, industry percentile curves may be plotted vs the individual company percentile curves followed by the individual salaries of all professionals within that company

From this configuration the analyst may observe where there is close agreement between industry and company schedules and additionally if this agreement is consistent over the entire range of ages.

Second, this type of plot enables management to observe the age distribution of employees.

Third, the detailed plotting of points enables the analyst to ascertain whether or not the percentile curves are unduly influenced by a particular age group. For instance, perhaps the company underwent a major expansion 25 years ago and hired more than the usual quota of starting young engineers. Without the individual plotted points this aberration would go unnoticed.

## The Future

Because of our computer configuration and core storage limitations the past salary surveys have been run as individual sub-programs requiring considerable manual card handling and machine sorting.

Each individual piece of data may be used several times, for instance, a BS may be included in each of the following groups:

Company - BS
BS - MS
BS - MS - PH.D.

Industry - BS
BS - MS
BS - MS - PH.D.

Total - BS
BS - MS
BS - MS - PH.D.

Now that our IBM 1620 Model II computer includes a disk drive we hope that we can eliminate much of the individual manual card handling, beyond the raw data stage.

Probably we will collect the raw data much the same as we have in the past, storing same on disks.

At the start of each group the group code number would be used to select the appropriate raw data and transfer it from disk to core storage.

For each group of data the five percentiles could be calculated in a manner similar to the present sub-programs. The output could be punched on cards primarily for checking purposes and also stored on another part of the disk.

A precautionary measure is to collect output regularly and periodically so that the operator may know where he is and, in the unfortunate happenstance of machine failure, all is not lost, but may be resumed from the point of last output. The percentile output could be retrieved from the disk storage to become input to the cubic finder sub-program. Several independent counters would have to be maintained as data are placed on or recovered from the disk. These counters become partial input to following programs, for instance, the weighting factor in cubic equation finder sub-program.

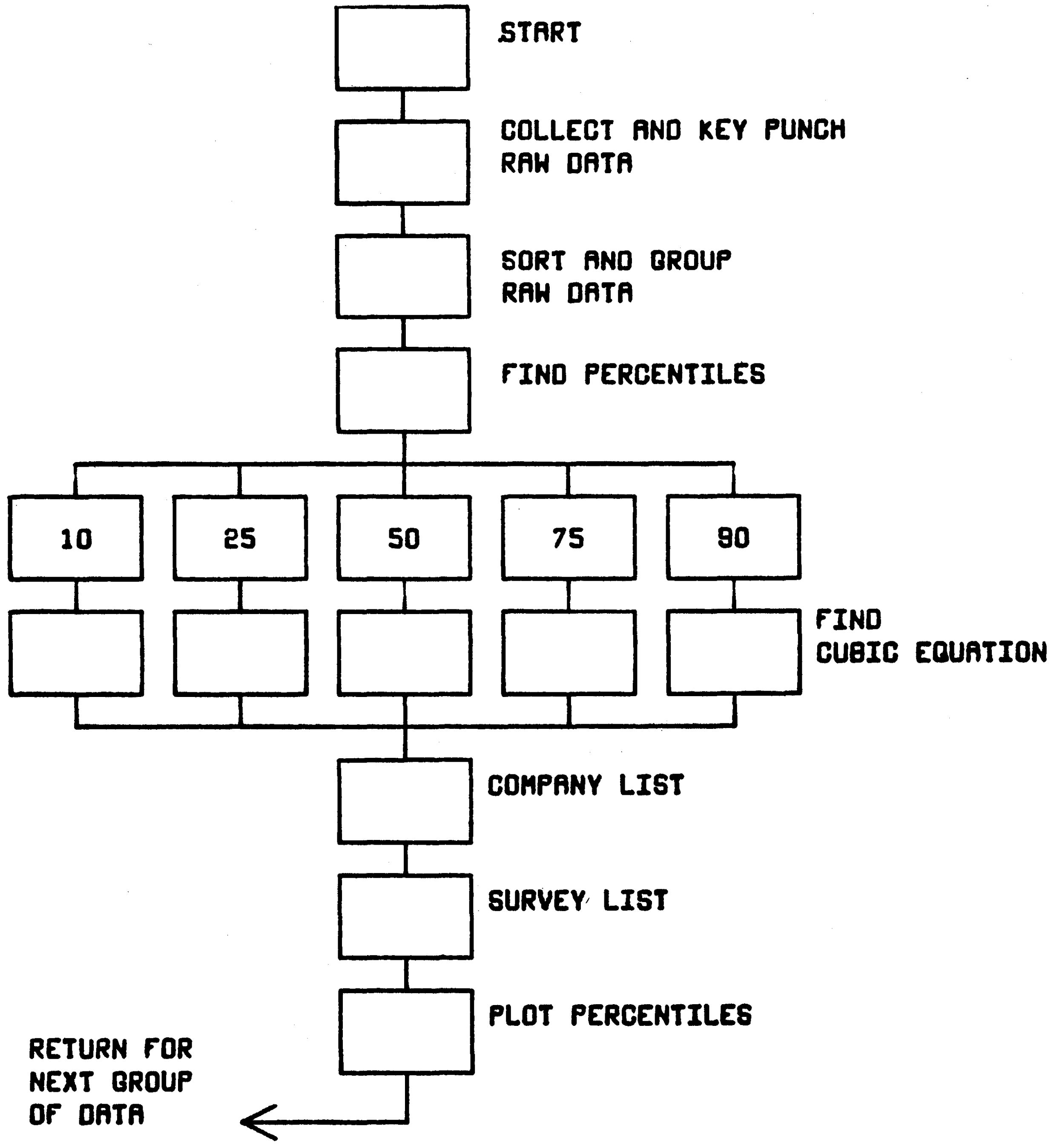
The Company List Program and the Survey List Program could be executed using as input data from earlier programs that had been stored on the disk. The plotting program may be run from data previously stored on the disk and in series with the other sub-programs.

Lapses in time between successive runs on the plotter may be sufficiently long to permit drying of the pen. In this case considerable time may be lost in priming the pen for each new chart.

As an alternative all the plotter sub-program input which is already on the disk could be punched on cards and set aside. Sometime later all the charts could be plotted, one directly after the other.

It is hoped that converting to a disk operation may reduce operating time for a large job from a number of weeks to a number of days.

Finally, we hope to expand our raw data to include additional parameters from which we may obtain more meaningful statistical information.



A COMPUTER SURVEY OF PROFESSIONAL SALARIES

WILLIAM J. ABNETT SEPTEMBER , 1865

FIGURE 1

A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETING
SEPTEMBER, 1965

## Card Format - Raw Data

Card Column

1 to 5 The group code number usually right justified. In these columns, two columns may be used to identify the effective date; that is, the year of the raw data. Two digits may be used to identify the company or group. Obviously, the code number may be used to sort and/or collate any combination of groups for further processing.

6 to 8 Blank

9, 10 The last two years of the year in which the individual records his first academic degree.

11 to 18 Blank

19 to 25 The monthly salary of the individual with the decimal point in card column 23.

26 to 29 Blank

30 The degree code:

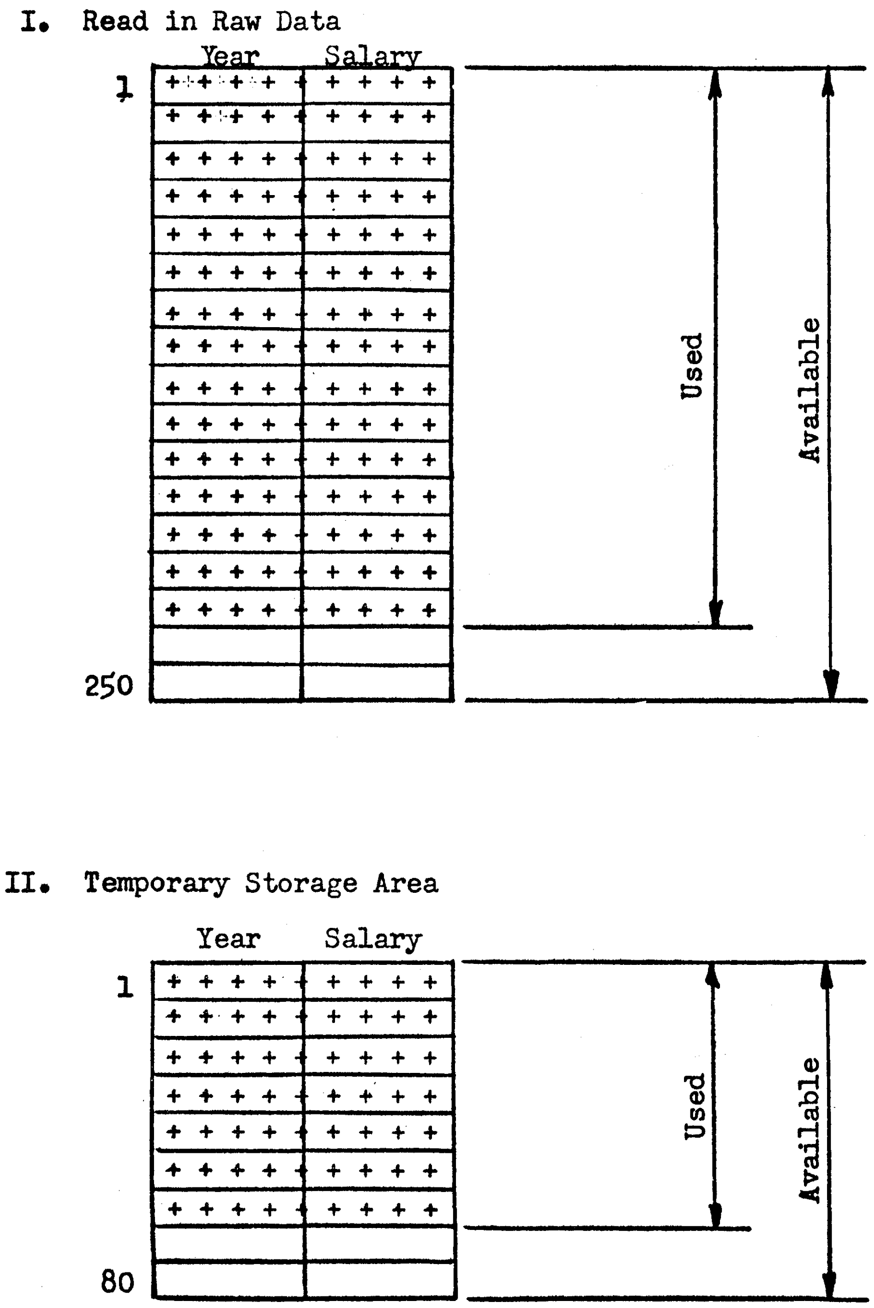
1. Bachelor
2. Master
3. Doctor

If the survey size is sufficiently large and a finer stratification or grouping of individuals is desired, card columns 28 and 29 may be used to note individual disciplines.

Figure 2

A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETING
SEPTEMBER, 1965

## Dimensioned Areas



I. Read in Raw Data

II. Temporary Storage Area

Figure 3

A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETING
SEPTEMBER, 1965

## Sorting Procedure in Temporary Storage Area

(Start)

| Step Block | I | II | III | IV |
|---|---|---|---|---|
| 1 | **400** | 300 | 300 | 300 |
| 2 | 500 | **500** | 400 | 400 |
| 3 | 600 | 600 | **600** | 500 |
| 4 | 700 | 700 | 700 | **700** |
| 5 | 800 | 800 | 800 | 800 |
| 6 | 300 | 400 | 500 | 600 |
| 7 | 1000 | 1000 | 1000 | 1000 |

(Finish)

| | I | II | III | IV |
|---|---|---|---|---|
| 1 | **300** | 300 | 300 | 300 |
| 2 | 500 | **400** | 400 | 400 |
| 3 | 600 | 600 | **500** | 500 |
| 4 | 700 | 700 | 700 | **600** |
| 5 | 800 | 800 | 800 | 800 |
| 6 | 400 | 500 | 600 | 700 |
| 7 | 1000 | 1000 | 1000 | 1000 |

| Step Block | V | (Start) VI | VII |
|---|---|---|---|
| 1 | 300 | 300 | 300 |
| 2 | 400 | 400 | 400 |
| 3 | 500 | 500 | 500 |
| 4 | 600 | 600 | 600 |
| 5 | **800** | 700 | 700 |
| 6 | 700 | **800** | 800 |
| 7 | 1000 | 1000 | **1000** |

| | (Finish) |
|---|---|
| 1 | 300 |
| 2 | 400 |
| 3 | 500 |
| 4 | 600 |
| 5 | **700** |
| 6 | 800 |
| 7 | 1000 |

Figure 4

CHART NO. 77

A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETING

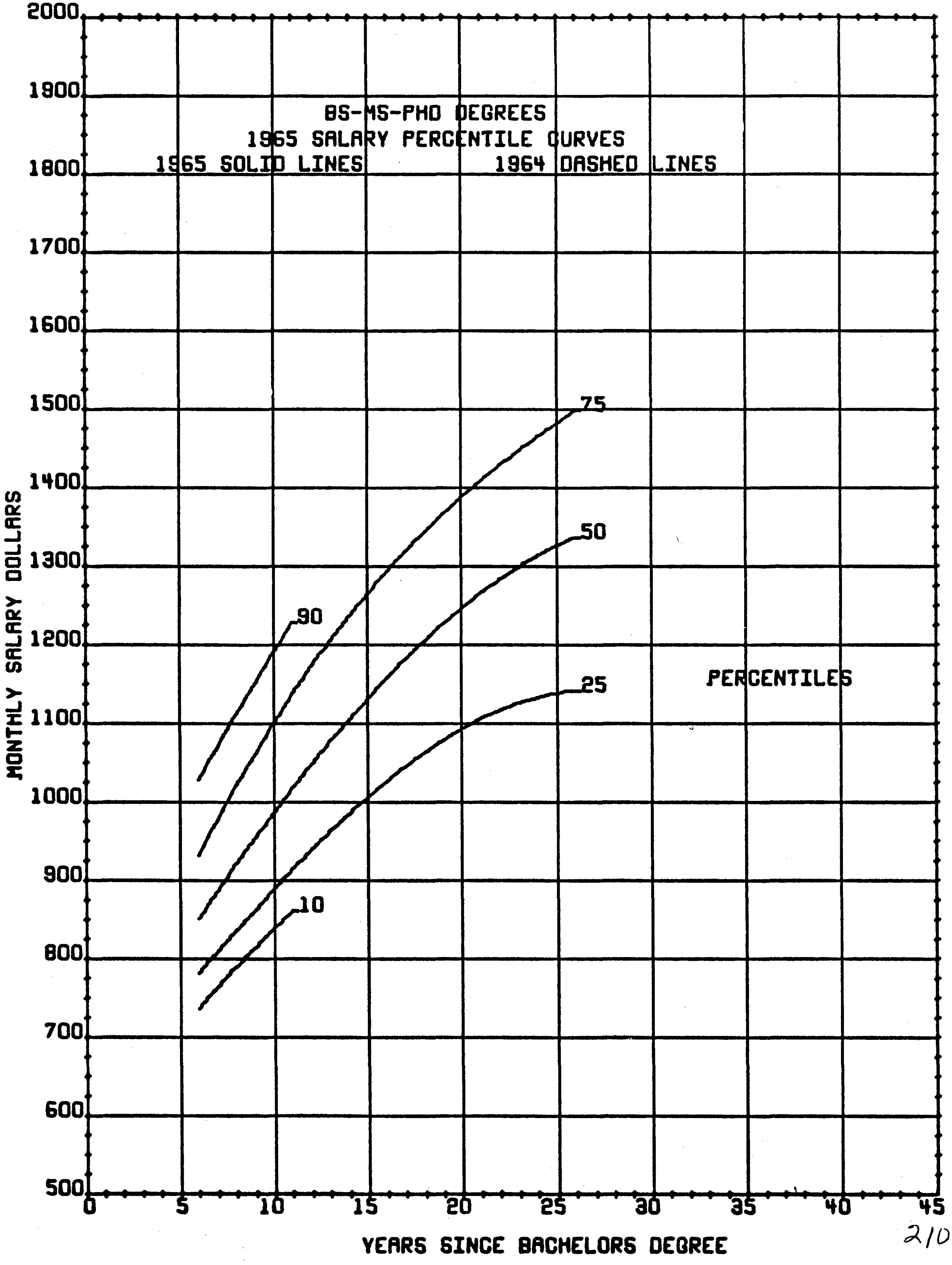


FIGURE 5

A COMPUTER SURVEY OF PROFESSIONAL SALARIES
1620 USERS GROUP MEETINGS

SEPTEMBER• 1965

TABLE-COMPANY SURVEY

ACTUAL PERCENTILE AND AVERAGE SALARIES
A COMPUTER SURVEY OF PROFESSIONAL SALARIES

| YEARS | | | | | | | |
|---|---|---|---|---|---|---|---|
| SINCE | NUMBER | | ACTUAL | PERCENTILES | | | AVERAGE |
| DEGREE | PERSONS | 10 | 25 | 50 | 75 | 90 | SALARY |
| 3•5 | 25 | 700 | 710 | 775 | 842 | 855 | 778 |
| 4•5 | 30 | | | | | | • |
| 5•5 | 46 | | | | | | • |
| 6•5 | 54 | | | | | | • |
| 7•5 | 58 | | | | | | |
| 8•5 | 59 | | | | | | |
| 9•5 | 69 | | | | | | |
| 10•5 | 78 | | | | | | |
| 11•5 | 75 | | | | | | |
| 12•5 | 75 | | | | | | |
| 13•5 | 77 | | | | | | |
| 14•5 | 76 | | | | | | |
| 15•5 | 70 | | | | | | |
| 16•5 | 59 | | | | | | |
| 17•5 | 50 | | | | | | |
| 18•5 | 44 | | | | | | |
| 19•5 | 37 | | | | | | |
| 20•5 | 33 | | | | | | |
| 21•5 | 27 | | | | | | |
| 22•5 | 29 | | | | | | |
| 23•5 | 27 | | | | | | |
| 24•5 | 24 | | | | | | |
| 25•5 | 25 | | | | | | |
| 26•5 | 22 | | | | | | |
| 27•5 | 22 | | | | | | |
| 28•5 | 18 | | | | | | |
| 29•5 | 19 | | | | | | |
| 30•5 | 21 | | | | | | |
| 31•5 | 21 | | | | | | |
| 32•5 | 22 | | | | | | |
| 33•5 | 20 | | | | | | |
| 34•5 | 19 | | | | | | • |
| 35•5 | 19 | | | | | | • |
| 36•5 | 18 | | | | | | • |
| 37•5 | 14 | 1045 | 1125 | 1362 | 1666 | 1717 | 1384 |

TABLE I

# A COMPUTER SURVEY OF PROFESSIONAL SALARIES
## 1620 USERS GROUP MEETINGS

SEPTEMBER, 1965

TABLE-SURVEY LIST

### ACTUAL AND LEAST SQUARES PERCENTILE SALARIES
### A COMPUTER SURVEY OF PROFESSIONAL SALARIES

| YEARS SINCE DEGREE | NUMBER PERSONS | ACTUAL PERCENTILES | | | | | LEAST SQUARE PERCENTILES | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10 | 25 | 50 | 75 | 90 | 10 | 25 | 50 | 75 | 90 |
| 3.5 | 25 | 700 | 710 | 775 | 842 | 855 | 665 | 712 | 760 | 805 | 917 |
| 4.5 | 30 | | | | | | | | | | • |
| 5.5 | 46 | | | | | | | | | | • |
| 6.5 | 54 | | | | | | | | | | • |
| 7.5 | 58 | | | | | | | | | | |
| 8.5 | 59 | | | | | | | | | | |
| 9.5 | 69 | | | | | | | | | | |
| 10.5 | 78 | | | | | | | | | | |
| 11.5 | 75 | | | | | | | | | | |
| 12.5 | 75 | | | | | | | | | | |
| 13.5 | 77 | | | | | | | | | | |
| 14.5 | 76 | | | | | | | | | | |
| 15.5 | 70 | | | | | | | | | | |
| 16.5 | 59 | | | | | | | | | | |
| 17.5 | 50 | | | | | | | | | | |
| 18.5 | 44 | | | | | | | | | | |
| 19.5 | 37 | | | | | | | | | | |
| 20.5 | 33 | | | | | | | | | | |
| 21.5 | 27 | | | | | | | | | | |
| 22.5 | 29 | | | | | | | | | | |
| 23.5 | 27 | | | | | | | | | | |
| 24.5 | 24 | | | | | | | | | | |
| 25.5 | 25 | | | | | | | | | | |
| 26.5 | 22 | | | | | | | | | | |
| 27.5 | 22 | | | | | | | | | | |
| 28.5 | 18 | | | | | | | | | | |
| 29.5 | 19 | | | | | | | | | | |
| 30.5 | 21 | | | | | | | | | | |
| 31.5 | 21 | | | | | | | | | | |
| 32.5 | 22 | | | | | | | | | | |
| 33.5 | 20 | | | | | | | | | | • |
| 34.5 | 19 | | | | | | | | | | • |
| 35.5 | 19 | | | | | | | | | | • |
| 36.5 | 18 | 953 | 1060 | 1425 | 1695 | 1705 | 917 | 1026 | 1344 | 1658 | 1697 |

TABLE II

# INTERACTION IN 2-WAY ANALYSIS OF VARIANCE WITH SINGLE REPLICATION

Carol, B. and DeLegall, W.
Computer Center
New York Medical College

The availability of standard statistical packages makes it possible for the user, particularly the layman, to process data with a seeming ability to extract desired statistical information from the computer. This procedure commendable as it is in popularizing the role of mathematics and probability in the evaluation of data is frought with danger owing to the lack of sophistication on the part of the user. It is not enough to ask for an analysis of variance with as many factors as there are in the design. Other questions must be asked such as, is the data nonorthogonal? Is it a fixed or a random design? Is it mixed? Are factors nested in other ones? Is it partially nested and crossed? What assumption can be made about the data? Can interactions be assumed to be a certain function of main effects? Answers to these questions distinguish the professional statistician from the lay user.

In response to the needs of expert handling of such problems, various supplementary programs are required to support the main body of packaged procedures available to the user. An example of this is a program that can test for interactions in a 2-way design with single replication with the assumption that interactions can be expressed as quadratic functions of main effects. This assumption is frequently reasonable.

We feel a full complement of sucn programs would round out the professional use of computers for analysis of variance as well as other branches of mathematical statistics.

Attached is a description of a Fortran program to make this test. The documentation will be submitted to the 1620 Users Group Library.

TITLE: Sum of Squares Due to Interactions Restricted to Quadratic Functions of 2 Main Effects, Single Replication.

SUBJECT CLASSIFICATION: Statistical

AUTHOR: Bernard Carol and Walter DeLegall

ORGANIZATION: Computer Center, New York Medical College

DATE: April 19, 1965

USERS GROUP MEMBERSHIP CODE: 1359

DIRECT INQUIRIES TO: Walter DeLegall

DESCRIPTION/ PURPOSE: This program can be very useful in those cases in a 2-way design with single replication where one wishes to test for interactions assuming that interactions can be expressed as quadratic functions of main effects.

MATHE. METHOD: The test is:

$$F=\frac{(N-I-J)\,SS_G}{SS_{res}}$$

Where N=total no. of observations
I=no. of levels in first factor
J=no. of levels in second factor

$$SS_G = \frac{\left(\sum_i \sum_j \hat{\alpha}_i \hat{\beta}_j Y_{ij}\right)^2}{SS \sum \hat{\alpha}_i^2 \sum \hat{\beta}_j^2}$$

$$SS_{Res} = SS_{int} - SS_G$$

$$SS_{int} = \sum_i \sum_j (Y_{ij} - Y_{i.} - Y_{.j} + Y_{..})^2$$

$$\hat{\alpha}_i = Y_{i.} - \bar{Y}$$

$$\hat{\beta} = Y_{.i} - \bar{Y}$$

$Y_{i.}$=mean of $i^{th}$ level of first factor
$Y_{.j}$=mean of $j^{th}$ level of second factor
$Y_{ij}$=observation in cell of $i^{th}$ level of first factor
and the $j^{th}$ of the second factor.

This statistic is compared to $F_{1,N-I-J}$. $SS_G$ is calculated on the computer..

$SS_G$ is the sum of squares due to the hypothesis that interactions are zero subject to the restriction of being quadratic functions of main effects. The computer calculates $SS_G$.

| | |
|---|---|
| RESTRICTIONS/<br>RANGE: | N/A |
| SOURCE LANGUAGE: | Fortran |
| MACHINE<br>CONFIGURATION: | 1620 Mod. I 20K memory, no special features |
| EXECUTION TIME: | Unavailable. |

PROGRAM OPERATION:

The program was written to utilize the punched output of means from the Harkins Analysis of Variance Program 6.0.014. However, any program which produces the corrected sums of squares for the first and second factors, the grand mean, the mean of the $i^{th}$ level of the first factor and of the $j^{th}$ level of the second factor and the value of the observation in the cell of the $i^{th}$ level of the first factor and the $j^{th}$ level of the second factor for all i and j can be used to supply the input data.

The computer center program requires that the values for the grand mean, $Y_{\cdot j}$, and $Y_{i\cdot}$ (after sorting on subscript from right to left) be read in from cards along with the corrected sums of squares. The output is printed on typewriter and is self-explanatory. The last printed line is the sum of squares of interaction restricted to be quadratic functions of main effects.

BIBLIOGRAPHY:

Scheffė, H. Analysis of Variance, Wiley, N.Y., 1958. Ch. 4.

# DECTRAN

## A Decision Table Language Translator

Presented By

Robert P. Bair
Elliott Company
Division of Carrier Corporation
Jeannette, Pennsylvania

1620 Users Group

Eastern and Midwestern Joint Meeting

Americana Hotel

New York City

October 7, 1965

DECTRAN is an acronym for DECision table TRANslator. It presents a new algorithm for decoding a decision table language into FORTRAN and extends the instruction repertory of the table-oriented language. Writing a group of decisions as a decision table is about as easy as creating a flow chart for the same problem. Any similarities or differences between the operations for different conditions can be easily seen due to the parallel structure of a table. Once it is formed,a decision table may be understood by your boss, your wife, and the crowd at the lunch table. A decision table program is also easier to revise, and document.

However,tables are two-dimensional, and the FORTRAN output is one-dimensional with all statements in a single line. Thus, the main function of DECTRAN is reducing a table to a single list of instructions.

The advantages of decision tables may be seen by considering the example in Figure 1. Here, in table form, are the necessary decisions for the difficult task of distinguishing between elephants, giraffes, and women. Before I explain this tricky problem, let me point out the form of the table. It is divided horizontally into two sections - the conditions and the actions. Each of these are divided into a stub and entries. The entries are subdivided into rules, of which this table has three. The function of the condition statements is to choose the first rule in which all of the entries are satisfied. The action statements then perform only those operations which are specified for that rule.

Any blank entries under a rule for some condition are considered indifferent, or "don't care", and the rule is determined without testing that condition. Thus, rule 3 can be satisfied regardless of the length of the nose.

Three different types of conditional statements are shown in the example. In the second conditional statement, both quantities to be considered and the relations between them are all given in the stub. The entries indicate that for rule 1 the relation must be true (T or Y) and that for rule 2 the relations must be false (F or N). Rule 3 is indifferent. This type of statement is called <u>limited - entry</u> because the entries are limited to the three possible values: Yes or True, No or False, and Indifferent.

The other two conditional statements in the example are in the extended - entry form. The quantities or relations change from one rule to the next and the values desired for each rule are simply used as entries. In the first conditional statement, the relation between the first quantity and the second is fixed, but the second quantity changes from rule to rule. In the third conditional statement, the only thing in common between the rules is the first quantity, which must always be in the stub. Its relation to some other quantity is specified in each entry. In either case, if the relation is true, the entry involved is satisfied.

All the entries in one rule must be satisfied for that rule to be chosen. In case of a tie when two rules fit simultaneously, the one closer to the stub wins. It is assumed that the rules are written in decreasing frequency of occurance from rule 1 to the right. The order of conditions is unimportant.

The example assumes that out of the entire kingdom, the only remaining choices for the unknown animal species are giraffe, elephant, and woman. A much larger table would be required to pick these three out of all the possible animals. Each rule in the entries of the conditional statements contains the criteria for one animal. Following the first rule, if the animal has 4 legs, and a nose over 36 inches long, and a neck not longer than 40 inches, it must be an elephant. On the other hand, if it has the same number of legs, but its nose is not longer than 36 inches and its neck is over 40, it is labeled a giraffe. But if it has only 2 legs, it must be a woman. This definition is very liberal, but it is sufficient when one considers the opposition.

Once a rule has been chosen, all the actions specified for that rule are executed in order. Only three actions are given here, but that number could be increased and many other types of operations may be performed.

Having satisfied ourselves that the decisions are sufficient, the table must now be coded for DECTRAN. Several decisions must be made about the form of the statement. Separation of the stub from the entries may be done in two ways - either by starting the entries in a special location on the card, such as card column 35, or by putting a special character between the two parts. In order to make the input format as flexible as possible, I have chosen to use the latter method and insert a dollar sign ($) at the end of the stub. The entries could be separated from each other by a specified number of columns, but again, in the interest of flexibility, I have chosen to use a comma to allow for different length entries.

Condition statements can be distinguished from action statements by specifying that they must start with the key work "IF". Replacement statements could also be identified by a starting key word, but this is unnecessary since its unique equal sign serves this purpose.

The mathematical symbols for less than (<) and greater than (>) will have to be replaced since they are not included in the FORTRAN character set. The symbols to be used for these relational operators are shown in Figure 2. The equal sign must also be replaced by a symbol so that it may be reserved for replacement statements. After variable names have been assigned to the quantities involved, our original table appears as in Figure 3. Figure 4 is the DECTRAN coding form on which the statements may be coded.

Once the conditional statements are all together in a block, they must be examined and translated into the most efficient sequence of FORTRAN IF statements. First, all decisions must be reduced to the binary True - False form of the limited entry statement, which involves simplifying all of the extended-entry statements. Figure 5 shows how this is done by making a new statement out of the stub and each <u>different</u> entry. The new entries are formed so that any Y result determines a rule, and N results continue the testing.

In this statement, as in most decision tables, it is possible for the quantities to take values for which none of the entries are satisfied. If this happens, DECTRAN prints a message and branches to MONITOR. To form a rule which is satisfied only when all the others are not, an "else" entry (which consists of an at sign (@) so as to not be confused with variables) may be used. This rule must be the last rule in the statement so that it follows the testing of all the previous entries. Figure 6 shows that an "else" entry causes a rule of all "N" entries to be formed in the limited-entry equivalent. Notice also that an indifferent entry is carried through the equivalent. An else entry is only to be used with extended-entry conditional statements. In limited-entry tables, it is just as easy and more logical to write N instead.

The example, now converted to all limited-entry conditional statements, appears in Figure 7. There are many different sequences of instructions which would make these decisions, and some require more branch points or compares than others. The problem then, is to determine a sequence of instructions with the fewest number of branch points, and develop some notation to express this sequence so that it may be stored. The entries from the conditional statements are considered by themselves, and Figure 8 shows how these entries are stored in memory in a matrix.

Two additional areas in memory are required. One is a mask which has a number of digits equal to the number of conditions. The second is a copy of the entry table which is modified by the mask. Initially the mask is set to all zeros. It is compared digit by digit with one of the rules in the original table, and the copy entry table is formed. See Figure 9 for the digit that is placed in the copy as a function of the digits in the mask and original. If a conflict is encountered, the rule in process in the copy is marked "non-existent" and it goes on to the next rule.

When the table is finished, the copied table is examined and the "best" condition to be used in the next compare is found. This condition is determined by using the following three steps:

Step 1. Choose that condition with the least number of indifferent entries.

Step 2. If step 1 results in a tie, substitute

0 for I

-1 for N

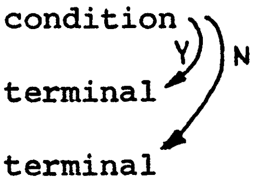
1 for Y

and add each row. Choose the condition whose sum has the highest absolute value.

Step 3. If rule 2 results in a tie, choose the first condition among those tied.

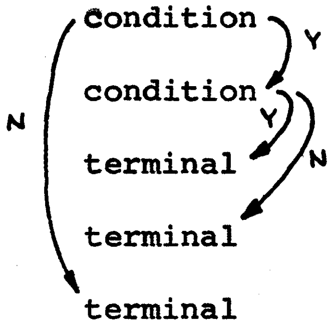
Looking at Figure 8, conditions 1 and 4 are tied at 0 after step 1. They are still tied at |+1| and |-1| after step 2, so step 3 chooses condition 1.

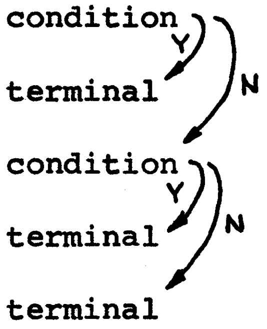
The flow chart in Figure 10 shows the complete process of generating the notation. The simplest notation is for one conditional statement and is in the following form:



The two terminal points indicate rule numbers or an error code in the event that some combination of conditions is not given in the rules. The first terminal is executed if the condition is Yes, the second if it is No.

For two conditions, the notation may be in one of two possible forms:





As the notation is built, the mask is changed, which produces changes in the copied entry table. Whenever any rule of that table becomes all zero, no further tests are needed and that rule is chosen by entering its number into the notation. Then the notation is scanned backwards to find the first condition which has not had its No result investigated. When no rule fits some combination of digits in the mask the copied table is non-existent and an error code is put into the table. If alternatives still exist, the three steps for determining the next condition are applied, and with the mask set to 1 in the digit opposite that condition, the copied entry table is re-formed.

When the notation is finally complete, it may be easily converted to a flow chart. Starting with the first condition in the notation, every condition number is a test statement which must have two different exits. Trace the Yes branches until a terminal (rule number or error) is encountered, then back up, make that condition negative, and start again. Figure 11 is the completed notation and flow chart for the example problem. By starting at $C_1$ in the flow chart and tracing all the Y paths, one may work backwards and get the notation directly.

Dectran has previously stored each of the conditions on disk so that it can read and punch them in the order of the notation. When this is completed, the action statements for the table are read.

Now that we know how to separate the women from the animals, we may write actions suitable for each case. Action statements may also be in either extended or limited entry form, and the three actions given in the example are all extended-entry because part of the replacement statement is in the entries. In a limited-entry action statement the stub contains a complete statement and the entries indicate by an "X" those rules for which it is to be done. The only possible entries are X or blank indicating do this action or do not do this action for this rule, respectively.

Each action is read, converted to limited-entry form if necessary, and stored on the disk. A table of entries is built as it was for the conditional statements. When the first statement of the next table is read, the present table must be complete and the decoding of the action statements can begin. The statement number of the first instruction produced for each rule is the table number plus the rule number. In table 126, the statement number for rule 8 is 12608. Each rule of the entries table is scanned and wherever an X appears, the corresponding action is read from the disk and punched. Most actions require no further translation; however, a few are new to FORTRAN and must be converted.

All the FORTRAN statement numbers required for the output are made up of the 3-digit table number and a 2-digit sequence number. The first statement generated for table nnn will be numbered nnnoo. A branch to any table, for example GO TO TABLE 34, may be translated immediately into GO TO 03400. Table numbers 997 through 999 are reserved for internal use to preceed FORMAT statement numbers and to enable some statements to be coded out-of-line.

To make the language as useful as possible, an iteration statement was desired that is compatible with the table structure. Since the DECTRAN-to-FORTRAN translation is one pass, it was necessary to invent a statement which could be compiled as it was read. The result is the following statement:

LOOP n, $i=e_1$, $e_2$ .r. $e_3$, $e_4$

n is the table to be executed after the loop is satisfied. i is the index which starts at a value of $e_1$ and is incremented by $e_4$. $e_1$ through $e_4$ are arithmetic expressions such that i and $e_4$, and $e_2$ and $e_3$ are of the same mode and .r. is a relational operator. The loop will be continued until $e_2$ .r. $e_3$ is true. As long as this terminating condition is false, the next statement is executed. The range of the loop is defined by the following statement:

END LOOP m

where m is the table number in which the loop statement appears. If we now specify that a loop statement must be the first statement in a table, all locations are known as soon as any loop statement is read. Figure 12 shows how a loop statement is coded into FORTRAN. Notice that any number of end loop statements could be used, and that they may appear anywhere in the program in relation to the loop statement. Since the terminating condition is tested before the execution of the loop, it is possible to do the loop zero times, an option not available in FORTRAN. Because a LOOP statement is not compiled into a FORTRAN DO loop, none of the DO's restrictions need be enforced. A branch into the range of a loop may be made if the index and all the parameters are defined. Nesting can become as complicated as desired and one loop may even end within the range of another.

Subroutining is a powerful programming technique that enables us to branch out of a sequence of statements, execute another set of instructions, and return. A much more intimite relationship between the two parts could be achieved if the subroutine where included in the compilation of the mainline program which calls it. Then the subroutine could on occasion branch to specific points in the mainline program instead of returning to the next instruction. Variables and working storage could be shared more easily, and short sequences of instructions with many parameters could make practical subroutines.

Since DECTRAN allows the familiar FORTRAN subroutines and functions, the word "subroutine" will be reserved for reference to that FORTRAN statement. The type of internal subroutine described above will be called an "internal procedure", referring to some set of tables within the program itself.

A procedure is called by the following statement.

```
DO TABLE n
```

where n is the number of the first table in the procedure and must be ≤ 99. The return statement has the following form:

```
RETURN n
```

where the number of the first table in the procedure is used again to differentiate between different procedures. The number of a procedure's first table is analogous to a subroutine's name. Since the return statement is identified, two procedures may be merged to share common statements, or one procedure may be entirely contained within another, and both still return to the proper point. The number of return statements is not limited.

A return from an internal procedure is an exception to the table concept because it branches to a statement within a table rather than to the beginning of the table. This is necessary to return to the next statement following the DO TABLE statement. This next instruction is given a number and that number is stored on the disk in a list for the table called. An internal variable of the form LLLPn is set equal to the number of times this procedure has been called so far. The return is accomplished by a computed GO TO instruction using the variable LLLPn and the list of statement numbers for procedure n that is stored on the disk. The actual computed GO TO statement is saved for the end of the FORTRAN program so that all of the references to the procedure will have been found by the time that it is coded. It is ~~a~~ given a statement number 998n. The RETURN n statement is then actually compiled into GO TO 998n. The return could be greatly simplified if a FORTRAN compiler with an ASSIGN statement, such as KINGSTON FORTRAN II were to be used, or if DECTRAN were to compile directly into machine language.

If a branch is made to an internal procedure or to one of the tables within it by a GO TO TABLE command, and a RETURN statement is subsequently executed, it will return to the same point as it did the last time it was executed, since the variable LLLPnn can be changed only by a DO TABLE command. If this procedure has not yet been referenced by a DO TABLE command the variable will be undefined and trouble will result.

Another new statement, a reverse replacement statement with the keyword MOVE, is allowed because it fits the table structure so nicely. The statement

```
MOVE B * * 2 $ Z , Y
```

is equivalent to two ordinary arithmetic statements:

```
Z = B * * 2  $  X

Y = B * * 2  $  , X
```

One other statement is different from its FORTRAN form. In the FORMAT statement, the format number has been moved so that it can not be confused with a table number.

FORMAT n $(s_1, \ldots s_n)$

n is the repositioned number, which must be ≤ 99. It is preceeded by the number 997 as the statement is rearranged for FORTRAN.

Finally, our original example compiled into FORTRAN is shown in Figure 13. A block of IF statements, in the same order as that described by the finished notation, is first. There are some possible combinations of the conditions that the rules do not allow, so an error trap is provided. Next comes the three actions for each rule. Note the assignment of statement numbers which assumes that the example was table number 25.

The advantages of writing decisions in table form can not be refuted. It is easier to understand, learn, write, and modify decision tables than the normal single-line-of-instructions program. As the number of decisions increases, and the flow chart becomes more unmanageable, the decision table is still clear and precise in its meaning. Many applications, especially in engineering, are so complicated as to make direct FORTRAN coding impractical. Now that a compiler is available to decode decision tables as well as a complete high-powered language to go with them, any program may take advantage of the use of decision tables.

Appendix A, which follows, contains a formalization of the language specifications of DECTRAN. Appendix B contains a description written by Mr. John Moschetti, of a specific engineering applications problem where DECTRAN is being used.

# DECTRAN EXAMPLE

| | STUB | ENTRIES | | |
|---|---|---|---|---|
| CONDITIONS | IF THE NO OF LEGS = | 4 | 4 | 2 |
| | IF NOSE LENGTH > 36 | Y | N | |
| | IF NECK LENGTH | ≤ 40 | > 40 | |
| ACTIONS | ANIMAL IS A | ELEPHANT | GIRAFFE | WOMAN |
| | FEED IT | PEANUTS | HAY | MARTINIS |
| | IT LIVES IN | INDIA | AFRICA | APARTMENT |
| | | (rule 1) | (rule 2) | (rule 3) |

Fig 1

# RELATIONAL OPERATORS

| | |
|---|---|
| .L. | LESS THAN |
| .LE. | LESS THAN OR EQUAL TO |
| .E. | EQUAL TO |
| .NE. | NOT EQUAL TO |
| .GE. | GREATER THAN OR EQUAL TO |
| .G. | GREATER THAN |

Fig 2

## DECTRAN EXAMPLE
## CODING

```
IF NUMLEG .E.      $   4    ,    4    ,    2
IF LNOSE .G. 36    $   Y    ,    N    ,
IF LNECK           $ .LE.40, .G. 40,
ANIMAL =           $ ELEPHT, GIRAFF, WOMAN
FOOD   =           $ PEANUT,  HAY   , MARTNI
HOME   =           $  INDIA, AFRICA,  PAD
```

Fig 3

## DECTRAN EXAMPLE
## LIMITED-ENTRY FORM OF CONDITIONS

```
IF NUMLEG .E. 4     $   Y,   Y,   N
IF NUMLEG .E. 2     $   I,   I,   Y
IF LNOSE .G. 36     $   Y,   N,
IF LNECK .LE. 40    $   Y,   N,
```

Fig 7

Elliott
A DIVISION OF
CARRIER
CORPORATION

# DECTRAN DECISION TABLE TRANSLATOR

## CODING FORM

Program: ______________________________ Page No. ______ of ______

Programmer: ____________________ Date: ____________________

| TABLE |  | STATEMENT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 3 | 4 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 |

Fig 4

```
IF  A = $ B, C, D, E
```

May be written as

```
IF  A = B $ Y, N, N, N
IF  A = C $ I, Y, N, N
IF  A = D $ I, I, Y, N
IF  A = E $ I, I, I, Y
```

Fig 5

```
IF  A = $ B,  , D, @
```

May be written as

```
IF  A = B $ Y, I, N, N
IF  A = D $ I, I, Y, N
```

Fig 6

# DECTRAN EXAMPLE
# ENTRY TABLE

|  | R1 | R2 | R3 |
|---|---|---|---|
| C1 | Y | Y | N |
| C2 | I | I | Y |
| C3 | Y | N | I |
| C4 | Y | N | I |

Entries of conditional statements

|  | R1 | R2 | R3 |
|---|---|---|---|
| | 1 | 1 | $\bar{1}$ |
| | 0 | 0 | 1 |
| | 1 | $\bar{1}$ | 0 |
| | 1 | $\bar{1}$ | 0 |

Matrix stored in core

I=0, N=$\bar{1}$, Y=1

Fig 8

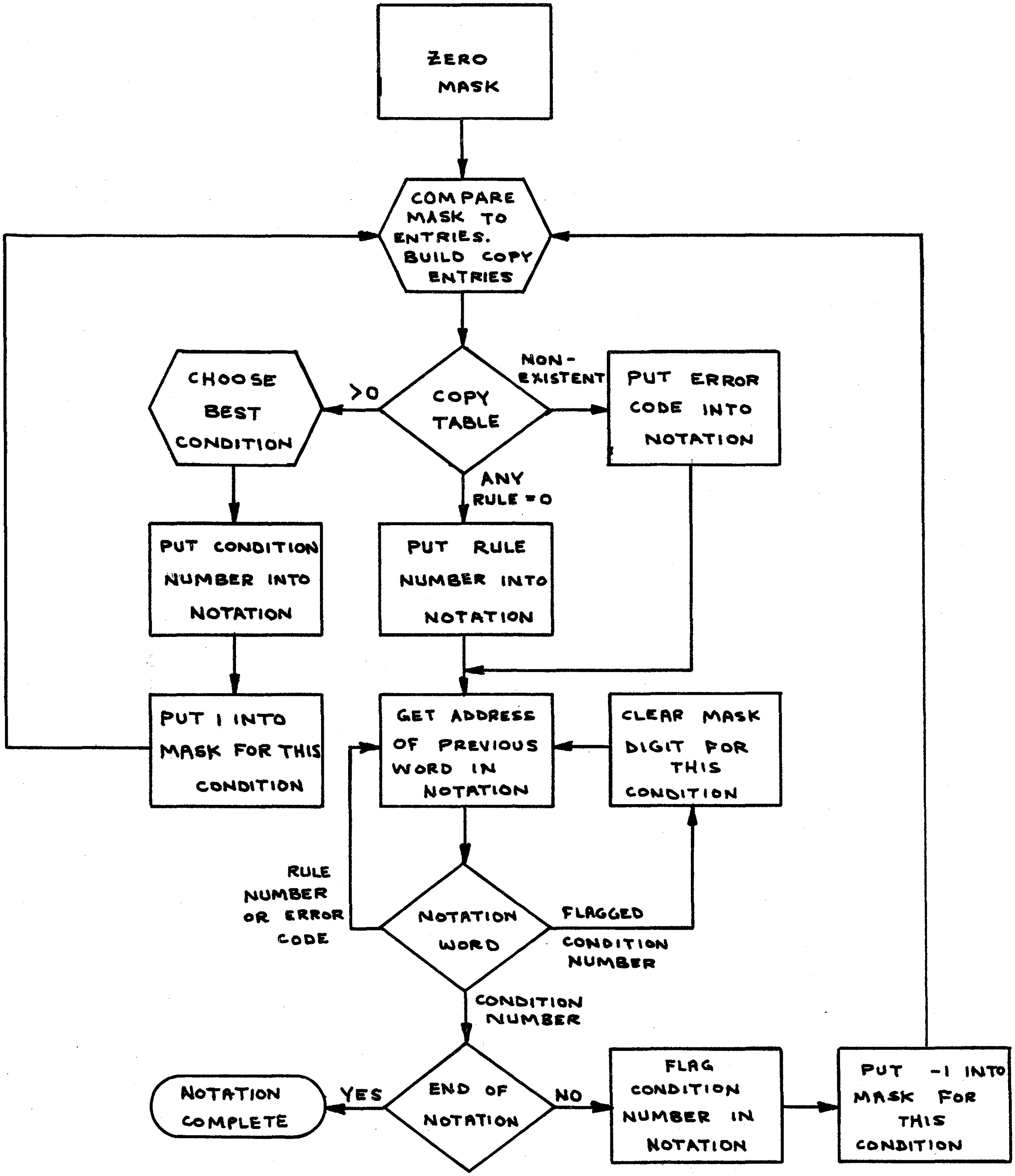
# GENERATION OF COPIED ENTRY TABLE

| digit in mask | digit in entry table | resulting digit in copy |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | $\bar{1}$ | $\bar{1}$ |
| 1 | 0 | 0 |
| 1 | 1 | 0 |
| 1 | $\bar{1}$ | X |
| $\bar{1}$ | 0 | 0 |
| $\bar{1}$ | 1 | X |
| $\bar{1}$ | $\bar{1}$ | 0 |

X indicates a conflict

Fig 9

# BUILDING NOTATION



ZERO MASK
COMPARE MASK TO ENTRIES. BUILD COPY ENTRIES
CHOOSE BEST CONDITION
>0
COPY TABLE
NON-EXISTENT
PUT ERROR CODE INTO NOTATION
ANY RULE = 0
PUT CONDITION NUMBER INTO NOTATION
PUT RULE NUMBER INTO NOTATION
PUT 1 INTO MASK FOR THIS CONDITION
GET ADDRESS OF PREVIOUS WORD IN NOTATION
CLEAR MASK DIGIT FOR THIS CONDITION
RULE NUMBER OR ERROR CODE
NOTATION WORD
FLAGGED CONDITION NUMBER
CONDITION NUMBER
NOTATION COMPLETE
YES
END OF NOTATION
NO
FLAG CONDITION NUMBER IN NOTATION
PUT -1 INTO MASK FOR THIS CONDITION

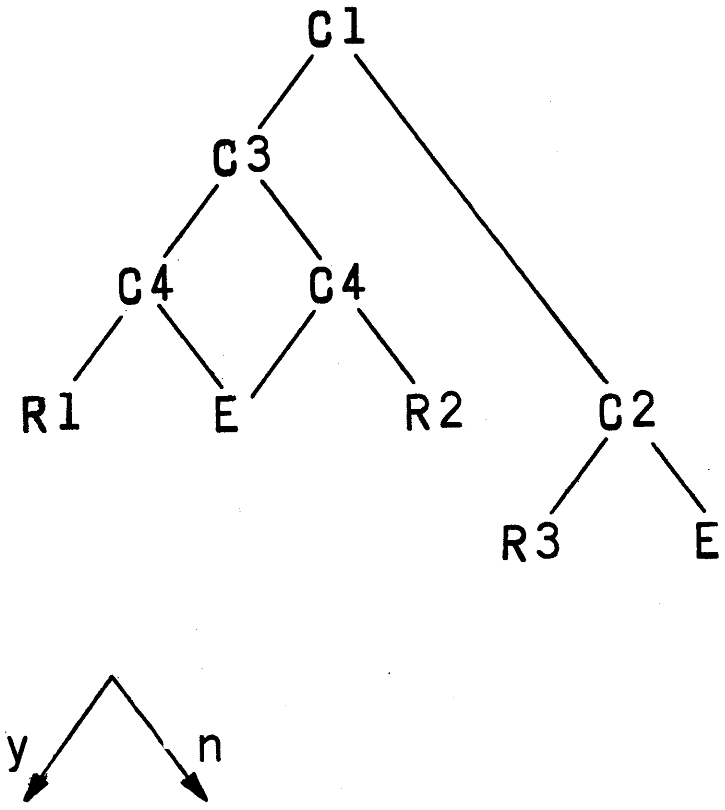
Fig 10

# DECTRAN EXAMPLE

## NOTATION

C1
C3
C4
R1
E
C4
E
R2
C2
R3
E

## FLOWCHART



E = error

Fig 11

# DECTRAN COMPILATION

LOOP n, $i=e_1$, $e_2$ .r. $e_3$, $e_4$

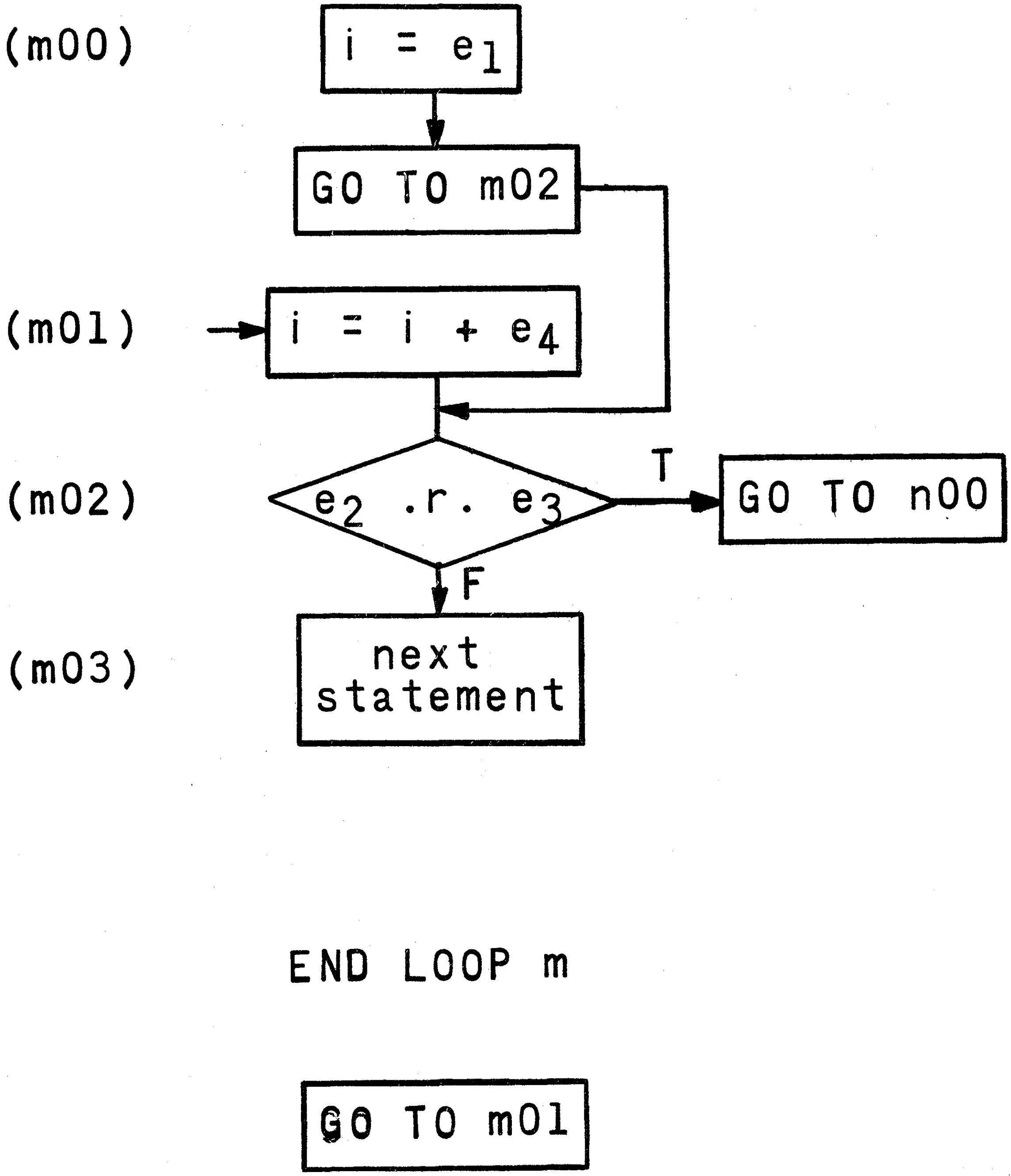


Fig 12

# DECTRAN EXAMPLE
# FORTRAN OUTPUT

```
02500 IF(NUMLEG-4)02507,02504,02507
02504 IF(LNOSE-36)02506,02506,02505
02505 IF(LNECK-40)02501,02501,02599
02506 IF(LNECK-40)02599,02599,02502
02507 IF(NUMLEG-2)02599,02503,02599
02599 LLLT=025
      GO TO 99999
02501 ANIMAL=ELEPHT
      FOOD=PEANUT
      HOME=INDIA
      GO TO 02600
02502 ANIMAL=GIRAFF
      FOOD=HAY
      HOME=AFRICA
      GO TO 02600
02503 ANIMAL=WOMAN
      FOOD=MARTNI
      HOME=PAD
```

(Table no. is assumed to be 025)

Fig 13

# APPENDIX A

## FORMAT OF DECTRAN STATEMENTS

A DECTRAN coding form appears in Fig. 4.

The first 3 columns contain the table number (unsigned integer from 0 through 996.)
Column 4 is the card code
    * indicates a comment card
    any other character except blank, + , - , or zero indicates a continuation of the last statement.
Columns 5 through 75 contain the statement
Columns 76 through 80 are ignored and may be used for identification.

The number of continuation cards is not limited, except for the following restriction: the length of the stub plus the longest entry of a statement must not exceed 71 columns.

A single dollar sign separates the stub of a statement from its entries. It may appear anywhere, but it may be convenient to place it in column 35.

Each entry is separated from the next by a comma. A comma may or may not follow the last entry.

Blanks are optional anywhere in the statement and may be inserted where desired for clarity in reading.

The last card of the program must be an end card containing END in columns 1 through 3 and blanks in columns 4 through 75.

## TABLE

A table is a group of related statements that may only be entered at one point and may not contain more than one block of conditional statements.

All statements in a table share the same table number, which is written to the left of the table's first statement. This is the only possible entry point to the table.

A table may start with either action or condition statements. If action statements are first, they may not have any entries. If no condition statements follow, it is an "unconditional table".

If a loop statement is used, it must be the first statement in the table, and no more than one such statement per table is allowed.

All of the condition statements in a table must appear together in a block.

All action statements which follow the last condition statement will be done only for the rules indicated in their entries.

No exit need be specified for an unconditional table. If omitted, the next table in the order entered will be executed. All other tables must specify an exit for every rule.

The number of rules and conditions in a table are limited by the following restriction: The FORTRAN translation of one table may not require more than 99 statement numbers. The maximum number of numbers required for a table of R rules and c conditions is R(c+1) + 1. (Add 3 if the table starts with a LOOP statement.) The actual number required will depend on the amount of similarity in the decisions, and will generally be much less than the maximum.

## CONDITION STATEMENTS

Three forms are permitted, depending on the division between stub and entries.

1) IF $e_1$ .r. $e_2$ \$

2) IF $e_1$ .r. \$ $e_2$

3) IF $e_1$ \$ .r. $e_2$

where $e_1$ and $e_2$ are arithmetic expressions of the same mode, and .r. is a relational operator.

In the first form, called limited entry, the condition is complete in the stub and is answered either yes or no by using the following entries.

| | |
|---|---|
| Y or T | Yes |
| N or F | No |
| I or blank | indifferent |

The other two forms are called extended entry and use part of the condition for the entry. For those rules which do not require testing this condition, the entries are blank. Indifference can also be indicated by an I entry for type 3, but is not allowed in type 2 where it would be taken as a variable.

The relational operator may be any one of the following:

| | |
|---|---|
| .L. | less than |
| .LE. | less than or equal to |
| .E. | equal to |
| .NE. | not equal to |
| .GE. | greater than or equal to |
| .G. | greater than |

The equal sign is not permitted as a relational operator.

To test one of the four console switches, replace $e_1$ with the words "CONSOLE SWITCH", use .E. or .NE. for the relational operator, and make $e_2$ the switch number. Examples:

```
IF CONSOLE SWITCH .E. 1 $ Y,N

IF CONSOLE SWITCH       $ .E. 1, .NE. 2, .E.4
```

ACTION STATEMENTS

If an action statement preceedes the conditions in a table, it is executed unconditionally, and may not specify any entries. A dollar sign following the statement is optional.

Action statements which follow the conditions must specify with their entries the alternatives for each rule. This may be done in two ways:

Limited entry - If the statement is complete in the stub, the entries need only indicate for which rules the action is to be performed. This is done by using an entry of x in those rules for which the action is desired, and leaving all other entries blank. Example:

```
A = B $ , X , , X , X , ,
```

(done only for rules 2, 4, and 5)

Extended entry - If the statement changes slightly for different rules, that part which changes can be used as an entry. When no action is desired, the entry is blank. Example:

```
A = $ B , , C , , , 2.*E , 3.
```

(no action for rules 2, 4, and 5)

A more detailed description of each type of action statement follows. The iteration statement is an exception and is under its own heading.

Arithmetic

Replacement statement

This is the only statement which does not start with a key word, since it is uniquely determined by its equal sign. In the extended entry form, the entries must include everything to the right of the equal sign. The equal sign itself must remain in the stub. No checking is done for mixed mode or any of the other possible faults in arithmetic. This error detection is left to the FORTRAN compiler.

MOVE

This is a reverse replacement statement. The expression following the keywork is moved to the variable name specified in the entry. It is not useful in the limited entry form since it could be replaced by an ordinary arithmetic statement. Example:

```
MOVE  B * * 2  $  A, , X, , D,
```

Input/Output

If the format number is omitted in a READ statement, a free-format read subroutine is used. All other statements are copied directly. Permissable statements: READ, ACCEPT, ACCEPT TAPE, PUNCH, PRINT, TYPE, PUNCH TAPE, FIND, FETCH, RECORD. Since commas are used in these statements, the extended entry form should not be used when a list is specified. Example:

```
PRINT    $ 3 , , 4 , 3

PUNCH      2,A,B,C $ X,X, , X
```

Control

GO TO TABLE n

Unconditional transfer. Extended entry form has GO TO TABLE in stub

PAUSE and STOP

Extended entry form does not exist

Internal Procedure

DO TABLE n

Calls an internal procedure which starts at table n. DO TABLE must be in the stub for extended entry.

RETURN n

Provides a return for the procedure which started at table n.

Subprogram

Functions and subroutines are handled the same way as in FORTRAN.

SUBROUTINE and FUNCTION

These statements are non-executable and so have no entries.

CALL

The extended entry form of this statement should not be used.

## ITERATION STATEMENT

LOOP n, i = $e_1$ , $e_2$ .r. $e_3$, $e_4$

where n = the number of the table to be executed when the loop is satisfied.

i = the index, a variable name

$e_1$ - $e_4$ = arithmetic expressions

.r. = a relational operator

i and $e_4$, and $e_2$ and $e_3$ must be of the same mode. The index is first set equal to the starting value $e_1$. The terminating condition, $e_2$ .r. $e_3$, is tested. If it is true, table n is executed. If it is false, the next statement is executed.

When a loop statement appears, it must be the first executable statement in a table. Hence, only one loop statement per table is allowed.

Shorter forms allowable

If $e_2$ is a single variable name which is the same as the index, it may be omitted.

If $e_2$ is omitted, and .r. is equal to .G., it may also be omitted.

$e_4$ may be omitted if it is equal to 1.

Examples

LOOP 3, I - A, B**2 .L. 4 ,-K

This may be read as "LOOP and then go to table 3, for I starting at A, until B squared is equal to 4, in steps of -K."

LOOP 2, C = 1, 10, D

This may be read as "LOOP and then go to table 2, for C starting at 1 and continuing through 10, in steps of D."

A loop is ended by the following statement:

```
END LOOP m
```

Where m is the number of the table which contains the loop statement. m must be an unsigned integer constant.

Any number of end loop statements may be given for one loop.

The index or any of the parameters of a loop statement may be changed within the loop.

A branch may be made to any table within a loop if the index and all parameters are defined.

## INTERNAL PROCEDURE

An internal procedure is referenced by the number of its starting table. It may be extended over any number of tables, and return at as many points as desired to the table which called it.

The procedure is called by the following statement:

```
DO TABLE n
```

where n is the number of the starting table.

Once the procedure is started, it may branch to any table in the mainline program, another procedure, or a table within the procedure.

The following statement causes a branch to the statement following the one which called the procedure:

```
RETURN n
```

where n is the number of the starting table.

Any number of return statements may be used.

There may be a maximum of 99 procedures in one program, with no more than 63 references to each one.

An internal proceaure may be executed by a GO TO TABLE command, but a RETURN n statement must not be attempted until a DO TABLE n statement has been executed at least onee.

## SPECIFICATION STATEMENTS

### DIMENSION, EQUIVALENCE, COMMON, and DEFINE DISK

No change from the form or position of these statements from that specified in FORTRAN.

### FORMAT m ($s_1$, . . . , $s_n$)

The format number, m, has been moved to the position shown so that it can not be confused with a table number. m must be ≤99. A format statement may appear anywhere in the program.

# APPENDIX B

John Moschetti

October 5, 1965

Appendix B

Elliott Company is a Division of Carrier Corporation and a manufacturer of engineered industrial products. We are specialists in industrial compression, industrial vacuum, and power recovery. Our major products are air and gas compressors, steam turbines and related equipment, power recovery equipment, steam jet ejectors, liquid strainers, marine equipment, and tube tools.

I'm sure many of you recognize that these products require design of a mechanical engineering nature. Since we have an IBM 1620 Computer as part of our Engineering Department, we felt that we should use the computer as a design tool and couple it with the engineer to achieve a higher level of understanding and to produce the best design possible as output. Today this process of using the computer as an aid to design is commonly referred to as "Automated Design Engineering". However, this title is not entirely descriptive because if you check the possibilities you will find that manufacturing considerations should also be included.

Our aim is to use the computer after the receipt of an order to perform four major functions. They are: design logic, equations and computations, design checking, and engineering paper work generation. Although we are using the computer to perform the first three of these functions, we are also interested in computerizing the paper work generation. Once a design has been established the engineering paperwork generation begins. The basic information must be listed, transcribed, sorted, and put in the form required for manufacturing. The checking and paper generation phases are usually routine repetitive tasks, and frequently take longer than the original design process.

Decision tables are a necessary tool to implement this automated system. The decision table principle is neither particularly new nor revolutionary. As a concept it is very easy to understand. Its significance lies primarily in its power to capture design logic, and where practical, in its use as a source program that can be directly compiled into an object computer program. Our needs are for a decision table translator that can be used with our 1620 computer. That is why we are developing DECTRAN.

The bibliography presented is just a small portion of the work that has been done in the area of Decision Tables. At present, a program for a 1401 computer equipped with magnetic tapes is available from the 1401 library. It is Decision Logic Translator - 1401-SE-05X. By using the DECTRAN program the advantages of decision tables will be available to 1620 Users. We believe we have also added some sophisticated decision table methodology.

# BIBLIOGRAPHY

1. Cantrell, H. C., King, J., and King, F. E. H., "Logic Structure Tables". Communications of the ACM, 4(June 1961), 272-275.

2. Dixon, Paul. "Decision Tables and Their Applications". Computers and Automation, April 1964, pp 14-19.

3. Egler, J. F. "A Procedure for Converting Logic Table Conditions into an Efficient Sequence of Test Instructions". Communications of the ACM, 6(September 1963), 510-514.

4. Grad, Burton. "Tabular Form in Decision Logic". Datamation, July, 1961, pp 22-26.

5. IBM 1401 Decision Logic Translator (1401-SE-05X) Program Reference Manual. White Plains, New York. IBM Technical Publications Department.

6. Kavanagh, T. F. "TABSOL - The Language of Decision Making". Computers and Automation. September 1961, pp 15-22.

7. Kirk, H. W. "Use of Decision Tables in Computer Programming". Communications of the ACM, 8(January 1965), 41-43.

8. Montalabano, Michael. Egler's Procedure Refuted. (Letter to the Editor). Communications of the ACM, 7(January, 1964), 1

9. Press, Laurence I. "Conversion of Decision Tables To Computer Programs". Communications of the ACM, 8(June, 1965), 385-390.

# An Open Shop for Engineers

Lawrence E. Wright
Sprague Electric Company
North Adams, Mass.

Two years ago the Electronic Engineering program team held its first meeting, with at most ten members present. At that time a seeming paradox was noted by all of the participants; namely, that electrical engineering is an extremely fertile field for computer applications, yet many engineers are passive or even resistant to computerizing their problems. In the intervening years, during which the engineering team has grown many-fold, techniques of circuit analysis, simulation, statistical programs and computer process control have become widely practiced, and young men who have had computer exposure in school are alleviating the problem of lack of use. There still remains, however, the question of convincing older men that the computer is a useful, even essential, tool of their trade, and that furthermore they can make use of the machine with a minimum of re-training. This paper is a progress report on one attempt to solve this problem, and at the same time an appeal for advice from those who may be further along.

The job of converting men with years of experience in lab work and hand techniques into computer users is in many ways more difficult than that of training students in school; time is hard to find, classroom habits are lost and in many cases the audience is skeptical rather than eager. However, we can hardly wait for attrition to bring to the lab what are to us the manifest advantages of computers, and it is impractical to teach programmers engineering. Hence we must bring the computer to the engineer, and we are led to the concept of an open shop operation. To successfully re-train engineers three conditions are needed - motivation, involvement and education. The open shop provides the first two, as well as allowing a man, after he is trained, to experiment more freely with his problems. That leaves us with finding a suitable method of education.

We began our open shop training in what might be called the classical manner - by holding classes for all interested people (or employees of interested bosses) and dispensing large doses of Fortran, coding forms and manuals. The result was a deafening silence! In retrospect I can spot many valid reasons for the failure of this program. First, it required time and patience in excess of that which was available. Secondly, the presentation of the full Fortran language can be a confusion of rules and exceptions to the beginner. Thirdly, people were looking at Fortran to find out how to solve their problems - and all they found were ways to code the solution once it is known - too much education, too little motivation and involvement.

The second go-round was considerably different in approach, and far more successful in execution. We began with the premise that it is necessary and sufficient to teach only a basic subset of Fortran, but to teach it thoroughly by means of example and actual supervised time on the machine for each student. The large classes were changed to small workshops, and the examples were chosen to correspond, as much as possible, to real problems faced by the particular group. Stress was put on flow charting.

Because I have not encountered a comparable class schedule in the literature, or a comparable text arrangement, it might be worth a few minutes to outline our lesson plan. We have tried to use four two-hour sessions, held every other day, as follows:

Day 1: Arithmetic, expressions, library functions, = , GO TO,
IF (    ).

Day 2: Coding Form, Dimension, Subscripts, STOP, PAUSE END.

Day 3: Machine time to run assigned exercise.
READ, WRITE, PUNCH with I, F, E, H formats.

Day 4: Machine time for exercises, procedures and review.

This Fortran subset is enough to solve any problem, and avoids the confusing parts of the language where possible. We have found that DO, computed GO TO , etc., can easily be learned by the student after the basics are well in hand, and when (and only when) they are motivated by need. After the class work, the men are urged to pick a problem of their own choosing, discuss methods of solution with our staff, then write their own program. This must be done soon or the class work becomes useless.

I wish we could report that this approach has led to total success, but such is not the case. Less than a third of our students have become programmers in any sense; however, we have made gains in computerizing problems formerly done by hand- or more strongly, not done at all. Areas of engineering such as capacitor design, production specs and process controls have been shifted to the computer by the engineers involved. This gain is extremely important and is independent of whether the engineer has learned enough to do the actual programming. In fact, we insist as a matter of self-preservation that our staff program any job that is going to be a routine long-running project, for the engineer will undoubtedly take on new problems while we are stuck with operating, updating and maintaining the program. All in all, however, the open shop seems to be the best answer to the question of getting the jobs to the machine, even if it is not ideal for getting the solutions through the machine.

Recent advances in hardware and software are adding a new dimension to this discussion. Engineers who have never used a computer have nonetheless read, or been told by IBM salesmen of the development of scopes, remote consoles, time sharing, on-line programming systems and the other innovations designed to allow closer man-machine relations. These devices do make it easier to explore problems and search for solutions, and the engineers are justifiably excited at the prospect. There is a tendency, however, for the untrained to claim that all their difficulties will be mitigated if we can give them a remote console to play with, especially if it comes equipped with a Quicktran-type language. The problem, of course, is that you still have to know how to communicate your ideas in whatever language is chosen. This becomes less severe if a language is developed which contains enough macro instructions or subprograms so that the solution becomes just a question of calling macros. This should include complex arithmetic, matrix operations, statistical formulae and numerical methods as basic, and flexible programs such as the 1620 ECAP as desirable. This will allow a neophyte to run any problems that fit into the framework without worrying about programming as we now think of it. There is still no substitute, however, for a logical approach to the problem. The main difficulties we have encountered have been in getting our men to assimilate the ideas behind program writing, and in problem analysis. In these areas, there is still much to be said for drawing block diagrams and in pre-scanning programs by hand; I have difficulting envisioning some of our beginners attempting on-line programming! Our open shop work is admittedly low in sophistication, but the short delays for punching and waiting for machine time have not yet impaired this effort. This is not to say that the new techniques are not going to be of tremendous value to the practicing engineer; but our experience would seem to indicate that they will not do away with the need for careful, well-motivated training of the beginner.

In conclusion, for those who are faced with prodding engineers to get their problems on the machine, the open shop is a workable technique. For those who have such a program in good shape, the rest of us would appreciate your comments.

# SMOLDS

## SYRACUSE MANAGERIAL ON-LINE DATA SYSTEM

SHARON M. STRATAKOS
Syracuse University Research Corporation
P. O. Box 26
University Station
Syracuse, New York

## PURPOSE

The SMOLDS programming system is designed to provide the User-manager with a large bank of data consisting of documents familiar and useful to him and a language with which he can rapidly and easily retrieve, process, and display information from the file while sitting at the console of the computer.

## DATA BASE

The data base consists of a set of documents which may be identified by name or number and which consist of an arbitrary number of blocks, each of which contains one item of information and may be identified by name or number. An example might be the typical personnel form consisting of blocks of information such as name, address, telephone, birth date, etc. The number of blocks in a document, the number of characters in a block, and the mode (alphabetic, chronological, integer, or decimal) of the block contents are completely arbitrary.

The data base has a matrix-like organization in which each row vector corresponds to a single form and each column vector to a set of block values; thus each unit of information may be called from the file according to its unique position within the array.

Rome Air Development Center Form 77 is currently being used as the data base for the SMOLDS system. Data are punched from the Form 77, read in alphabetic format, sequence numbered, and loaded on the disk file. The form is described to the system by a set of tables which contain block numbers, lengths, and modes.

## INTERNAL STRUCTURE

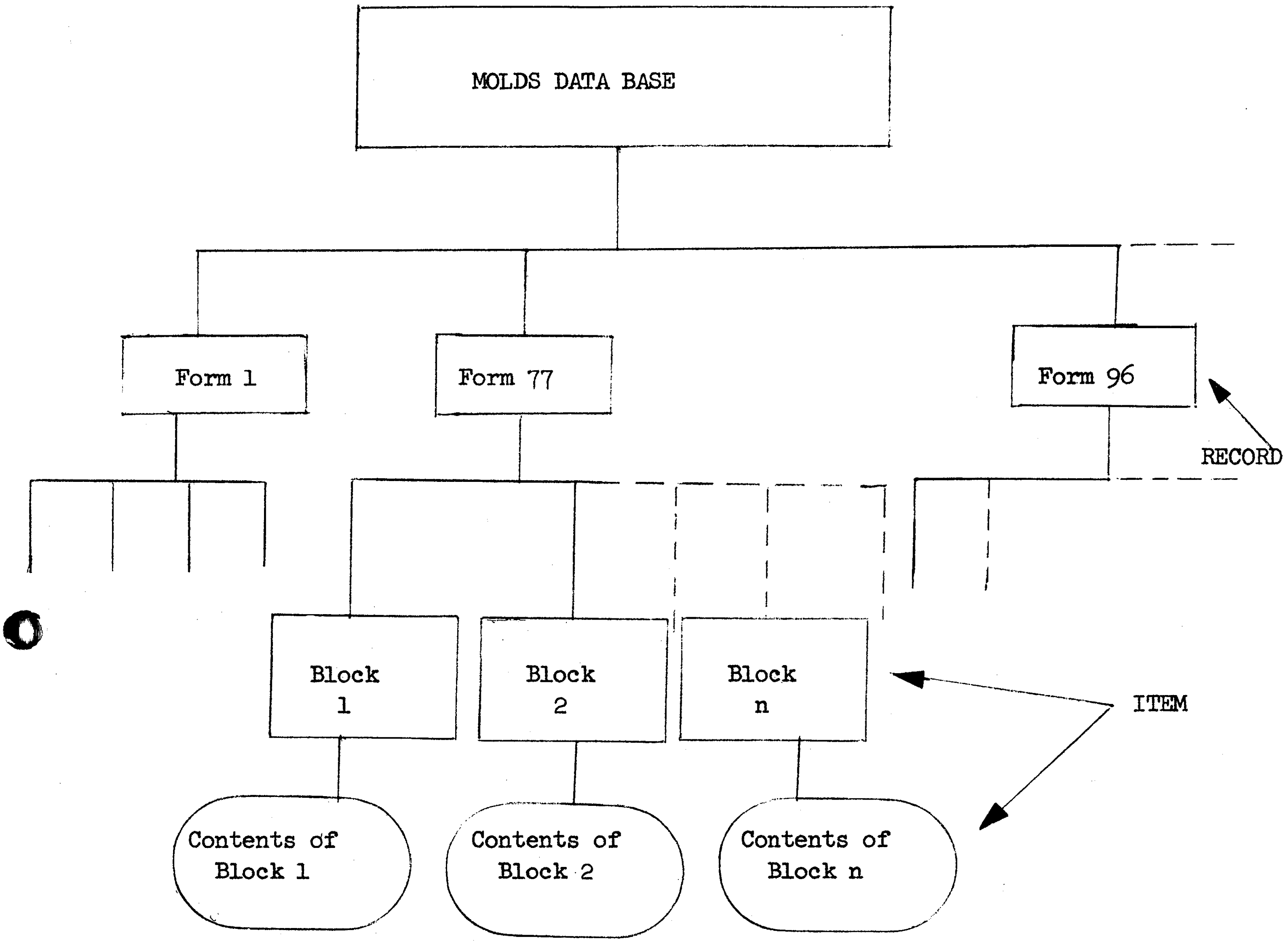
Since the purpose of SMOLDS is to provide the user with a repertoire of independent but often related commands, certain qualities are required of the system:

1. That the operations may be linked in any conceivable order with any amount of repetition.

2. That the operations may be used independently only when needed.

3. That the results from one operation may be retained and made available for use by the others.

4. That a convenient method of adding, deleting, or revising operations exist.



DEFINITIONS

| | |
|---|---|
| ITEM: | the contents of a block of a form |
| RECORD: | the set of blocks constituting a form |
| FILE: | a set of records of the same type |
| DATA BASE: | the set of all files for the system |

STRUCTURE OF MOLDS DATA BASE

# REQUEST FOR PREPARATION OF PURCHASE REQUEST

OBJECTIVE

SIGNATURE OF DIRECTORATE CHIEF

SIGNATURE OF LABORATORY CHIEF

SIGNATURE OF ENGINEER

TYPED NAME OF DIRECTORATE CHIEF

TYPED NAME OF LABORATORY CHIEF

TYPED NAME OF ENGINEER

TEL EXT

2. COORDINATION ROR □ YES □ NO ACCOMPLISHED □ YES □ NO
□ DOD □ ARPA □ NASA □ USER □ OTHER

3. RESOURCES REQUIRED □ YES □ NO AVAILABLE □ YES □ NO
□ GFP □ GLP □ FACILITIES □ OTHER *(Specify)*

4. NOMENCLATURE REQUESTED

5. REQUIREMENT NO AND PARAGRAPH SUPPORTED

6. PRECEDENCE

7. FORM 77 DATE

8. PRCS DATE

9. COMMITTEE ACTION

10. FUNDS AVAILABLE □ YES □ NO

11. FY FUNDS
FY.

12. CONTRACT DELIVERY

| MINIMUM | MAXIMUM |
|---|---|
| | |

13. BP SN

14. AMOUNT

15. UNSOLICITED PROPOSAL
RCK

16. SOLE SOURCE COMPANY

17. OPEN BID □ YES □ NO

18. NEW CONTRACT □ YES □ NO

19. CONTRACT EXT □ YES □ NO

20. OVER, RUN □ YES □ NO

21. TYPE OF BUY □ RADC □ AMC □ OA □ MIPR □ CSO

22. TITLE

23. PRIORITY

24. TASK NO.

25. PROJECT NO.

26. PROG STR

27. FROM *(Lab)*

28. PR NO.

29. DIR. SER NO.
X

CHANGE

In order to achieve the desired flexibility SMOLDS was written as a set of independent subroutines sharing a common storage area and linked by a monitor-type mainline program. Since each subroutine corresponds to a particular command in the repertoire, the order in which they are executed and the frequency with which they are called is completely determined by the user. As the need for new or revised commands arises, they may be added to the system without changing those already in existence.

Appendix A shows the storage map for the SMOLDS system.

## LANGAUGE

The SMOLDS language depends upon the assumption that the user can make the necessary association between his query and the data base. For example, consider the manager who is trying to evaluate the expenditure on computer work during the year 1963. In particular he would like to know which computer contracts in excess of $50,000 were initiated in 1963, excluding the one entitled "Signals, Processing, and Noise." In order to describe his query in terms of the data base he must consider:

A. Which form and which blocks contain the information of interest.

B. Which criteria must be satisfied by the values in those blocks.

C. Which combinations of block values must be satisfied.

Considering the present example, the user determines that the solution to his query may be found in the file of Form 77's and that:

1. Computer contracts have a keyword equal to "COMPUTER" in Block 1.

2. Contracts in excess of $50,000 have a value greater than "50,000" in Block 14.

3. Contracts initiated in 1963 have a value between "01 JAN 63" and "31 DEC 63" in Block 7.

4. The contract entitled "Signals, Processing, and Noise" contains a value equal to that title in Block 22.

5. The contracts in question must have the properties 1, 2, and 3 but not 4.

## RETRIEVAL SUBSYSTEM

The object of the retrieval portion of SMOLDS is to extract and place into some intermediate storage area a subset of forms from the data base which fulfill certain criteria, and to provide some convenient notation by which this subset may be referenced for subsequent processing or display.

The retrieval language consists of two types of statements which specify the conditions, either basic or compound, under which a given form is to be selected from the data base.

The basic condition defines an arithmetic relationship between an input value and the contents of a block on some form. The SMOLDS command "KNOWN" is used for specifying basic conditions such as those numbered 1-4 in the previous sample problem.

```
GENERAL FORM

        KNOWN            Label          Form/Block/Relation/Value/

EXAMPLES

        KNOWN            AMOUNT         77/14/G/50000./

        KNOWN            START          77/FORM 77 DATE/B/01 JAN 63/31 DEC 63/
```

The labels are completely arbitrary although they are usually selected for their mnemonic value; they may consist of from 1 to 10 alphanumeric characters. The form name or number and block name or number must be spelled exactly as they appear on the source document including spaces and punctuation. The allowable relationships are equal (E), not equal (NE), greater than (G), greater than or equal (GE), less than (L), and less than or equal (LE). The slashes must follow each operand and the entire command must fit on one typewritten line.

The KNOWN operator causes a search through the data base for documents which possess the desired property. As such documents are found, their sequence numbers are recorded in a list which may subsequently be referenced by the name given it in the label operand. At the completion of the scan, the list of sequence numbers are written on the disk and the label and length of the list are recorded in a table which resides in the common storage area of memory.

The compound condition defines a logical relationship between two conditions, either basic or compound. The SMOLDS command "DEFINE" is used for specifying compound conditions similar to number 5 in the sample problem.

| | | | |
|---|---|---|---|
| GENERAL FORM | | | |
| | DEFINE | Label | $L_1$/Relation/$L_2$/ |
| EXAMPLE | | | |
| | DEFINE | UNION | AMOUNT/OR/START/ |

The label has the same properties and performs the same function as in the KNOWN command. L1 and $L_2$ refer to labels defined in previous condition statements, either basic or compound. The allowable relationships are AND, OR, and NOT. The slashes must follow each operand and the entire command must fit on one typewritten line.

The DEFINE operator causes the lists $L_1$ and $L_2$ to be read from the disk and a new list of document numbers to be generated. The data base itself is not searched; the presence or absence of a particular document number in one or both of the lists determines whether it is to be included in the new list.

Although only one logical operation may be performed for each retrieval statement, by repeated use of the KNOWN and DEFINE commands, conditions of any complexity may be constructed. Appendix B shows a program for the solution of the sample problem.

PROCESSING SUBSYSTEM

The object of the processing portion of SMOLDS is to provide the user with a basic set of operations with which to process the results of previous retrieval operations in preparation for output. While the retrieval subsystem produces lists of raw data, it is more frequently the case that the user seeks some function of these raw data; i.e., totals, averages, ordered lists. On-line processing capabilities enable the user to achieve the desired result rapidly and accurately, and to by-pass the time consuming task of outputting lengthy lists of unnecessary data.

The processing language, like the retrieval language, assumes that the user can make the necessary association between his query and the data base. In the case of the processing subsystem, the data base consists of lists of documents extracted by previous retrieval operations. Referring to a particular list by the label assigned to it during retrieval, the user may process any block of the document.

GENERAL FORM

```
Command Name/Label/Block/
```

EXAMPLE

```
ORDER/INT/FORM 77 DATE/
```

The label refers to some list defined in a previous retrieval command, either basic or compound. The block name or number may refer to any block of the source document, not necessarily the block which determined the retrieval, but must be spelled correctly. The slashes are used to separate operands.

The processing operators are divided into two classes depending upon whether they result in a single value or a list of documents. Class I operators, which produce a single value, are AVERAGE, TOTAL, MEDIAN, VARIANCE, MAXIMUM, and MINIMUM. Class II operators are ORDER, REVERSE ORDER, and PROFILE.

AVERAGE, TOTAL, and VARIANCE operate only on blocks which contain numeric data. MAXIMUM, MINIMUM, ORDER, and REVERSE ORDER automatically determine the mode of the data to be processed and are capable of performing alphabetic, numeric, and chronological calculations.

## DISPLAY SUBSYSTEM

The object of the display portion of SMOLDS is to provide the user with a set of commands with which to specify and control the output of previous retrieval or processing operations. While the display subsystem has not been implemented to date, both tabular and graphic displays are anticipated.

For the present, the results of processing operations are displayed automatically as a means of verifying their validity. Such results are not retained in memory and may not be used in subsequent commands.

In order to verify the results of retrieval operations, the user may select either of three output options, COUNT, PRINT, or DISPLAY.

```
GENERAL FORM

          COUNT/Label/
------------------------------
EXAMPLE

          COUNT/UNION/
```

The label refers to some list defined in a previous retrieval command, either basic or compound, and the slashes separate operands. The result of COUNT is a single number typed at the console.

```
GENERAL FORM

          PRINT/Label/Block/
------------------------------
EXAMPLE

          PRINT/UNION/PROJECT NO./
```

The block name or number may refer to any block of the source document regardless of length or mode, but must be spelled correctly. The output medium is the on-line printer.

```
GENERAL FORM

          DISPLAY/Label/
------------------------------
EXAMPLE

          DISPLAY/UNION/
```

The output consists of reproductions in exact format of each document in the referenced list. The output medium is the on-line plotter.

## UTILITY SUBSYSTEM

The object of the utility subsystem is to enable the user to delete entries from the label table or to clear the common storage area, reinitialize the system, and begin again.

The DELETE operation provides the former capability.

GENERAL FORM

```
DELETE/Label/
```

EXAMPLE

```
DELETE/UNION/
```

"Label" is any label defined in some previous retrieval operation, either basic or compound and the slashes separate operands. The label is removed from the table and the entry made available for future use.

Reinitialization is accomplished with the CLEAR command.

GENERAL FORM

```
CLEAR
```

EXAMPLE

```
CLEAR
```

Only the command name is used. The common storage area is set to zeroes and the system is reinitialized.

## REMARKS

It should be stressed that the present SMOLDS system, although operable, is as yet incomplete. In addition to the implementation of display capabilities, future versions of SMOLDS will contain the following:

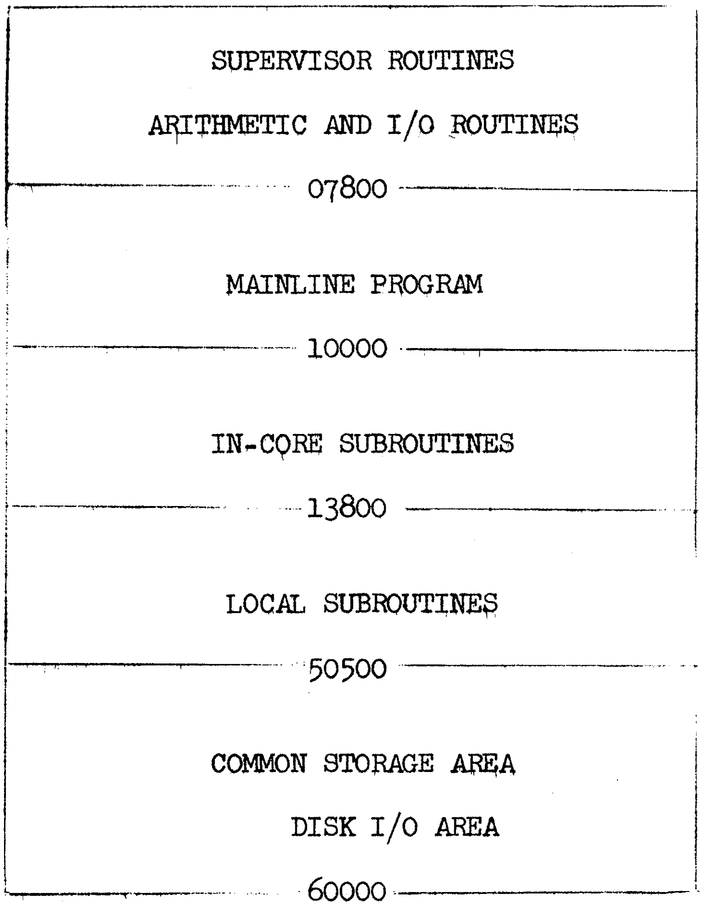
1. Multiple-form data bases and interform retrieval.

2. A STORE command for the manual entry of constant data by the user.

3. Processing commands for the addition, subtraction, multiplication, and division of corresponding elements of two lists.

4. On-line definition by the user of new operators constructed from a sequence of operators available in the processing subsystem.

## APPENDIX A



STORAGE LAY-OUT FOR SMOLDS

# APPENDIX B

```
KNOWN          COMPUTERS          77/OBJECTIVE/E/COMPUTER/
KNOWN          DOLLARS            77/AMOUNT/G/50000./
KNOWN          START              77/7/B/01 JAN 63/31 DEC 63/
KNOWN          NAME               77/22/E/SIGNALS,PROCESSING,AND NOISE/
DEFINE         INT 1              COMPUTERS/AND/DOLLARS/
DEFINE         INT 2              INT 1/AND/START/
DEFINE         RESULT             INT 2/NOT/NAME/
PROCESSING OPERATIONS
DISPLAY OPERATIONS
```

SOLUTION TO SAMPLE PROBLEM

# SORTING ALGORITHMS AND THEIR USE WITH A 1620 WITH TWO DISK DRIVES

by

Janet E. Allen
Pioneer Hi-Bred Corn Co
1206 Mulberry St.
Des Moines, Iowa 50308

1620 Users Group
Eastern & Midwestern Joint Meeting
October 6-8, 1965
New York, N. Y.

# SORTING ALGORITHMS AND THEIR USE WITH A 1620 WITH TWO DISK DRIVES

This discussion represents a preliminary attempt to define sorting procedures and programs making most efficient use of a 20K Model I 1620 with two disk drives. It is hoped that this proposal will elicit responses from other users experienced in such techniques.

A typical program involves reading cards, sorting, and editing, indicating cards in error. Corrected cards will later be read, to be inserted in the file in the correct sequence. The file is again sorted and processed, and the output punched. This output may be again sorted, and further processing done. Input to a program such as this is from 4000 to 12,000 cards.

Presently, most of our sorting is done on the mechanical sorter. Some of our programs use the 1620-1311 Sort/Merge Program, SM-047. We feel, however, that a program not so general and written especially to use two disk drives would probably be more efficient. It may be that converting the program SM-047 to use the two disks, thus allowing for longer strings and merges, would be the best approach.

There are many articles describing various sort algorithms. The most thorough and helpful I have found is the papers of the ACM Sort Symposium, published in the May, 1963, issue of the "Communications of the ACM". These, and others, are listed at the end of this discussion. The sort techniques discussed below are described in the article in that issue "Sorting on Computers", by C  C. Gotlieb. In the same issue the article "Some Characteristics of Sorting in Computing Systems Using Random Access Storage Devices", by George U. Hubbard, is very helpful. As he points out, the use of a random access device presents considerations different from those using tapes.

Proposed Procedures

The ideal sort program, of course, is an extremely efficient one, which is relatively uncomplicated to write, but which runs very fast. It should also be fairly easy to interrupt and recover. This paper is no attempt to describe the ideal, but a start must be made somewhere.

The usual procedure is to form a key on which to sort, store the entire record elsewhere, and sort the key only. Attached to each key is some indication of where the record is stored. When the records are to be sorted later, there are thus problems of reading the records from all over the disk. Sorting the entire record initially, however, leads to storage problems, and additional seek time on the disk. Since there may be times when the entire record file need not be sorted, as at the beginning of our edit routine, we will sort the keys only.

As each card is read, the key and tag are formed. If the key is 20 numeric digits and the address of the record 5, the complete tag is 25 digits, 800 tags per cylinder. The card is stored in a buffer in storage, which, when full, is written on disk 2 The tags are stored in another buffer in storage, which is written on disk 1. These blocks of data are stored in the same order as read. This allows the seek time for the next cylinder to be overlapped with reading the next card. Another method might be to distribute the cards on disk 2 according to the value of the most significant column(s) in the key. This has the advantage of ordering the records and shortening the key, but creates problems of specifying areas of the disk and allowing for overflow of these areas. Since these areas would probably be different cylinders, more seek time would be necessary. Therefore, the cards will be stored as read.

When all the cards have been read, sorting can begin. The general method is to form strings--sequenced sets of tags--within each cylinder, merge these to form entire sorted cylinders, then merge the cylinders. In general, the larger the merge, and the length of the strings, the more efficient the sort, taking into account the amount of storage both in core and in the cylinder. It seems feasible to form strings of a quarter cylinder, thus having a four-way merge within the cylinder.

To sort each string, either of two methods, or a variation, seems appropriate. Using the two-way merge (Fig. 1), pairs of keys are examined, with the smaller placed first in the output buffer. This is repeated, with the groups doubling, until the list is sorted. This method requires n passes, where the size of the group is $2^n$, and an output buffer the size of the area being sorted. The other method (Fig. 2) is called the pair exchange, in which each key in the string is compared to the next one. The two are exchanged to put the smaller one first. This results in the largest key always being placed at the end, thus requiring one less compare each pass. The order of the numbers may decrease the number of passes required. If an indicator is set when an exchange is made, this can be tested. When no exchanges have been made, the sort is complete. This method does not require an output area. These methods have been selected because they are uncomplicated to program and because they do not need large amounts of core storage.

When the strings are formed on each cylinder, the most efficient approach seems to be first to sort each cylinder. Blocks of each string from the first cylinder on disk 1 are read and merged into an output area. As each input block is exhausted, the next block is read, until the whole string has been read. This is done for all stringson that cylinder. As the output area fills, it is written on disk 2. This then results in a sequenced cylinder on disk 2. This is done for all cylinders on disk 1. Now the problem is to merge the cylinders, all on disk 2. The most obvious solution, and probably the most efficient, is to merge from disk 2, and write the output on disk 1. The input areas, into which blocks of data are read from disk 2, should be as large as possible, minimizing the number of seek operations. Probably a two-way merge would be the most efficient. One other possibility is to write one cylinder of data from disk 2 to disk 1, merging this with another cylinder from disk 2 into as large an output area in core as possible, writing this output into another area on disk 1. The transfer of data from disk 2 to disk 1 initially, however, would probably use whatever time would be saved in the seek operations. This again would depend on the amount of core storage available.

Methods of obtaining the original records are discussed in the article mentioned above.

## Conclusion

This paper is meant to stimulate discussion of programs and techniques being employed by other users. It is by no means a thorough analysis of the subject, but is our first thinking in the area. Sort programs are fairly complicated, and it is hoped that, through the Users' Group, we can benefit from each others' experiences.

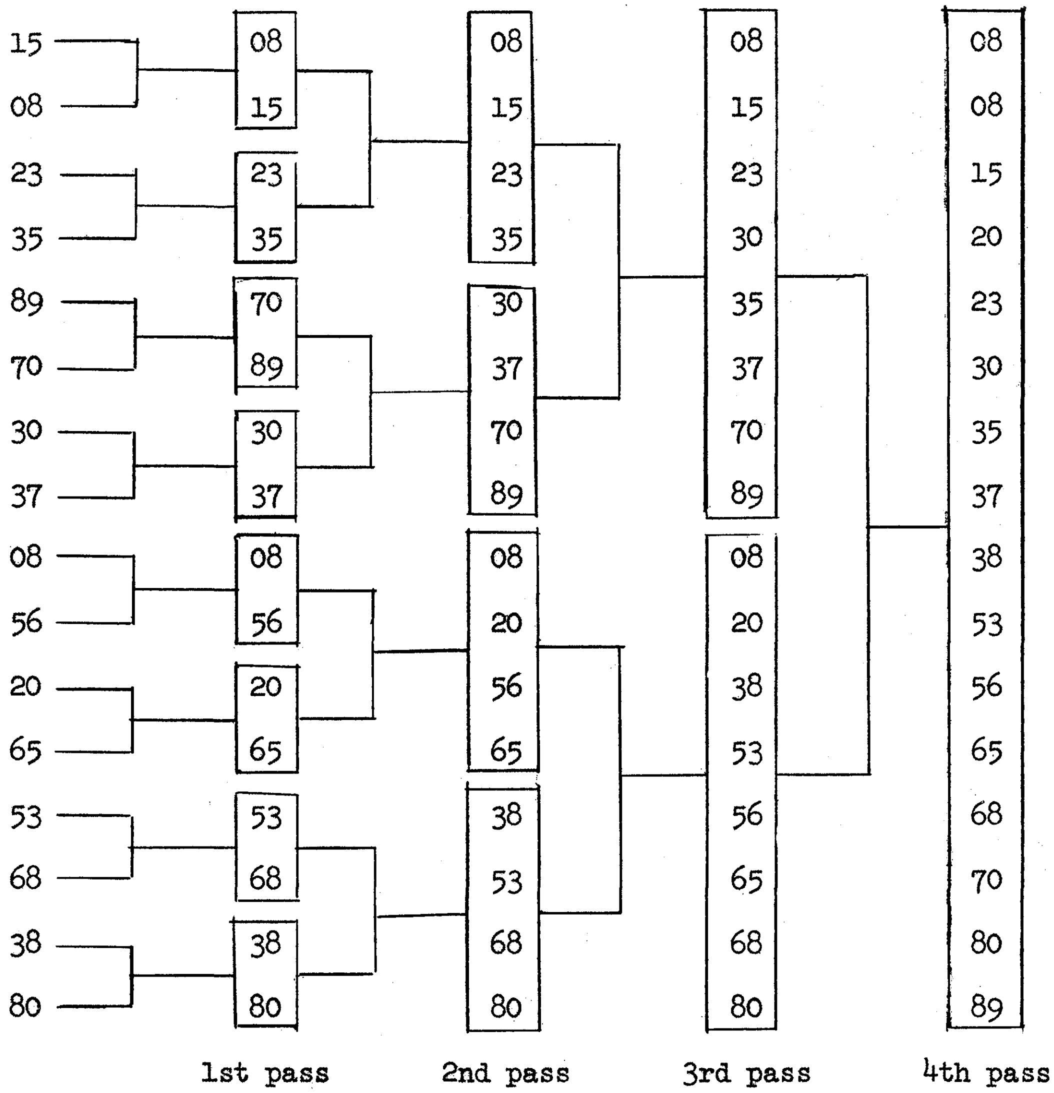
## References

1. Gotlieb, C. C. "Sorting on Computers" Communications of the ACM 6(1963), 194-201.

2. Hubbard, George U. "Some Characteristics of Sorting in Computing Systems Using Random Access Storage Devices" Communications of the ACM 6(1963), 248-255.

3. Flores, Ivan "Analysis of Internal Computer Sorting" Communications of the ACM 1(1961), 41-80.

## TWO-WAY MERGE



|  | 1st pass | 2nd pass | 3rd pass | 4th pass |
|---|---|---|---|---|
| 15 | 08 | 08 | 08 | 08 |
| 08 | 15 | 15 | 15 | 08 |
| 23 | 23 | 23 | 23 | 15 |
| 35 | 35 | 35 | 30 | 20 |
| 89 | 70 | 30 | 35 | 23 |
| 70 | 89 | 37 | 37 | 30 |
| 30 | 30 | 70 | 70 | 35 |
| 37 | 37 | 89 | 89 | 37 |
| 08 | 08 | 08 | 08 | 38 |
| 56 | 56 | 20 | 20 | 53 |
| 20 | 20 | 56 | 38 | 56 |
| 65 | 65 | 65 | 53 | 65 |
| 53 | 53 | 38 | 56 | 68 |
| 68 | 68 | 53 | 65 | 70 |
| 38 | 38 | 68 | 68 | 80 |
| 80 | 80 | 80 | 80 | 89 |

Figure 1

- - - - - - - - - - - - - - - - - - - - - - - -

## PAIR EXCHANGE

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 15 | 08 | 08 | 08 | 08 | 08 | 08 | 08 |
| 08 | 15 | 15 | 15 | 15 | 15 | 15 | 08 |
| 23 | 23 | 23 | 23 | 23 | 23 | 08 | 15 |
| 35 | 35 | 35 | 30 | 30 | 08 | 23 | 20 |
| 89 | 70 | 30 | 35 | 08 | 30 | 20 | 23 |
| 70 | 30 | 37 | 08 | 35 | 20 | 30 | 30 |
| 30 | 37 | 08 | 37 | 20 | 35 | 35 | 35 |
| 37 | 08 | 56 | 20 | 37 | 37 | 37 | 37 |
| 08 | 56 | 20 | 56 | 53 | 53 | 38 | 38 |
| 56 | 20 | 65 | 53 | 56 | 38 | 53 | 53 |
| 20 | 65 | 53 | 65 | 38 | 56 | 56 |  |
| 65 | 53 | 68 | 38 | 65 | 65 |  |  |
| 53 | 68 | 38 | 68 | 68 |  |  |  |
| 68 | 38 | 70 | 70 — |  |  |  |  |
| 38 | 80 | 80 — |  |  |  |  |  |
| 80 | 89 — |  |  |  |  |  |  |
| Pass 1 | Pass 2 | Pass 3 | Pass 4 | Pass 5 | Pass 6 | Pass 7 | Pass 8 |

Figure 2

# PDQ FORTRAN COMPILE AND GO SYSTEM

Karl Dunn, Jr.
Rensselaer Polytechnic Institute

PDQ Fortran has been used at RPI since it was first available, on a completely open-shop basis. With the arrival of a 1311, it became desirable to store the processor, object programs, data and subroutines on disk, at the same time maintaining as rapid compilation and running as possible with enough simplicity for open-shop operating.

The resulting system consists of a self loading (to disk) deck of about 700 cards, plus five one and two-card programs used to start a compile-execute job, obtain a punched object deck, store the compiled portion of an object program in core image on disk, or rerun a recently compiled or core-image stored program. Minimum machine requirements at present are Model II with special instruction package, 40K core, one 1311, and card I/O. No options for use of a 1443 are available. Compile time is about 10% faster over C2 compile time due to no waiting for the 1622 punch. Load time is about five seconds per hundred cards of object program plus (very roughly) 2 seconds for each relocatable subroutine called (disk I/O statements call a special one.)

The compiler itself takes about 3 seconds to load. Execution time is the same as for C2-compiled programs. Disk FIND takes about 8ms if a physical seek is not necessary; FETCH and RECORD each take about 35ms on the average if no physical seek is necessary.

ACCEPT and CONTROL have been eliminated in the interest of decreased run time. Switch options during compile have been deleted so that switches may be set for the execute phase; a source list including addresses and symbol table is available with a control card. PRINT will output to the typewriter only the first 30 lines and will punch the rest. END, if executed, instead of stopping the machine, will simulate the load key. FIND, FETCH, and RECORD are used as Fortran IID with two exceptions: List elements must conform to PDQ limitations and no DEFINE DISK statement can be used, records always being 1 to 10 variables (and one sector) long.

Input to the system is a source deck preceded by a system call card, and if desired a list control card, and followed by data cards if any. The compiler is loaded to core in one piece (there is no diagnostic phase) and as long as no errors are detected, the object program is stored in the disk working area in card-image format, one card image to a sector.

At compile end, the loading phase begins, terminating by leaving the compiled program fully loaded. Execution begins immediately at address 07000. If core-image or punched objects are desired, a PAUSE must be the first executable statement. At pause, one of the two-card programs is used to effect the desired storage.

Rerun can start at the above load phase, or can be started either by using a prepared rerun card for core-image objects, or by loading a punched object deck.

The system occupies two complete cylinders plus 25 sectors of disk space which can be reallocated by minor changes to the self-loading system deck. This deck expects the Monitor I utility routines to be on the disk.

Plans for the future (for a library version):

(1) Set up segment link statements for overlay.

(2) Detect undefined variables.

(3) New arithmetic routines for extended exponent range and optional 10-digit fixed point numbers.

(4) Versions for 20 and 40K Model I's.

In summary:

Statements added: FIND, FETCH, RECORD, REREAD.
Statements whose functions have changed: PRINT, END.
Compiler operating changes: no switch options, list card.
Statements deleted: CONTROL, ACCEPT.

# GENERAL PURPOSE USE OF SORT/MERGE, 1620-SM-047

Presented at the

1620 Users Group Meeting

October 1965

New York City

Fred A. Hatfield
Systems Analyst

Line Material Industries
McGraw-Edison Company
Zanesville, Ohio

Sort/Merge (1620-SM-047) is an extremely versatile utility program. It seems ideally suited to the needs of a 1620-1311 system operating under Monitor 1 where handfulls of cards are being processed instead of drawersfull. Our desire was to use this program to simulate the operation of an 083 Sorter to create a new deck of cards in the desired sequence. Our approach was to develop a system which would read a deck of cards, sort them as specified, and punch out a new deck identical card for card but in the desired sequence. It is the purpose of this paper to pass along some of our findings so that others who find themselves in the same quandary we were with a manual in one hand and a deck of cards in the other can benefit from our experience.

The first problem we recognized was that, as received, there was no provision for getting the resequenced deck out of the machine. It will read the cards, sort like crazy, then go on to the next job. Part of the versatility of Sort/Merge is that it gives the user complete freedom after the sorting has occurred. If Phase 4 is executed, data records are read into core in sorted order, then placed in the disk output area. While the record is in core, one has access to it if one wishes by specifying a "Phase 4 Users Routine". It is by this means that you can output the record if you wish.

To reach our goal of a new deck, we provided a Phase 4 Users Routine (Appendix 1), punched the identifying information in cc 21-33 of Control Card 1, provided for an output area by cc 15-21 of Control Card 3 and ordered execution of Phase 4 by a 0 in cc 14 of Control Card 1 and a 1 in cc 33 of Control Card 3. The name given to the output program is SRTPCH, taken from Sort-Punch. Since the user's programs may not extend below 18950, we arbitrarily chose 19000 as the core location for our output routing. Appendix A of the manual states that the field address of the address of the record passing through core is 02690. Card 1020 of SRTPCH moves this address to a Punch statement, Card 1030 adds one to it to convert it to an alphameric address, and Card 1040 punches the record in alphameric mode. (Columns 3-8 of Control Card 1 identified the input as being alphameric and an 80-column card long).

By branching to 02836, further processing of the record is eliminated. Having punched the card, we have reached our goal. By not returning the record to disk, we will not be using the output area specified in Control Card 3. We have chosen to use the address of the input area for the address of the output area to fulfill what appeared to be the needs of the system, but yet to prevent possible damage to something permanent if we had guessed wrong about the operation of the system. This turned out to be a wise move, as we later found that even though we prevented the movement of the sorted records to the output area, the 0-RM-RM end-of-file indicator was moved. The first sector of our input area is destroyed. To prevent this, a separate output area of at least one sector would be needed.

Other general information concerning our approach (besides the 80 cc alphameric records and punched output previously described) is as follows:

1. All input is from one deck of cards.

2. The output will be in ascending sequence.

    Because of the colating sequence of the 1620, numbers come after letters, not before as they would on an 083 Sorter. We have accepted this even though we do not care for it.

3. The three control cards are read in each time Sort/Merge is used.

    For single purpose, sorting the control information could be kept on disk. Reading the information each time allows flexibility plus standard procedure.

4. Input and output are not blocked.

    This saves disk storage space and for our purposes blocking offered no benefits that we could see.

5. Hash totals were not used.

    To keep the system simple and to include nothing more than necessary, we ignored hash totals.

6. Return to Monitor when done.

7. So far, straight numeric data is handled in alphameric form for standard procedure purposes.

The preceding discussion applies regardless of the disk area used as specified in Control Card 1. To further simplify use of the program, we determined that our needs could be met handily by using the 24-cylinder Monitor work area for both the input area and the tag file work area. The only disk storage space permanently required for the system, then, is the resident space for the five phases of the program plus the output routine. This philosophy has worked well for us even though one pass through the Sorter is sometimes required to "block sort" the file down into bite-size chunks. For the sorting we do, we can generally handle from 1500 to 2500 cards in the 24-cylinder work area. Great care is taken to assure that the 24-cylinder limit is not exceeded. If it were, the DIM table would be damaged.

In calculating the space required, the following information may be helpful: The input records are stored on disk in the input area in the order in which they are read in. From the control field information in Control Card 2, the sorting information is extracted and a "tag" built up; one field containing all of the sorting information correctly positioned as to the relative significance as specified in Control Card 2. To the low order end of the tag is appended a sequence number which relates the tag to the correct record in the input file. To the tag for the first input record would be attached the sequence number 1, to the second a 2, to the third a 3, and so on. When input has been completed and the tag file established, the tags are sorted using two areas, each beginning with a new cylinder and each large enough to contain the entire tag file. These two areas are what is known as the tag file work area. Upon completion of the sorting of the tags, Phase 4 may be entered wherein the records would be located from the sequence number in the consecutive tags, moved into core and, in our case, punched out. The digit in cc 10 of Control Card 1 specifies the number of digits in the sequence number which will be appended to the tag. As a matter of practice, we use 4. We normally expect more than 1000 cards as a maximum but never 10,000. A larger number in cc 10 would increase the size the the tag file work area and lower the limit on the number of records which could be handled by the 24-cylinder work area, just as more sorting information would.

The instructions for calculating the space requirements of the input area and the tag file work area as given in the manual are quite complete. We assume that one additional record in the input file is taken up by the zero-record mark—record mark end-of-file indicator. Appendix B gives examples of the calculations, and the necessary control cards for two of our applications of Sort/Merge. Since, in most cases, the input area has been the limiting factor, the number of data records comes out to be one less than a whole multiple of 20,000/160 or 125 which is the number of 80-column cards which when converted to two-digit representation, will fit in one cylinder. The one, of course, is for the zero-record mark-record mark end-of-file indicator.

Another small point came up during the loading of the five phases of the program. Care must be taken to insure that the five phases are assigned consecutive DIM numbers. Once these numbers are known, they should be specified on the DELET and DLOAD cards (reloading is necessary as new mod levels arrive) to insure that the programs stay in sequence and that you know where they are. We assigned SORT, SORT1, SORT2, SORT3, and SORT4 to the five phases. The name and DIM number may be used on the DLOAD cards but only the DIM number may be used on the DELET cards. While the name is not used except on the XEQ SORT card, we like to have all programs identified in the Equivalence table by name.

While SM-047 is not a cure-all, it has been a great time saver for us. Future plans call for using it to up-date permanent disk files with data recorded by Fortran programs and later used by Fortran programs. Hash totals, blocking and input editing will be investigated. To really get a feel for what is happening, sort a small number of records by the simple means I have described, then dump those portions of the disk and see what it looks like.

I know of no better way to learn how to make use of a system as versatile as Sort/Merge. I hope that these comments will be helpful and that it will put others on the road to another worthwhile application of the 1620-1311 and Monitor 1. I would be interested in hearing from others who use the program, so that we can make better use of it.

## APPENDIX A

### Users Output Program Called by Phase 4

Version I
(Punches Cards)

```
‡‡JOB 5

‡‡SPS 5

*LIST TYPEWRITER
*STORE CORE IMAGE
*NAME SRTPCH


01010          DORG 19000                    19000
01020 START    TF   PUNCH+6,2690             19000 26 19030 02690
01030          AM   PUNCH +6,1,10            19012 11 19030 0000̅1
01040 PUNCH    WACD                          19024 39 00000 00400
01050          B    2836                     19036 49 02836 00000
01060          DEND START                    19000

END OF ASSEMBLY.
19048 CORE POSITIONS REQUIRED
00006 STATEMENTS PROCESSED


DK LOADED SRTPCH 0̅175 10̅51770̅011̅90001̅9000‡


END OF JOB
```

# APPENDIX A

## Users Output Program Called by Phase 4

Version 2

(Punches Cards, Tape, or Types)

```
‡‡JOB 5

‡‡SPS 5

*LIST TYPEWRITER
*STORE CORE IMAGE
*NAME SRTPCH


01010           DORG 19000                       19000
01020 START     TF   PUNCH+6,2690                19000 26 19138 0̲2690
01030           AM   PUNCH+6,1                   19012 11 19138 0̄0001
01010           TF   *+30,PUNCH+6                19024 26 19054 1̄9138
01020           AM   *+18,160                    19036 11 19054 0̄0160
01030           TF   ,RM                         19048 26 00000 19152
01031           BNC1 *+36                        19060 47 19096 00100
01032           RCTY                             19072 34 00000 00102
01033           WATY PUNCH+6,,6                  19084 39 19138̄ 00100
01034           BNC2 *+24                        19096 47 19120 00200
01035           WAPT PUNCH+6,,6                  19108 39 19138̄ 00200
01036           BNC3 *+24                        19120 47 19144 00300
01040 PUNCH     WACD                             19132 39 00000 00400
01050           B7   2836                        19144 49 0283̲6 00000
01040 RM        DC   2,@                         19152 00002 0̄‡
01060           DEND START                       19000

END OF ASSEMBLY.
19154 CORE POSITIONS REQUIRED
00016 STATEMENTS PROCESSED


DK LOADED SRTPCH 0̄175 10̄51770̄021̄90001̄9000‡


END OF JOB
```

Sense Switch Settings

1 On to Type
2 On to Punch Tape
3 On to Punch Cards

(At least one switch must be on to get any results.)

## APPENDIX B

### Example 1

### Sort KWIC Index Cards

These cards are to be sorted by 25 columns, cc 37-61.

Determine maximum number of cards and disk sector addresses.

Tag size = (25 x 2) + 4 = 54 digits per tag

$$X = \frac{5000 - 54}{54} = 91 \text{ tags per quarter cylinder}$$

$$N = \frac{\text{No. of Records}}{4 \times 91} = \frac{\text{No. of Records}}{364}$$

Tag File Work Area = 2 x N

$$\text{Input Area} = (\text{No. of Records} + 1) \times \frac{160}{20000} = (\text{No. of Records} + 1) \times .008$$

160 = digits per input record (80 cc x 2)

20000 = digits per cylinder

| No. of Records | Tag File, N | Tag File Work Area, 2N | Input Area | Total |
|---|---|---|---|---|
| 999 | 3 (2.75) | 6 | 8 | 14 |
| 1499 | 5 (4.12) | 10 | 12 | 22 |
| 1999 | 6 (5.5) | 12 | 16 | 28 |

If the tag file work area of 10 cylinders were filled, the input area required would be 5 x 364 x .008 = 14.6 or 15 cylinders. This exceeds the 24 cylinders available in the work area. The number of records which can be sorted is, therefore, what will fit into a 14-cylinder input area.

Fourteen divided by .008 gives 1750 including the 0-RM-RM. The number of KWIC Index Cards which can be sorted in the Monitor work area is, then, 1749. To check our calculations,

Tag size = (25 x 2) + 4 = 54 digits per tag

$$X = \frac{5000 - 54}{54} = 91 \text{ tags per quarter cylinder}$$

$$N = \frac{1750}{4 \times 91} = 4.8 \text{ or } 5 \text{ cylinders per tag file}$$

Tag file work area = 10 cylinders

Input area = 1750 x 160/20000 = 14 cylinders

Total space required = 10 + 14 = 24 cylinders

The disk sector address of the input area is specified in Control Card 3 as 100000, the beginning of the work area; the disk sector address of the tag file work area is specified as 102800, fourteen cylinders higher. The complete control cards for the application is as follows:

Control Card 1

| cc | Contents |
|---|---|
| 1 | J |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5-8 | 0160 |
| 10 | 4 |
| 14 | 0 |
| 20 | 0 |
| 21-25 | 19000 |
| 30-33 | 0283* |
| 34 | 0 |
| 38 | 0 |
| 80 | 1 |

Control Card 2

| cc | Contents |
|---|---|
| 1-4 | 0073 |
| 5-7 | 050 |
| 71, 72 | 01 |
| 80 | 2 |

Control Card 3

| cc | Contents |
|---|---|
| 1-6 | 100000 |
| 7 | 0 |
| 15-20 | 100000 |
| 21 | 0 |
| 22-27 | 102800 |
| 28 | 0 |
| 29 | 0 |
| 30 | 0 |
| 31 | 0 |
| 32 | 0 |
| 33 | 1 |
| 34 | 0 |
| 35 | 0 |
| 80 | 3 |

*Unique to our case. This is where Monitor loaded the SRTPCH program.

## APPENDIX B

### Example 2

### Sort Test Cards

This application consists of sorting by columns, 11, 12, 13, 7, 8, 9, 5, 6, 30, 31, 32 in descending order of significance. Again we wish to determine the maximum number of cards which can be accommodated by the Monitor work area and the addresses involved.

In this example, we have four control fields to be specified in Control Card 2. The most significant field (last sort) is a three-column field consisting of cc 11-13. The next to last field to be sorted on is cc 7-9; the second field is cc 5-6, and the first sort is to be on cc 30-32. The calculations are as follows, assuming the maximum number of records to be 9999.

$$\text{Tag size} = (11 \times 2) + 4 = 26 \text{ digits per tag}$$

$$X = \frac{5000 - 26}{26} = 191 \text{ tags per quarter cylinder}$$

$$N = \frac{\text{No. of Records}}{4 \times 191} = \frac{\text{No. of Records}}{764}$$

$$\text{Tag File Work Area} = 2 \times N$$

$$\text{Input Area} = (\text{No. of Records} + 1) \times \frac{160}{20000} = (\text{No. of Records} + 1) \times .008$$

160 = digits per input record (80 cc x 2)

20000 = digits per cylinder

| No. of Records | Tag File, N | Tag File Work Area, 2N | Input Area | Total |
|---|---|---|---|---|
| 1499 | 2 | 4 | 12 | 16 |
| 1999 | 3 | 6 | 16 | 22 |
| 2499 | 4 | 8 | 20 | 28 |

If the tag file work area of 6 cylinders were filled, the input area required would be 3 x 764 x .008 = 18.4 or 19 cylinders. This exceeds the 24 cylinders available. The maximum number of records which can be sorted, then, is the number which will fit into an 18-cylinder input area. Eighteen divided by .008 gives 2250 which includes the 0-RM-RM. The number of test cards which can be sorted in the Monitor work area is 2249. Now to double check.

Tag size = (11 x 2) + 4 = 26 digits per tag

$$X = \frac{5000 - 26}{26} = 191 \text{ tags per quarter cylinder}$$

$$N = \frac{2250}{4 \times 191} = 2.95 \text{ or } 3 \text{ cylinders per tag file}$$

Tag File Work Area = 6 cylinders

Input Area = 2250 x 160/20000 = 18 cylinders

Total Space Required = 6 + 18 = 24 cylinders

Using a disk sector address of 100000 for the input (and output) area, the address of the tag file work area would be 103600. The three control cards for this application are as follows:

Control Card 1

| cc | Contents |
|---|---|
| 1 | J |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5-8 | 0160 |
| 10 | 4 |
| 14 | 0 |
| 20 | 0 |
| 21-25 | 19000 |
| 30-33 | 0283* |
| 34 | 0 |
| 38 | 0 |
| 80 | 1 |

Control Card 2

| cc | Contents |
|---|---|
| 1-4 | 0021 |
| 5-7 | 006 |
| 8-11 | 0013 |
| 12-14 | 006 |
| 15-18 | 0009 |
| 19-21 | 004 |
| 22-25 | 0059 |
| 26-28 | 006 |
| 71-72 | 04 |
| 80 | 2 |

Control Card 3

| cc | Contents |
|---|---|
| 1-6 | 100000 |
| 7 | 0 |
| 15-20 | 100000 |
| 21 | 0 |
| 22-27 | 103600 |
| 28 | 0 |
| 29 | 0 |
| 30 | 0 |
| 31 | 0 |
| 32 | 0 |
| 33 | 1 |
| 34 | 0 |
| 35 | 0 |
| 80 | 3 |

*Unique in our case, the DIM number of SRTPCH our output program called by Phase 4.

## ABSTRACT

TITLE MONITOR I SYSTEM PROGRAM PACKER

AUTHOR Jack B. Watson, Texas Gulf Sulphur Co. #3275

DESCRIPTION The MONITOR I PROGRAM PACKER is a Series of six programs that are designed to pack the cylinders under Monitor in which programs and/or data is stored for greater utilization of the available areas on the disk pack. There is no reassignment of DIM numbers or programs which are file protected and/or permanently assigned.

METHOD The DIM entries are sorted in descending order by sector length, and then the programs are relocated on disk beginning with cylinder 25 in the order in which their DIM entries were sorted. A new Sequential Program Table is then generated from another sort of the DIM Table in cylinder order.

RESTRICTIONS This set of programs was written for a 1620 Monitor I System which had no modifications to the original system layout.

EQUIPMENT IBM 1620-20K, 2 - 1311's, 1443, additional instructions and indirect addressing.

PROGRAMS Six programs, source language SPS IID; the last program is a special table dump on the 1443 for verification, which is optional. All card decks are in system output format and operate under Supervisor Control.

TIME Process time for all programs is approximately one-half hour.

# INTRODUCTION

The purpose of this report is to describe the function, logic, and operating procedure of this set of programs.

The Monitor I System does not provide a simple or easy method for maintaining the greatest possible available sectors for storing User programs or data. Over a period of time, program additions and deletions tend to leave many gaps between the programs or data that Users store under Supervisor Control. Depending on the User's applications and requirements, a real problem can exist if it is necessary to maintain two Monitor disk packs or revert to loading programs via cards. This becomes more evident if many of the programs utilize Call-Links for jobs too large for one program and where available machine time is already at a minimum.

The set of programs described in this report will not eliminate the problem, but will provide Monitor packing ability until there is no significant space left on the Monitor I disk pack for User storage.

# PROGRAM DESCRIPTION

A PHASE I

A TRACK MODE DISK DUPLICATOR Program is used to copy the Monitor I Disk Pack on Drive "O" to another Disk Pack on Drive "1".

B PHASE II

This phase is a DIM TABLE SORT Program which first will read the DIM Table into core and put the DIM Number in positions 16-19 of its corresponding DIM ENTRY and then output the Table to Drive "O" address 200. Next, it will sort the DIM ENTRIES in descending order by sector count and output the resulting table to Drive "O" address 00000.

C PHASE III

This Program will utilize the modified and sorted DIM Table Entries stored on Drive "O" by Phase II and the duplicated Monitor Pack on Drive "1" to pack the programs stored under Monitor. It first notes the sector length and address of the program from the sorted DIM ENTRIES stored on Drive "O", gets the program off Drive "1", searches an In-Core Availability Table for the first available space, up dates the original DIM TABLE on Drive "O" sector address 4800, and then relocates the program accordingly. No Monitor I routine is relocated nor is any User program which is permanently assigned or file protected. There is no restriction on the length of a User's program.

D PHASE IV

This program sorts the DIM TABLE with its corresponding DIM number into cylinder order and outputs the resulting table on Drive "O" sector address 00000 as input to Phase V.

E PHASE V

A new Sequential Program Table is generated utilizing the DIM Table sort of Phase IV and is written over the previous table from sector address 19801-19880 on Drive "O". This program completes the packing of the Monitor I Disk Pack on Drive "O".

F PHASE VI

This program prints a special formated dump of the DIM TABLE, EQUIVALENCE TABLE, and Sequential Program Table on a 1443 on-line printer for verification. This is an optional part of the System and is not required for Packing the Monitor Disk. It is the only program utilizing the 1443 Printer. A card output program is optional.

# PROCEDURE

PHASE I

Use the Phase I program deck to duplicate the Monitor pack on Drive "O" to another pack on Drive "1". It operates under Monitor Supervisor, and a Cold Start Card should be used to load the program. The program will run about seven minutes. When the program is loaded, the message "Turn On Write Address Key" is typed out. After the Write Address Key has been turned on, push Start on the 1620 for processing. The program will end on a Halt instruction after typing the message "Turn Off Write Address Key." The Write Address Key must be turned off before entering Phase II.

PHASE II

The Phase II Sort Program should be loaded with a Cold Start Card. Phase III, IV, V, and VI can be stacked, in order, behind Phase II in the card reader hopper. A "JOB" card and an "XEQ" card with a 5 punched in cc 27 are required header cards preceeding each program. An "END OF JOB" card should follow each program.

Phase II execution time is approximately 8 minutes. No operator or error messages are required and the program ends on a Call EXIT.

PHASE III

The Phase III program is loaded under upervisor Control at the completion of Phase II. Execution time is approximately five minutes.

Phase III begins execution with the message "SW1 on to Use DIM 170". If Switch 1 is turned on, all programs with DIM numbers less than 170 will not be relocated. This encompasses all Monitor I System Programs since DIM 170 is the first available to the User. If switch 1 is off, the message "TYPE DIM NO, XXXX" is typed out. You may enter a 4-digit DIM Number greater than 170 where you wish program packing to begin and press the R-S key for execution. No other messages are required. The program ends with a CALL EXIT to Monitor Supervisor for loading Phase IV.

PHASE IV

The Phase IV sort program is loaded at the completion of Phase III. Execution time is approximately eight minutes. No operator messages are required; and, upon completion of the DIM TABLE SORT, control is returned to Monitor Supervisor.

PHASE V

The Phase V Sequential Program Table generator is loaded under Monitor Supervisor at the completion of Phase IV. A sequence check is performed on the sorted DIM Table Entries; and, if the Message "DIM Table Out Of Sector Sequence" is typed out, return to Phase IV and begin execution from that point. Execution time is approximately one minute; and, upon completion, control is returned to Monitor Supervisor.

PHASE VI

The Phase VI program is loaded by Monitor Supervisor. Execution begins with the program name being typed out and then the message "Enter Beginning DIM No., 3 Digits, SW4 ON IF ERROR" is typed. A three-digit DIM number must be entered and the program will begin processing from that point. This is the only program which utilized a 1443 on-line printer. If an error is made while typing the DIM number, turn switch 4 on and press R-S Key, and the program will start over. Switch 4 must be turned off to continue. For four hundred DIM ENTRIES, execution time is approximately five minutes.

# ANOTHER DISK PACKER AVAILABLE

The Engineering Computing Laboratory of the University of Wisconsin, (User 3155) announces that they have in the library a Disk Packer for Monitor I which requires only one disk drive and which requires about 10 minutes to repack the entire disk.

Library number is 1.6.137.

# THE RIT PRE-COMPILER

by

Frederick R. Henderson
Director, Computer Center
Rochester Institute of Technology
Rochester, New York 14608

(1620 User # 1393)

Presented at
1620 Users Group Joint Meeting
Americana Hotel, New York, New York
October 8, 1965

# THE RIT PRE-COMPILER

by Frederick R. Henderson (# 1393)

## ABSTRACT

Most Fortran Compilers for the IBM 1620 lack adequate diagnostics for beginning students, and the use of a Pre-Compiler is recommended. For installations with only 20K storage, the IBM Pre-Compiler is the best that is available, and it works well with the Fortran with Format Compiler.

Many schools with 20K, however, are now using PDQ Fortran, or if they have a 1311 Disk, are using Fortran II-D. Both of these compilers have added language facilities not available in Fortran with Format, and the IBM Pre-Compiler prints out too many spurious error messages when used with PDQ or Fortran II-D. To help our new students in debugging their programs, we have modified the IBM Pre-Compiler to make it more useful with PDQ Fortran and Fortran II-D.

There are presently three card versions of the RIT Pre-Compiler. The first is for use with PDQ Fortran; the second is for Fortran II-D without a printer; the third is for Fortran II-D with a 1443 Printer. These are not completely compatible with their respective compilers, but they are more useful than the IBM Pre-Compiler for beginning students.

## SUMMARY OF CHANGES INCORPORATED IN RIT PRE-COMPILER

### Changes applicable to PDQ and II-D Fortran

1. One continuation card allowed on I/O and FORMAT statements.
2. Undefined variables in statements like N = N + 1 detected.
3. "A" type FORMAT accepted (also "D" for PDQ).
4. "IF (SENSE SWITCH 9)" accepted.
5. All "C" Comment cards printed regardless of switch settings.

### Additional Changes under Monitor I (PR-025)

6. I/O statements containing implied DO-loops accepted.
7. "CALL EXIT" recognized as valid statement.
8. Program called by "‡‡PCOM" Control Card and on completion of job, branches back to "Moncal".
9. Program switch 1 only used (ON to print all statements).
10. "*" cards ahead of source deck and all cards after "END ignored.

### Further changes under Monitor I with Printer (PR-033A)

11. All output on printer except "‡‡" cards.
12. "*" Fortran Control Cards printed but not checked.

THE RIT PRE-COMPILER
by Frederick R. Henderson (# 1393)

The IBM Pre-Compiler (FO-006) was designed for use with Fortran with Format on an IBM 1620 with only 20K memory and no special features. It works well, typing out all statements in which errors are detected and giving an appropriate error message. And it is oriented toward the beginning student. In contrast, most compilers contain limited diagnostics, and these are oriented toward the experienced programmer. For example, Fortran II-D intentionally does not detect most undefined variables; this is a very common error with beginning students. Also most compilers do not type out the erroneous statements; they merely give a cryptic statement reference which beginners find difficult to interpret.

Many installations with 20K memory are now using PDQ Fortran, or if they have a 1311 Disk Storage Drive, they are using Fortran II-D. Both of these compilers have added language facilities not available in Fortran with Format. Some of these are useful only to an advanced programmer, but the provisions for continuation cards, "A" and "D' type format statements, and input/output in matrix form (II-D only) are of immediate use to beginners. If the IBM Pre-Compiler is used with these compilers, it prints so many spurious error messages that the results are of little value in detecting real errors.

Also the IBM Pre-Compiler fails to detect one very common beginning error; namely, an undefined variable in a statement like N = N + 1. Since failure to initialize is a very frequent beginning error, it would be helpful if the Pre-Compiler could detect this type of mistake.

While attending an NSF sponsored Summer Computer Conference at Seton Hall University in June, 1965, the author undertook the task of modifying the IBM Pre-Compiler and succeeded in providing for one continuation card on FORMAT, READ, PRINT, PUNCH, and TYPE statements. "A" and "D" formats, and an "IF (SENSE SWITCH 9)" instruction were also included. Subsequently additional work at the Rochester Institute of Technology Computer Center resulted in further improvements and modifications to adapt the program to a 1311 Disk and to a 1443 Printer.

There are, therefore, three card versions of the RIT Pre-Compiler: one for PDQ Fortran, one for Fortran II-D with a Disk only, and one for Fortran II-D with a Disk and a Printer. Details as to the changes incorporated in each one are summarized below.

The principal difficulty encountered in modifying the IBM Pre-Compiler for PDQ Fortran was the fact that the program required almost 20K of core; there was, therefore, no place to put the desired modifications. However, our experience was that beginning students almost never used all of the space allotted to the symbol table, and it was felt that we could recapture perhaps 1,000 cores from the symbol table. Also it turned out that the cores from 402 to 1207 were used only to initialize the program the first time it was read into core so that we could gain 805 cores here by overlaying the original program. Also we gained 56 cores (enough to take care of the "A" format) by eliminating the "Clear Beta" routine at 14836 since this is not needed for card input.

The first major modification undertaken was to provide for the processing of one continuation card. To do this it is necessary to reserve an additional area in which to store the next card before processing the first card; this storage is located from 00900 to 01059 with a record mark in 01061. If a digit in column 6 of the next card indicates that it is a continuation card, the appropriate changes are then made in both card images so that they appear to the Pre-Compiler to be separate cards and are so processed. This was done instead of trying to combine the two cards into one core image in order to save core storage space and to avoid having to re-write the entire Pre-Compiler. The first card must end with a comma or slash as specified in PDQ Fortran.

The other major modification for use with PDQ Fortran was designed to detect an undefined variable in a statement like A = A + 1. Our procedure here is to take the symbol A (for example) on the left of the equal sign, out of the symbol table temporarily while the Pre-Compiler is analyzing the expression on the right of the equal sign and then to put it back before reading the next card.

Three minor changes were made as follows. To render the "A" and "D" type formats acceptable, these are simply changed to "I" type in core image and then processed. Similarly "IF (SENSE SWITCH 9)" is handled by changing the 9 to a 1 before it is processed. Finally, all "C" Comment cards are printed regardless of switch settings.

If a 1311 Disk is available, there is, of course, no problem of finding storage space for changes; these can be put on the disk and called as overlays when needed. This should make it possible to provide for more than one continuation card if desired. This, however, has not been done; instead the PDQ version was utilized simply to save programming time.

It is obviously desirable if possible to put the Pre-Compiler under Monitor I for Disk operation, and this is done by substituting a Monitor Control Card, "‡‡PCOM", for the "‡‡TYPE" card which we never used. The Pre-Compiler is then stored on the disk and called when needed. When processing of a source program has been completed, instead of halting, the Pre-Compiler branches back to "Moncal" and "END of JOB" is typed out.

Fortran II-D permits the use of Matrix Input/Output statements, and if the Pre-Compiler is to process such a statement containing an implied DO-loop, the DO-loop must be deleted. This is done by first checking all Input/Output statements for equal signs. If one is found, an overlay is called in from the disk which converts the original core image to one of standard form. For example: READ 7, (A(N), N = 1, 9) becomes in core image READ 7, A(1) and is so processed. This procedure does not check the subscript to see if it matches the DO index; perhaps this can be included at a later date. Indirect addressing has been used in this overlay, but not elsewhere.

Several other minor modifications were made as follows. A check for "CALL EXIT" is made just before the regular check for "CONTINUE" since both statements begin with "C" and have eight letters. In order to operate routinely under Monitor I, all program switch options except 1 (to print out all statements) have been eliminated, but these can be very easily activated again if desired. The statement "ENTER SOURCE PROGRAM THEN PUSH START" has also been deleted. Also any "*" control cards ahead of the source deck and any cards after the "END" card are simply ignored by the Pre-Compiler. Lastly, in the event of a check stop, it is possible to branch manually to 17600, transfer control back to "Moncal", and proceed with the next job.

For installations which have a 1443 Printer in addition to a 1311 Disk, the program has been modified further to transfer all output except the Monitor Control Cards from the typewriter to the Printer. Fortran "*" Control Cards ahead of the source program are also printed but not checked for validity. This version, of course, operates under the Printer version of Monitor I (PR-033A) instead of the standard version (PR-025).

One minor drawback has resulted from these modifications; sometimes the error messages indicated do not seem to make much sense. This is because in some instances the core image has been changed before being processed by the Pre-Compiler. However, it is the original source statement that is printed out, and usually the error is fairly obvious even though it isn't exactly the one designated.

A preliminary version of this program was used successfully this past summer in an NSF Summer Institute in Computer Programming at the Rochester Institute of Technology. The final version, however, has not yet been extensively tested, and the author will greatly appreciate learning of any "bugs" which develop in actual use.

# AN OPERATING SYSTEM FOR THE 1620/1443 CONFIGURATION

E. J. Orth, Jr.
W. A. Norton
Southern Services, Inc.
Birmingham, Alabama

presented at the
1620 Users Group Meeting
October 5-8, 1965
Americana Hotel
New York City

## ABSTRACT

An operating system has been designed incorporating certain of the features of the disk-oriented monitor system. The operating system permits processing of a stacked file consisting of many groups of program-data chained together and requiring no operator intervention between programs. A FORTRAN error message will abort a run and automatically load the next program. The operator has complete control over the operating system through console switches 2 and 3. Switch 2 up will terminate processing on a job and cause cards to be passed through the 1622 until the next program is reached. Switch 3 up bypasses the automatic abort feature on FORTRAN errors. The system has been implemented in FORTRAN II on a 60K 1620-II/1443-I system at no core cost. The FORTRAN II implementation on a 1620-I would cost approximately 200 locations. SPS implementation costs are variable but small. Compilation/assembly procedures are not included at present. The operating system is but one aspect of Southern Services' total approach to automating routine, repetitive work.

## ENVIRONMENT

The Southern Services Computer Center is a job shop operation for engineering problem solving. The average problem runs in about 10 minutes on the 1620-II. The workload is heavy: For instance, during September 1965 computer clock time amounted to 203 hours.

In such an environment, an operating system has two advantages:

1. Lost machine time between jobs is minimized, and
2. Data handling errors are minimized since all data files are prepared off-line, away from the pressure of on-line deck shuffling.

## SOUTHERN SERVICES OPERATING SYSTEM (SSOS)

Requirements for a basic operating system may be summarized as follows:

A. Programming, operation, hardware

1. The output device must provide a file of infinite length.
2. Programs must be console switch independent.

It is generally desirable to program certain error checking routines into each program. If catastrophic data errors occur the programmer may initiate search for an EOJ card at source program level.

## OBJECT-TIME TYPEWRITER MONITOR

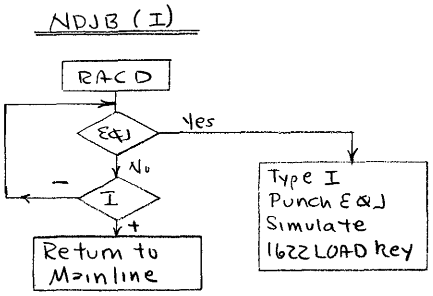
Each program will skip a line and type "GO NNNN-" before reading the BOJ card. The monitor message is completed on normal exit by a repetition of the program number. An abnormal exit due to any of the three abort procedures is indicated either by the repeated program number with a flag over the low order position, or the complete absence of the repeated program number.

## MODIFICATIONS TO FORTRAN II

The prime mover in the operating system is a relocatable FORTRAN II Library routine called NDJB. NDJB has two functions:

1. Scan all input cards for an EOJ card, and
2. Allow termination of a run under program control.

As shown in the flow chart, NDJB reads a card into the FORTRAN input buffer. If this card is an EOJ card the very next card is assumed to begin a new program. If it is not an EOJ card and if the argument is positive control is returned to the mainline program. If the next card is not an EOJ card and if the argument is negative, cards are passed through the reader until an EOJ card is encountered.



Once a card has been read into the FORTRAN input buffer by NDJB the buffer must be scanned under the appropriate format. The FORTRAN II ACCEPT TAPE routine has been modified by killing that section which initializes the I/O buffer and reads a tape record, so that the buffer may be reread by the statement ACCEPT TAPE n, List. The buffer may be reread by as many ACCEPT TAPE statements with differing formats as necessary.

Advantage may be taken of certain features of the 1620-II, thus allowing all coding required by the operator abort and FORTRAN error message abort procedures to be squeezed into the 11K subroutine deck. The specific features are transmit floating (FRFAC and TOFAC) and add hardware (no add tables).

## IMPLEMENTATION USING SPS III

Certain macros were originally planned to simplify use of SSOS in SPS III. This has been abandoned for the following considerations:

1. Relatively few programs are coded in SPS III.
2. SPS III assembly procedures would be complicated since all assemblies would require loading the subroutine deck to pull out the macros.
3. It is very simple to code the operating system logic in-line.

## AVAILABILITY

Those interested in a detailed description of the operating system may obtain such by writing E. J. Orth, Jr., Southern Services, Inc., PO Box 2641,

PENNSYLVANIA TRANSFORMER DIVISION
McGraw-Edison Company

# FORTRAN LABEL INDEXER

by

Lawrence S. Powell

For presentation at the Fall, 1965, 1620 Users Group Meeting

October 6 - 8

New York, New York

# FORTRAN LABEL INDEXER

## General Description

A Fortran label in the FLI is a fixed point number, floating point number, statement number, or variable. The FLI provides an ordered listing (sequential and alphabetic) of each label used in a Fortran source program. The fixed and floating point numbers have the card sequence number of the cards on which each is used to the right of the label. The statement numbers or variables have the card sequence number of one definition to the left of the label and any other definitions are listed on a separate line designated by an asterisk (*). All references of a statement number or variable appear to the right of the label. Statement numbers or variables may appear without either a definition or a reference.

## Background and Uses

The three main reasons for the development of the FLI were (a) to allow detection of inefficiences or non-critical errors in a Fortran program not detected by a pre-compiler; (b) to serve as an aid in the debugging of a program; (c) to serve as an aid for modifying or adding to a present program.

A program can be compiled and run without error even though it contains statements that are never referenced or variables that are developed but not used. Both of these conditions waste core and increase compilation time and the second increases execution time. The lack of a reference for a statement number or a variable points out both conditions at a glance using the FLI.

The identification of the card location of each statement number makes following the flow of a program much easier. The identification of each definition and reference of a variable lets a programmer make sure that the proper calculation is used in conjunction with each use.

The FLI, in providing a list of all statement numbers and variables, shows the programmer what cannot be used in any addition to the program and by using a listing of the sequenced deck, he can tell what each variable represents and where it is used.

## Some Specifics

The FLI is written for use with IBM Fortran/Format, PDQ Fortran, and PDQ C4D. It will not handle the six character variable of Fortran II. The FLI can use the card sequence numbers already on the source deck or develop new sequence numbers and output the new sequenced deck either on the typewriter or on cards. A provision for large programs that overflow the allotted symbol table space is provided to break the source deck and output two separate indexes. The FLI should handle any program that can be compiled without overlay on the users machine and in one case has processed, on a 40K machine, a program that would use 89K if compiled without overlay. The FLI takes about 8K of storage itself with each label taking 10+5N digits where N = number of uses. Arrays are treated the same as unsubscripted variables.

At present, output is either by cards or typewriter but a printer version should be available soon after the present version. I hope to submit the FLI to the library in late October or early November.

```
C       SAMPLE PROGRAM FOR FORTRAN LABEL INDEXER                       0000
        DIMENSION A(10),B(5,5)                                         0001
      1 READ 100,(A(J),J=1,10),AC                                      0002
        DO 2 I=1,5                                                     0003
        B(1,I)=A(I)*AC                                                 0004
      2 B(2,I)=SQRT(A(I)*AC)                                           0005
        IF(AC)3,4,C                                                    0006
        AC=AC+9.                                                       0007
        DO 5 I=1,5                                                     0008
        B(3,I)=A(I+5)*2400.671                                         0009
        B(4,I)=0.                                                      0010
      5 B(5,I)=0.                                                      0011
        GO TO 6                                                        0012
      3 PRINT 102,(B(1,I),I=1,5),(B(2,I),I=1,5)                        0013
        GO TO 1                                                        0014
      4 ACT=AC-1.                                                      0015
        B(1,1)=9.                                                      0016
        GO TO 3                                                        0017
    100 FORMAT(11F6.0)                                                 0018
    101 FORMAT(3I2)                                                    0019
    102 FORMAT(10F8.2)                                                 0020
        END                                                            0021

FIX CONSTANT
          1           0016 0016 0013 0013 0013 0008 0004 0003 0002
          2           0013 0005
          3           0009
          4           0010
          5           0013 0013 0011 0009 0008 0003
         10           0002

FLOATING CONSTANT
.00000000 E 00        0011 0010
.10000000 E 01        0015
.90000000 E 01        0016 0007
.24006710 E 04        0009

STATEMENT NUMBERS
0002          1       0014
0005          2       0003
0013          3       0017 0006
0015          4       0006
0011          5       0008
              6       0012
0018        100       0002
0019        101
0020        102       0013

VARIABLES
0002       A          0009 0005 0004
0007       AC         0015 0007 0006 0005 0004 0002
0015       ACT
0016       B          0013 0013
                     *0011 0010 0009 0005 0004
0013       I          0013 0013 0011 0010 0009 0009 0005 0005 0004 0004
                     *0013 0008 0003
0002       J          0002
           SQRT       0005
```

Computer Center
RANDOLPH-MACON COLLEGE
Ashland, Virginia 23005

# PLOT SUBROUTINE FOR PDQ FORTRAN

Richard E. Grove

A paper given before the 1620 Users Group Joint Eastern-Midwestern Meeting, Americana Hotel, New York, October 6, 7, 8, 1965.

A relocatable subroutine known as PLOT has been added to the Randolph-Macon Computer Center PDQ FORTRAN system. This subroutine allows the programmer to put arbitrary single alphanumeric characters in any desired card column for the purposes of generating graphs of one or more variables or the plotting of "contour maps". The form of the statement is

DUM = PLOT (ARG)

where DUM is a dummy variable used only because this plot techniques utilizes the relocatable subroutine device. The variable on the left of the = sign (DUM in this case) should not be used in any arithmetic or logical operations since the subroutine generates no numerical value for this dummy variable. Any valid floating point variable name may be used instead of DUM. The argument, ARG, in the above statement may be any valid floating point constant, floating point variable, or floating point expression within the restrictions below:

1. When ARG is positive, it should take on numerical values of the form XXXX. (four digits, the leftmost non-zero, followed by a decimal point).

2. When ARG is positive and of the form specified by (1), two pieces of information are contained within this notation: (a) The two leftmost digits specify the alphanumeric character which is to be used to plot the point and the two rightmost digits specify the card column into which this character is to be placed. The character codes are shown in a table below.

Example 1. GARB = PLOT(4123.) will place the character A (41 in two digit alphameric notation) column 23 of the PLOT output card.

Example 2: X = 7678.
Y = PLOT(X) will place a numeric 6 in card column 78.

3. Perhaps the easiest way to use the subroutine is to develop, by appropriate computation and scaling, the number of the card column as a floating point integer between 1 and 80 (call this CC), and write

DUM = PLOT (CC + 4100.)

if an A is desired in the computed card column, CC. The codes below are listed with this point of view in mind. That is, it is assumed the code below will be added to the floating point variable expressing the desired card column.

| Character | Code | Character | Code |
|---|---|---|---|
| + | 1000. | O | 5600. |
| $ | 1300. | P | 5700. |
| * | 1400. | Q | 5800. |
| - | 2000. | R | 5900. |
| / | 2100. | S | 6200. |
| , | 2300. | T | 6300. |
| ( | 2400. | U | 6400. |
| = | 3300. | V | 6500. |
| A | 4100. | W | 6600. |
| B | 4200. | X | 6700. |
| C | 4300. | Y | 6800. |
| D | 4400. | Z | 6900. |
| E | 4500. | 0 | 7000. |
| F | 4600. | 1 | 7100. |
| G | 4700. | 2 | 7200. |
| H | 4800. | 3 | 7300. |
| I | 4900. | 4 | 7400. |
| J | 5100. | 5 | 7500. |
| K | 5200. | 6 | 7600. |
| L | 5300. | 7 | 7700. |
| M | 5400. | 8 | 7800. |
| N | 5500. | 9 | 7900. |

Note that several characters may not be used as plotting characters. In particular the period and the right parenthesis are excluded because their two-digit alphanumeric codes begin with zero. The prime symbol (') is excluded because it plays a special role as described later.

4. Since there are eighty columns in a card, numbered from 1 to 80, a request to put a character in card column zero or in a card column higher than 80 is

an error condition. A new error message

P ERR

has been provided (on the typewriter) to indicate the requested card column is zero or greater than 80. This invalid request is indicated by the error message but is otherwise ignored and processing continues.

5. If the programmer should specify a character code which does not correspond to a valid character, a WRITE CHECK condition will exist.

6. The programmer may use as many sequential requests for the PLOT subroutine as may be desired to place many characters in the card image. Of course, if several requests reference the same card column, only the last character requested will appear in that card column when the card is punched.

7. After the card image has been developed, the card may be punched using a negative floating point argument for the PLOT subroutine. Thus

   Z = PLOT(-1.)

   will force the punching of the card image as it then exists in core and will cause the output area to be cleared to alphameric blanks (zeros in all 160 core locations).

## 407 LISTING OF THE PLOT

A special 407 control board has been wired to allow the listing of 16 columns of numeric information from one card and 80 columns of plotting characters (or blanks) from a second card on the same printed line. Our 407 (Model E8) has only 96 print positions. It may be desired that the actual numerical values of the plotted data be listed with the plot. This may be done by generating a card under the usual PDQ FORMAT control such that a prime symbol (4, 8 punch in the card) appears in card column 1 of the output card and up to 16 positions of numerical information appear in cc 2-17. For example, you may desire to list an X and Y value to the left of the plot of the Y value by the following program fragment:

```
        PUNCH 73, X, Y
73      FORMAT (1H',F7.2,2X,F7.2)
           .
           .      In this portion of the program, develop the value of CC,
           .      the card column into which the plotted point associated
           .      with Y is to be plotted.
           .
        DUM = PLOT (7600.+CC)
        DUM = PLOT ( -1.)
```

Two cards will be punched. The first contains a prime symbol (') in cc 1 and two seven digit numeric fields separated by two blanks. The second card will contain the plot symbol, numeric 6, in its appropriate card column. The special 407 plotting control board will cause these two cards to be listed on the same printed line, perhaps as follows:

$^{b}123.45^{bbb}987.65$                                        6

Since 80 print positions are required for the card generated by the PLOT subroutine, only 16 positions are available for printing of numeric information and this must appear only in cc 2-17 of the card in which the prime symbol (') appear in cc 1. The prime symbol in cc 1 is needed for control purposes on the 407.

The card bearing numeric information should be punched immediately before its associated plot card is punched. There is no requirement that the numeric card be punched; there is no requirement that a numeric card be punched for every plot card. Thus, it would be possible to give the numeric value for, say, every tenth point plotted.

The listing of the 1620/1710 SPS uncondensed object deck for this subroutine is shown on the next page. Full instructions for incorporating the condensed object deck into the PDQ processor are contained in the documentation from the Users Group Library.

```
0500                                DORG 5000
05000 M4 05110 19809                BNF  PUT,19809,0
05012 L9 05259 004-0      B         WACD OUT,,010
05024 J0 05054 -5417                TFM  *+30,OUT+158,017
05036 J6 05023 000-0                TFM  COUNT,0,010
05048 16 00000 -0000      AGAIN     TFM  ,,7
05060 J2 05054 000-5                SM   *-6,5,010
05072 J1 05023 000-1                AM   COUNT,1,010
05084 J4 05023 000L2                CM   COUNT,32,010
05096 M7 05048 01200                BNE  AGAIN,,0
05108 42 00000 00000                BB
0511                                DORG *-9
05110 J0 05133 -5257      PUT       TFM  LOC,OUT-2,017
05122 32 19802 00000      A         SF   19802
05134 32 19804 00000                SF   19804
05146 14 19805 000Q0                CM   19805,80,10
05158 M6 05232 01100                BP   ERR,,0
05170 14 19805 000-1                CM   19805,1,10
05182 M7 05232 01300                BN   ERR,,0
05194 13 19805 000-2                MM   19805,2,10
05206 K1 05133 00099                A    LOC,99,0
05218 K6 0513L 19803                TF   LOC,19803,06
05230 42 00000 00000                BB
05232                               DORG *-9
05232 34 00000 00102      ERR       RCTY
05244 L9 05419 00100                WATY MESS1,,0
05256 42 00000 00000                BB
05258                               DORG *-9
05259    00080            OUT       DAS  80
05419    00006            MESS1     DAC  6,P ERR'
05133    00000            LOC       DS   ,A+11
05023    00000            COUNT     DS   ,B+11
                                    DEND
```

# EDIT LIBRARY SUBROUTINE FOR SPS II-D

by

Robert P. Bair
Freas-Rooke Computing Center
Bucknell University
Lewisburg, Pennsylvania
October 4, 1965

1620 Users Group

Eastern-Midwestern Joint Conference

Americana Hotel

New York City, New York

October 8, 1965

Editing numbers in preparation for output is necessarv in SPS-written programs to (1) convert numeric fields to alphameric, (2) make the numbers meaningful by inserting decimal points, minus signs for negative numbers, and dollar signs, and (3) to make the numbers easier to read by inserting commas, dashes, or other punctuation and eliminating unnecessary leading zeros. Any of these operations may be done by a small routine within a program, but this EDIT subroutine performs a mixture of any of these features with the least possible effort on the part of the programmer.

In programs where the cards have to be read alphamerically, the data that does not enter into any calculations can be left alphameric, while other fields such as prices or quantities must be striped to numeric for processing. This subroutine accommodates both modes and is able to edit numbers stored either numerically or alphamerically.

The desired options are specified by forming an alphameric mask which contains all the punctuation that the result is to have, and is blank where the digits from the input are to be filled. An example of a mask, with its input and output, appears in Figure 1.

Dollar signs are handled differently than the other characters because they have two special requirements. (1) They are desired in the same numbers at different times, such as at the top of a page and in totals. (2) The position of a dollar sign is not fixed since it depends on the length of the number. This edit routine will place a dollar sign in the output only if the low-order digit of the mask being used is flagged. This flag can be set when a new page of output begins or totals are listed, and cleared after every output instruction. One mask can then be used exclusively, and the dollar sign does not need to be specified as a character in the mask.

It is assumed that a minus sign would always be desired to indicate a negative number, and so it is produced automatically, independent of the mask, and placed to the right of the output. This again makes the mask easier to construct and use. The minus sign is suppressed for a negative zero.

Two new OP codes, EDTN and EDTA, are defined to refer to this routine and indicate numeric or alphameric input, respectively. The macro-instruction must have three operands which give the field addresses of the mask, input, and output in that order. An example is shown in Figure 2.

The mask should be formed by a DSAC statement so that the label is assigned to the low-order digit of the mask. A flag over the high-order zone digit defines its length. The edit routine fills the digits from the input field into the output area whenever a blank is in the corresponding digit of the mask, and copies punctuation in the mask that it passes. If a zero is substituted for a blank in the mask, and if the left-most digit transferred to the output corresponds with this zero, all consecutive non-blank characters, including other zeros, in the mask to the left of that point will be included in the output. When a blank is reached in the mask to the left, transmission of characters stops. If the left-most digit in the number corresponds with a blank in the mask, transmission stops immediately. No blanks are inserted in the output. If the number of blanks and zeros in the mask is greater than the number of digits used from the input field, the unused blanks in the mask have no effect. If the number of blanks and zeros in the mask are fewer than the number of digits to be used from the input, the excess high-order digits of the input are truncated and not included in the output.

The program was written for a subroutine number of 18 (first subroutine added to SPS library), and an SPS subroutine set number of 02 (variable length). The listing shows how a subroutine with two entry points may be assembled into the Monitor system. To change the subroutine to some other number n, in set s, do the following:

Define EDTN as OP CODE n.

Define EDTA as OP CODE n + 1.

Change the ID NUMBER to 130 - 30 s + n.

(130 - 30 s is the base number).

Change the operand in the DEND statement to s n 2.

This subroutine can be used with dates, page numbers, inventory numbers, quantities, and costs. It is no harder to get a good-looking report than a sloppy one, but it makes a difference to the man who has to read it. Use EDTN in place of a transmit numeric fill, and EDTA instead of a transmit field for clear reading results.

```
ZZJOB
ZZXEQ SPSLIB
*DEFINE OP CODE
           EDTN-181
           EDTA-191
*LIST OP CODE
*ENDLIB
ZZDUP
*DELETEDTN
ZZSPS
*NAME EDTN
*ID NUMBER 0088
*LIBR
*ASSEMBLE RELOCATABLE
*STORE RELOADABLE
*LIST PRINTER
00010**********************************************************************
00020*
00030*            AN EDIT SUBROUTINE FOR SPS II-D
00040*
00050*            ROBERT P. BAIR
00060*            BUCKNELL UNIVERSITY
00070*            LEWISBURG, PA.    17837
00080*            MAY 1, 1965
00090*
00100**********************************************************************
00110*
00120*            REQUIRED HEADER
00130ORIG  DSA EDTN,EDTA
00140      DORGORIG-4
00150*
00160EDTN  TFM DECR     ,1        ,10  ,INPUT IS NUMERIC
00170      B7  GET
00180*
00190EDTA  TFM DECR     ,2        ,10  ,INPUT IS ALPHAMERIC
00200*
00210GET   AM  ADDR     ,4        ,    ,GET ADDR OF 1ST DSA IN LINKAGE
00220      TF  MASK     ,-ADDR    ,    ,SET ADDRESS OF MASK
00230      TF  SMASK    ,MASK     ,    ,SAVE ADDRESS OF MASK
00240      AM  ADDR     ,10       ,    ,GET ADDR OF 3RD DSA IN LINKAGE
00250      TF  OUT      ,-ADDR    ,    ,SET ADDRESS OF OUTPUT AREA
00260      SM  ADDR     ,5        ,    ,GET ADDR OF 2ND DSA IN LINKAGE
00270      TF  IN       ,-ADDR    ,    ,SET ADDRESS OF INPUT AREA
00280*
00290      AM  IN       ,1        ,10  ,SET IN TO ZONE DIGIT
00300      S   IN       ,DECR
00310      TFM D        ,1        ,10  ,D = NO OF DIGITS IN INPUT
00320FIND  AM  D        ,1        ,10  ,COUNT DIGITS
00330      S   IN       ,DECR     ,    ,FIND FLAG AT END OF FIELD
00340      BNF FIND     ,-IN
00350      A   IN       ,DECR     ,    ,END OF INPUT FIELD
00360      SM  IN       ,1
00370BLANKSBD  SIGN     ,-IN      ,    ,SUBT NO OF PRECEEDING ZEROS
00380      SM  D        ,1        ,10
00390      A   IN       ,DECR     ,    ,GO TO NEXT INPUT DIGIT
00400      B7  BLANKS
00410*
00420SIGN  CM  D        ,0        ,10  ,DONT CHECK FOR SIGN IF ZERO
00430      BE  QUIT
00440      TF  IN       ,-ADDR    ,    ,RESET IN
```

314

```
00450       CM  DECR     ,1        ,10  ,IS INPUT ALPHA OR NUMERIC
00460       BE  NUM
00470       SM  IN       ,1        ,    ,TEST FOR NEG ALPHA INPUT
00480       TD  FIELD    ,-IN
00490       CM  FIELD    ,77       ,10
00500       BE  GETIN    ,         ,    ,B IF NUMBER IS POSITIVE
00510NEG    AM  OUT      ,2        ,    ,NEGATIVE INPUT. SET - SIGN
00520       TFM -OUT     ,20       ,10  ,SET MINUS SIGN
00530       SM  OUT      ,2
00540       B7  GETIN
00550NUM    BNF GETIN    ,-IN      ,    ,TEST FOR NEG NUMERIC UNPUT
00560       B7  NEG      ,         ,    ,B IF NEG TO SET SIGN
00570*
00580GETIN  TF  IN       ,-ADDR    ,    ,RESET INPUT ADDRESS
00590       TDM ZERO     ,0        ,    ,SET INDICATOR
00600       TDM ENDM     ,0        ,    ,CLEAR INDICATOR
00610*               REPEAT LOOP D TIMES
00620REPET  CM  D        ,0        ,10
00630       BNP QUIT
00640       SM  D        ,1        ,10
00650NODIG  BD  DSIGN    ,ENDM     ,    ,STOP IF END OF MASK REACHED
00660*
00670       TD  FIELD+1  ,-MASK
00680       CF  FIELD+1  ,         ,    ,IGNORE FLAGS
00690       SM  MASK     ,1
00700       TD  FIELD    ,-MASK    ,    ,PUT ALPHA CHAR INTO FIELD
00710       SM  MASK     ,1        ,    ,ADVANCE TO NEXT CHAR IN MASK
00720       BNF TESTM    ,FIELD    ,    ,TESTS FOR END OF MASK
00730       TDM ENDM     ,1        ,    ,SET INDICATOR
00740       CF  FIELD
00750TESTM  TDM ZERO     ,1        ,    ,CLEAR INDICATOR
00760       CM  FIELD+1  ,700      ,9   ,TEST FOR BLANK IN MASK
00770       BE  DIGIT
00780       CM  FIELD+1  ,770      ,9   ,TEST FOR ZERO IN MASK
00790       BP  DIGIT
00800       BN  USE
00810       TDM ZERO     ,0        ,    ,ZERO IN MASK. SET INDICATOR
00820       B7  DIGIT
00830USE    SF  FIELD
00840       TF  -OUT     ,FIELD+1  ,    ,USE CHAR FROM MASK
00850       SM  OUT      ,2
00860       B7  NODIG
00870DIGIT  BNR *+20     ,-IN      ,    ,TEST FOR RECORD MARK IN INPUT
00880       B7  SIN      ,         ,    ,IGNORE RECORD MARKS
00890       TD  FIELD    ,-IN      ,    ,GET DIGIT FROM INPUT AREA
00900       TF  -OUT     ,FIELD    ,    ,PUT FILLED NUMBER IN OUTPUT
00910       SM  OUT      ,2
00920SIN    S   IN       ,DECR
00930       B7  REPET
00940*
00950QUIT   BD  DSIGN    ,ZERO     ,    ,RETURN IF LAST DIGIT IN MASK IS
00960*                                        NOT A ZERO
00970       TD  FIELD+1  ,-MASK    ,    ,LOOK FOR MORE ZEROS IN MASK
00980       CF  FIELD+1  ,         ,    ,IGNORE FLAGS
00990       SM  MASK     ,1
01000       TD  FIELD    ,-MASK
01010       CF  FIELD
01020       CM  FIELD+1  ,700      ,9   ,DETERMINE IF MASK IS BLANK
01030       BE  DSIGN
01040       SF  FIELD
```

```
01050       TF   -OUT       ,FIELD+1   ,     ,TRANSMIT CHARACTER FROM MASK
01060       SM   OUT        ,2
01070       BNF  *+20       ,-MASK     ,     ,TEST FOR END OF MASK
01080       B7   DSIGN
01090       SM   MASK       ,1
01100       B7   QUIT+12
01110*
01120DSIGN BNF END         ,          ,     ,Q OPERAND FILLED IN
01130SMASK DS               ,*
01140       TFM  -OUT       ,1300      ,8    ,SET $ AND BLANK
01150END    AM   ADDR       ,8         ,10   ,GET RETURN ADDRESS
01160       B7   -ADDR      ,          ,     ,RETURN
01170*
01180*
01190ADDR   DS               ,2375      ,     ,PICK +10
01200D      DS   2
01210FIELD DC   2,70
01220       DS   1
01230ZERO   DS   1
01240ENDM   DS   1
01250DECR   DS   2
01260MASK   DS   5
01270IN     DS   5
01280OUT    DS   5
01290       DEND02 18 2
ZZZZ
```

## EDIT

| | |
|---|---|
| MASK | b,bbb,bbb.bb |
| INPUT | $\bar{1}687234$ |
| OUTPUT | 16,872.34 |

| | |
|---|---|
| MASK | b,bbb,bbb.00 |
| INPUT | $\bar{0}\bar{1}$ |
| OUTPUT | .01- |

Fig 1

## LINKAGE

```
EDTN  M, NUM, OUT
```

where M = field address of mask

NUM = field address of numeric input

OUT = field address of output

Fig 2

```
ZZJOB
ZZSPSX 02
     *         TEST PROGRAM FOR EDIT SUBROUTINE
     *         MASK AND INPUT FIELDS ARE ENTERED FROM CARDS
     START RCTY
           RNCD CARD
           EDTN CARD+19,  CARD+39,  CARD+69
           TD  CARD+73  ,RECMK
     *         TYPE MASK, INPUT, AND OUTPUT NUMERICALLY
           WNTY CARD
           TD  CARD+21  ,RECMK
     *         TYPE MASK AND OUTPUT ALPHAMERICALLY ON NEXT LINE
           RCTY
           WATY CARD+1
           WATY CARD+43
           RCTY
           B   START
     CARD  DSS 50
           DSS 30
     RECMK DSC 1,-
           DEND START
    0023000000030000              000000
    0023000000030000              000001
    0023000000030000              000021
    0023000000030000              000321
    0023000000030000              054321
    0023000000030000              054321
    0023000000037070              000000
    0023000000037070              000001
    0023000000037070              000321
    0023000000037070              00002J
    0023000070037070              000000
    0000217070217070              050165
M1547000002000000046              J23456
    P003707070707070              001002
            00000000             J234567
      00000000000000              J23Z56
00000023000000037070               00408
          P070030000               00078
```

```
00000̅00230000000300000000000000000000̅000000000000000000000000000000000000000000

00000̅00230000000300000000000000000000̅000010000000000000000000000000000000007̅1000
   ,    .                1
00000̅00230000000300000000000000000000̅000210000000000000000000000000000007̅27̅1000
   ,    .               21
00000̅00230000000300000000000000000000̅003210000000000000000000000000007̅30̅37̅27̅1000
   ,    .             3.21
00000̅00230000000300000000000000000000̅543210000000000000000000000007̅57̅47̅30̅37̅27̅1000
   ,    .           543.21
00000̅00230000000300000000000000000006̅54321000000000000000007̅62̅37̅57̅47̅30̅37̅27̅1000
   ,    .         6,543.21
00000̅00230000000370700000000000000000̅0000000000000000000000000000000000̅37̅07̅0000
   ,    .00            .00
00000̅00230000000370700000000000000000̅000010000000000000000000000000000̅37̅07̅1000
   ,    .00            .01
00000̅00230000000370700000000000000000̅003210000000000000000000000000007̅30̅37̅27̅1000
   ,    .00           3.21
00000̅00230000000370700000000000000000̅00021̅00000000000000000000000000000̅37̅27̅1̅200
   ,    .00            .21-
00000̅00230000700370700000000000000000̅00000000000000000000000000000007̅00̅37̅07̅0000
   ,   0.00           0.00
00000̅00021707021707000000000000000000̅50165000000000000000000007̅52̅17̅07̅12̅17̅67̅5000
     /00/00        5/01/65
4̅15470000020000000460000000000000000001̅23456000000000000004̅15̅47̅1̅7̅27̅32̅07̅47̅57̅64̅6000
AMO    -   F    AM123-456F
00007̅003707070707070000000000000000000̅010020000000000000000007̅00̅37̅07̅07̅17̅07̅07̅2000
  0.000000       0.001002
000000000000000̅0000000000000000000001̅234567000000000000000000000000007̅47̅57̅67̅7000
                      4567
000000000̅0000000000000000000000000001̅23
                     12356
0̅0000002300000003707̅0000000000000000000̅0408000000000000000000000001̅3007̅40̅37̅07̅8000
   ,    .00          $ 4.08
000000000007̅070030000000000000000000000̅0078000000000000000000000000000007̅7̅8000
     00.                78
```

PRINCETON UNIVERSITY

DEPARTMENT OF AEROSPACE AND MECHANICAL SCIENCES
GUGGENHEIM LABORATORIES FOR THE AEROSPACE PROPULSION SCIENCES

October 8, 1965

# MODIFIED SP-035

by

L. Hoffman

SP-035 is an IBM S.P.S. processor for S.P.S. III (very similar to SP-029) for use on a 1620 equipped with a 1443 printer. This was originally a two-pass processor and the listings of the assembled program come out with the flagged digits as letters. I was not satisfied with this output, so I set out to make a separation of flags and digits on the listings. There was a source deck for SP-029 and I had source listings for SP-035. By changing a few cards, I was able to reproduce the source SP-035 and then modify this to give an output format as shown on the attached listing. This worked out rather well so I next decided to try a "one-pass" processor modification by reading S.P.S. statements into upper core starting at the top and working down until there could be a overlap with the symbol table which was working up in core. When, and if, this occurs, a message is typed and the S.P.S. statements are then cleared to make way for more symbol table and two-pass processing must be done. For one-pass processing, there is one restriction: there can be no record marks in the S.P.S. statement. This is usually done with the apostrophe which is translated into a record mark.

The minimum useful machine configuration is a 40K 1620 Model I with card I/O, 1443 printer, indirect addressing, and a carriage tape with the overflow channel on line 60. The processor uses 19092 locations but still has all 1620 Model II instructions implemented.

Several sense switch options have been changed. In the four years that we have been doing S.P.S., we have never typed a program at the typewriter. So, switch 1 only controls list, no-list during pass II and switch 2 on stores card images during pass I and fetches card images during pass II. Switch 3 is on for "old" compressed listing output and off for the new, more readable, output format. Switch 4 still controls deck option during pass II. Notice that sense switches do not need to be changed between passes (except if there is a card image-symbol table overlap), so that one now can batch compile small programs of about 400 statements for 40K core.

Our method of operation is to set the origin of the S.P.S. program at 20000 so that the processor is not usually disturbed by loading and executing the assembled program. When we check-stop and find the error, there is no need to re-load the processor, we just transfer to Pass I Initialization (01938). Our printer trace starts at 35000 so that we often have processor, assembled program, and trace in the machine at one

There is one disadvantage to this processor. It is slow, as nearly all the IBM S.P.S. processorshave been. The loss of speed is in the symbol table search. If we were to stay with 1620's for several years, I feel certain a small patch could increase the speed of this processor quite substantially. I know there are faster S.P.S. processors written by users, but I happened to have the source cards for SP-029, and documentation of SP-035.

It's been a lot of fun playing with this processor. I do not plan to submit this to the library, but instead, I want to present an output format that is extremely useful and readable for S.P.S. subroutines for FN II, and a disk-less one-pass processor.

```
PAGE  1
```

MODIFIED OUTPUT

```
       -          -
00402  15  01692  00000      4/8    START  TDM   STARTT,0,07
       -
00414  46  00426  00900      4/8           BLC   *+12,,0
       -          -
00426  15  01693  00000      4/8           TDM   SWSET,0,07
       -             -
00438  17  00650  00000      4/8    PBC    BTM   PBCHK,0,010,

00450  34  00000  00971      4/8           K     ,00971
       -             -
00462  17  00786  00000      4/8    CSS    BTM   SWCHK,0,010,
       -
00474  47  00506  00300      4/8           BNC3  SW1,,0
       -             -
00486  17  01058  00000      4/8           BTM   NUM,0,010,
       -
00498  49  00614  00000      4/8           B     LAST,,0

00506                        4/8           DORG  *-3
       -
00506  47  00554  00100      4/8    SW1    BNC1  SW2,,0
       -
00518  15  01693  00001      4/8           TDM   SWSET,1,0,
       -
00530  37  01695  00500      4/8           RACD  INPUT,,0
       -             -
00542  17  01322  00000      4/8           BTM   PRINT,0,010,
       -
00554  47  00614  00200      4/8    SW2    BNC2  LAST,,0
       --
00566  43  00602  01693      4/8           BD    *+36,SWSET,01
       -
00578  15  01693  00001      4/8           TDM   SWSET,1,0
       -
00590  37  01695  00500      4/8           RACD  INPUT,,0
       -             -
00602  17  01678  00000      4/8           BTM   REPROD,0,010,
       -
00614  46  00414  00900      4/8    LAST   BLC   START+12,,0
       -
00626  15  01693  00000      4/8           TDM   SWSET,0,0
       -
00638  49  00462  00000      4/8           B     CSS,,0

00646                        4/8           DORG  *-3

                             4/8    *

00649                        4/8           DORG  *+4
       -
00650  39  00741  00901      4/8    PBCHK  WA    PCHK,00901,0
       -           -
00662  16  00739  01900      4/8           TFM   PBCNT,1900,08
       -           -
00674  12  00739  00001      4/8           SM    PBCNT,1,08
       -
00686  47  00674  01200      4/8           BNZ   *-12,,0
```

OLD OUTPUT

```
PAGE    1

00402  00   00000   00000        4/8                  -
00414  J5   01692   -0000        4/8      START   TDM   STARTT,0,07
00426  M6   00438   00900        4/8              BLC   *+12,,0
00438  J5   01693   -0000        4/8              TDM   SWSET,0,07
00450  J7   00650   000-0        4/8      PBC     BTM   PBCHK,0,010,
00462  34   00000   00971        4/8              K     ,00971
00474  J7   00786   000-0        4/8      CSS     BTM   SWCHK,0,010,
00486  M7   00506   00300        4/8              BNC3 SW1,,0
00498  J7   01058   000-0        4/8              BTM   NUM,0,010,
00510  M9   00614   0000C        4/8              B     LAST,,0
00518                            4/8              DORG *-3
00518  M7   00554   00100        4/8      SW1     BNC1 SW2,,0
00530  J5   01693   00001        4/8              TDM   SWSET,1,0,
00542  L7   01695   0050C        4/8              RACD INPUT,,0
00554  J7   01322   000-C        4/8              BTM   PRINT,0,010,
00566  M7   00614   00200        4/8      SW2     BNC2 LAST,,0
00578  ML   00614   01693        4/8              BD    *+36,SWSET,01
00590  J5   01693   00001        4/8              TDM   SWSET,1,0
00602  L7   01695   00500        4/8              RACD INPUT,,0
00614  J7   01678   000-0        4/8              BTM   REPROD,0,010,
00626  M6   00414   00900        4/8      LAST    BLC   START+12,,0
00638  J5   01693   00000        4/8              TDM   SWSET,0,0
00650  M9   00462   00000        4/8              B     CSS,,0
00658                            4/8              DORG *-3
                                 4/8      *
00661                            4/8              DORG *+4
00662  L9   00741   00901        4/8      PBCHK   WA    PCHK,00901,0
00674  J6   00739   0J900        4/8              TFM   PBCNT,1900,08
00686  J2   00739   0-001        4/8              SM    PBCNT,1,08
00698  M7   00686   01200        4/8              BNZ   *-12,,0
00710  M7   00734   03500        4/8              BNI   PBEND,03500,0
00722  34   00000   00102        4/8              RCTY
00734  L9   00745   00100        4/8              WATY PBCOM,,0
00746  42   00000   00000        4/8      PBEND   BB
00748                            4/8              DORG *-9
00751       00004                4/8      PBCNT   DS    4
00753       00002                4/8      PCHK    DAC   2, ',,
00757       00019                4/8      PBCOM   DAC   19,PRINTER NOT READY.',,
                                 4/8      *
                                 4/8      *
00797                            4/8              DORG *+4
00798  J5   00988   00000        4/8      SWCHK   TDM   SWC,0,0
00810  M7   00834   00100        4/8              BNC1 *+24,,0
00822  J5   00988   00001        4/8              TDM   SWC,1,0
00834  M7   00858   00200        4/8              BNC2 *+24,,0
00846  J5   00988   00001        4/8              TDM   SWC,1,0
00858  M7   00882   00300        4/8              BNC3 *+24,,0
00870  J5   00988   00001        4/8              TDM   SWC,1,0
00882  ML   00986   00988        4/8              BD    SWBB,SWC,01
00894  34   00000   00102        4/8              RCTY
00906  L9   00995   00100        4/8              WATY SWCOM,,0
00918  M6   00950   00100        4/8      SWBC1   BC1   SWBB1,,0
0093C  M6   00950   00200        4/8              BC2   SWBB1,,0
00942  46   00950   00300        4/8              BC3   SWBB1
00954  49   00906   00000        4/8              B     SWBC1
00962                            4/8              DORG *-3
00962  16   00992   0J999        4/8      SWBB1   TFM   SWCNT,1999,8
00974  12   00992   0-001        4/8              SM    SWCNT,1,8
```

# SHORT CUT METHODS IN PROGRAMMING USING SPS

By Richard D. Ross
University, Mississippi

# SHORT CUT METHODS IN PROGRAMMING USING SPS

BY RICHARD D. ROSS
UNIVERSITY OF MISSISSIPPI

PROGRAMMING THE 1620 COMPUTER USING SPS IS A VERY DIFFICULT TASK FOR THE BEGINNING PROGRAMMER. AFTER MANY HOURS OF WORK AND TRIAL AND ERROR PROCEDURES A PROGRAM CAN BE WRITTEN THAT WILL DO A SPECIFIED JOB. THE MOST IMPORTANT CONCERN ABOUT ANY PROGRAM IS THAT THE PROGRAM BE CORRECT. ALTHOUGH THE PROGRAM FUNCTIONS PROPERLY, IT MAY OR MAY NOT BE THE MOST EFFICIENT PROGRAM FOR THE JOB. ESPECIALLY FOR THE PAYING CUSTOMER, THE FASTER THE PROGRAM THE LESS MONEY IS SPENT PER JOB. IF SEVERAL INSTRUCTIONS COULD BE DELETED FROM A PROGRAM THAT WAS RUN ONCE A DAY FOR ONE YEAR THIS WOULD SAVE THE CUSTOMER SEVERAL MINUTES AND IN SOME CASES SEVERAL HOURS OF RUNNING TIME OVER AN EXTENDED PERIOD. THEREFORE, ONCE A PROGRAM IS WRITTEN AND PRODUCES CORRECT OUTPUT, THEN THE PROGRAM SHOULD BE MADE TO BE AS EFFICIENT AS POSSIBLE.

PRESENTED IN THIS PAPER ARE PROBLEMS WITH SEVERAL DIFFERENT SOLUTIONS TO THE PROBLEM WRITTEN IN SPS. YOU WILL BE ABLE TO SEE HOW THE RUNNING TIME OF A PROGRAM CAN BE REDUCED TO LESS THAN HALF THE RUNNING TIME OF THE ORIGINAL PROGRAM AND IN SOME INSTANCES LESS THAN 1/10, 1/20, OR EVEN LESS. THE SOLUTIONS TO THE PROBLEMS GIVEN MAY GIVE NEW IDEAS TO THE BEGINNER AND TO THE ADVANCED PROGRAMMER WHEN USING SPS.

PROBLEM NO. 1   TRANSFER MATRIX N(J) TO MATRIX M(J) WHERE J VARIES
                FROM 1 TO K.

FORTRAN VERSION

```
      DIMENSION M(41),N(41)
      DO 103 J=1,K
103   M(J)= N(J)
```

NO. OF INSTS. EXECUTED = APPROX. 15*K

S P S VERSION-1

```
     *     CONSTANTS USED
     M     DSB 5,41
     N     DSB 5,41
     J     DC  2,0
     *
     *     PROGRAM
     START TFM J,1,10
           TFM T+6,M
           TFM T+11,N
     T     TF  M,N
           AM  T+6,5,10
           AM  T+11,5,10
           AM  J,1,10
           C   J,K
           BNP T
     *
```

NC  OF INSTS. EXECUTED = 6*K + 3

S P S VERSION-2

```
     * CONSTANTS USED
     M       DSB 5,41
     N       DSB 5,41
     X       DSA M,N
     Z       DSC 2,0
     Y       DS  12
     CON     DC  8,50000501
     *
     *       PROGRAM
     START   CF  X+1
             TF  Y,Z+1
             AM  Y,100,9
             S   Y,K
     T       TF  Y-7,Y-2,611
             A   Y,CON
             BD  T,Y-1
```

NO. OF INSTS. EXECUTED = 3*K + 4

IF THERE ARE NO RECORD MARKS IN MATRIX N, THEN ONE OF THE FOLLOWING VERSIONS MAY BE USED.

S P S VERSION-3

```
      *     CONSTANTS USED
      M     DSB 5,41
      N     DSB 5,41
      *
      *     PROGRAM
      START MM  K,5,9
            AM  99,N-4
            TD  90,99,11
            TDM 99,,6
            DC  1,-,*
            TR  M-4,N-4
            TD  99,90,6
            AM  99,M-4-N+4
            TD  99,90,6
```

NO. OF INSTS. EXECUTED = 8

S P S VERSION-4

```
      *     CONSTANTS USED
      M     DSB 5,41
            DS  1
      N     DSB 5,41
            DC  1,-
      *
      *     PROGRAM
      START TR  M-4,N-4
```

NO. OF INSTS. EXECUTED = 1

PROBLEM NO. 2 READ A CARD ALPHABETICALLY AND IF THERE IS A 0, 1, 6, OR 9 PUNCH IN COLUMN 80, PUNCH THE CARD, IF NOT READ ANOTHER CARD.

S P S VERSION-1

```
            DORG 500
      X     DAS 80
      K     DC  2,0
      *
      START RACD X
            TD  K,X+80*2-2
            CM  K,0,10
            BE  WR
            CM  K,1,10
            BE  WR
            CM  K,6,10
            BE  WR
            CM  K,9,10
            BNE START
      WR    WACD X
            B   START
            DEND START
```

NO. OF INSTS. EXECUTED = 12

S P S VERSION-2

```
            DORG 500
      CON   DC  10,0011110110
      X     DAS 80
      *
      START RACD X
            TD  R+11,X+80*2-2
      R     BD  RD,CON
            WACD X
            B   START
            DEND START
```

NO. OF INSTS. EXECUTED = 5

PROBLEM NO. 3 GIVEN MATRIX A(J), WHERE J=1,K AND K IS LESS THAN 900. EACH VALUE OF A(J) IS A 10-DIGIT CONSTANT OF THE FORM XXXXXXXXXX WITH A FLAG IN THE FIRST POSITION OF EACH VALUE. WRITE A PROGRAM TO FIND TEN TOTALS B(1), B(2), - - -, B(10) WHERE B(1) IS THE SUM OF THE FIRST DIGITS OF THE VALUES OF A(J), B(2) IS THE SUM OF THE SECOND DIGIT OF THE VALUES OF A(J), ETC.

```
S P S VERSION-1
      A      DSB 10,899
      B      DSB 4,10
      J      DC  3,0
      K      DC  3,0
      L      DC  2,0
      *      PROGRAM
      START  TFM J,10,10
             TFM TA+6,B
      TA     TFM TA,0,8
             AM  TA+6,4,10
             SM  J,1,10
             BNZ TA
      BEGIN  TFM RA+11,A-9
             TF  J,K
      ST     TFM L,10,10
             TFM RB+6,B
      RA     TD  RB+11,A-9
      RB     AM  B,0,10
             AM  RA+11,1,10
             AM  RB+6,4,10
             SM  L,1,10
             BNZ RA
             SM  J,1,10
             BNZ ST
             CF  B
NO. OF INSTS. EXECUTED = 64*K + 45
```

```
S P 5 VERSION 2
A      DSB   10,899
B      DSB   4,10
J      DC    3,0
K      DC    3,0
X      DSA   A-9,B+1+40
       DSC   2,0
Y      DC    7,0
KA     DC    8,100000401
START  TFM   Y-2,B+1
       SM    Y,10,10
TA     TFM   Y-2,0,68
       AM    Y,401,9
       BD    TA,Y-1
       TFM   X+5,B+1+40
       CF    X+1
       TFM   X,A-9
       TF    J,K
ST     SM    X+5+2,4010
RA     TD    RB+11,X,11
RB     AM    X+5,0,610
       A     X+5+2,XA
       BD    RA,X+5+1
       SM    J,1,10
       BNZ   ST
       CF    B
NO OF INSTS. EXCUTED = 43*K + 36
```

```
S P S VERSION 3
      A      DSB 10,899
      XA     DC  8,0
      B      DS  ,XA+4
      X      DS  4*10
             DAS 1
             DS  1
      CONA   DC  21,0
             DS  1
      CONB   DC  43,0
             DS  5
      TB     DC  8,0
      K      DC  3,0
      J      DC  3,0
      CN     DC  42,707770777077707770777077707770777077707770
      START  S   CONB,CONB
             TF  X,CONB+5
             TF  99,X
             TFM TB-3,A+1
             S   TB,K
      TA     TNF CONA,TB-3,11
             TNF CONB,CONA
             A   X,CONB
             AM  TB,10001
             BD  TA,TB-2
             M   CN,K
             S   X,99
             TFM TB-3,B-3+1
             SM  TB,10,10
      TR     SF  TB-3,,6
             AM  T+,4001,8
             BD  TR,TB-2
NO OF INSTS. EXCUTED = 5*K + 39
```

EAST CAROLINA COLLEGE
Greenville, North Carolina

October, 1965

# PROGRAMMED INTERRUPT USING THE 1311

by

F. Milam Johnson

This system allows the interruption of any specially designed program. All interrupted programs are stored on the disk until they are retrieved and execution is continued.

Interruption is accomplished through a subroutine named OUT. This routine examines console switch one (C1) to determine if it is on and/or examines core position 98 for a record mark. Either condition will cause immediate interruption and storage of the current program.

Up to 30 programs may be in partial stages of execution. A special control routine will automatically select the program of highest priority and bring it into core for execution when the machine is not needed for something else.

The system is effected through the OUT subroutine. The form of the subroutine is:

```
6        12       16
LABEL    OUT      A, B, C
```

where A is the use code, B is the program number, and C is the address of the next instruction to execute when the program is retrieved. It is suggested that this subroutine be placed in the program only once and the appropriate return address be transmitted to it when necessary. The following example will illustrate the technique:

```
6     12      16
START OUT     10, 23, RETURN
RETURN B
BEGIN .
      .
      .
      ANY SECTION OF THE PROGRAM
      .
      .
      .
      BNC1    * + 32
      TFM     RETURN + 6, * + 20
      B7      START
      .
      .
      .
      DEND    START
```

C1 should be tested at appropriate intervals and at least once in every long loop.

The system is most easily effected by changing the appropriate cards in the monitor deck to make monitor think that 30 cylinders are occupied by stored programs. Monitor is then loaded in the usual fashion. The source decks are then compiled under monitor. They are self storing. The last phase is to compile and execute a program that changes monitor to establish all the interfacing between the two systems.

A brief explanation of the complex and its interaction with the monitor system follows.

## CONTROL CARD CHECKER

Control Card Checker (CCCK) - After having read a monitor control card and determined whether it is a JOB card or otherwise, monitor branches to the appropriate place in CCCK. If the card read was a JOB card, CCCK extracts the use code and sees if it is 999, calls utility routine (see page 3) and transfers control to it. If the code is not 999, control is returned to monitor. If the card read was not a JOB card, control is transferred to meter control (METERC - see page 2). The use code numbers are in columns 36, 37, and 38.

## METERC - Meter Control

METERC - This routine checks to see if the new user is the same as the last user. If the use codes are the same, control is transferred back to the previously specified program. If the use codes are different, a meter reading is requested and time is credited to the last user. Control is then returned to Monitor.

## NEXTIN - Next Caller

NEXTIN - At the time of call exit, NEXTIN is loaded into core position zero. This program copies the part of monitor located in core positions 16000 through 20000 onto the disk to prevent damage. NEXTIN then copies NEXT into core and transfers to it.

## NEXT - Main Control Program

NEXT - This program types NEXT C2 ON FOR MONITOR and halts to allow the operator to change console switch two. If C2 is turned on, control is transferred to the Control Card Checker - CCCK. If C2 is turned off, NEXT scans to see if there are any active programs stored. An active program has a priority greater than zero. If any active programs are found, NEXT examines all active programs to delineate the one of highest priority and types PROGRAM XX CONTINUED. Control is transferred to Meter Control (METERC) after indicating to METERC that control is to be returned to NEXT. When control is returned to NEXT, program XX is loaded into core and execution is continued at the proper instruction. If there are no active programs, the message NO PROGRAMS STORED is typed and control is transferred to monitor.

## Utility Routine

Utility Routine - This program types UTILITY ROUTINE. A control card is read and checked for validity. If the request is not correct, the message INVALID CONTROL CARD is typed and control is returned to the beginning of the Utility Routine. If the request card is valid, the program listed on the card is typed. The required program is located on the disk, read into core, and receives control. If the called program needs information, such is requested prior to the transfer of control.

## NIA - Execute Program

NIA - This program allows for the immediate execution of any stored program regardless of its priority number. NIA is called down by Utility Routine (see page 3) with control card *Execute Program XX (where XX is the program number). ---NIA first checks to see if the program number is valid. If it is not, "Invalid Program Number" is typed, and control is transferred to Utility Routine. If the number is valid, NIA checks to see if the program is stored. If it is not, "Not Stored" is typed. If it is stored, the program is read into core and control is transferred into it.

## NIB - Delete Program

NIB - This routine allows for the deletion of a stored program. NIB is called down by the Utility Routine (see page 3) using the control card, Delete Program XX (where XX is the number of the program to be deleted). NIB first checks to see if the program number is valid. If it is not, INVALID PROGRAM NUMBER is typed and control is transferred to the Utility Program. If the number is valid, NIB deletes the program from the Program Location Map and makes the cylinders in which program XX have been stored available for other programs.

## NIC - Type Time and Initialize

NIC - This routine types a list of all users that have accumulated time since the last time request. The user number, the amount of time used since the last time request, and the meter reading in hours for the last time that the user was on the computer. All time readings are set to zero after being typed. This routine is called by the Utility Routine using the control card *TYPE TIME AND INITIALIZE.

## NID - Type Open Programs

NID - This routine types a list of all the program numbers that are not currently being used. NID is called by the Utility Routine using the control card *TYPE OPEN PROGRAMS.

## NIE - Type Priority Table

NIE - This routine types a list of the active programs, their Use Code, and their priority numbers. NIE is called by the Utility Routine using the control card *TYPE PRIORITY TABLE.

## NIF - Change Priority

NIF - This routine allows the operator to change the priority of any active program. NIF is called by the Utility Program using the control card *CHANGE PRIORITY. NIF requests the program number by typing PROGRAM XX. The program number is checked for validity. Invalid program numbers cause one of the following messages to be typed: OVERTYPE or INVALID PROGRAM NUMBER. Control is then returned to the beginning of NIF.

If a record mark is typed for the program number, control is returned to the Utility Routine. If the program number is valid, NIF checks to see if the program is stored. The message NOT STORED is typed if the requested program cannot be found and control is transferred to the beginning of NIF. When the requested program is located, NIF asks for the desired priority number by typing PRIORITY XX. The priority number is also checked for a record mark and for validity. The results are the same as indicated above. If the priority request is valid, NIF changes the priority of the stated program and adjusts the priorities of as many programs as necessary to insure unique priority numbers for each program.

## NIG - Type Location Table

NIG - This routine types the number of each currently stored program, its users code, the address of the first sector of the first cylinder, the address of the first sector of the second cylinder used, and the location of the next instruction in the program that is to be executed. NIG is called by the Utility Routine when the control card *TYPE LOCATION TABLE is read.

## NIH - Add Program

NIH - This routine puts the information concerning a newly stored program into the Program Location Map. NIH is called by the Utility Routine when the control card *ADD PROGRAM XX is read. The program number is XX.

NIH first checks to see if the program number is valid. An invalid program number will produce the message INVALID PROGRAM NUMBER and control is returned to the Utility Routine. If the requested number is valid, a check is performed to see if any stored program is already using that number. The message IN USE is typed when redundancy occurs and control is transferred to the Utility Routine.

When an acceptable number is requested the message USE CODE XX is typed. The operator must supply a use code which also receives a validity check. If the use code is acceptable, it is stored in the appropriate place in the program location map.

NIH then types CYLINDER ONE SECTOR XXXXX and follows the same procedure in checking and storing the use code. When an acceptable sector address has been received, an availability check is performed. If the cylinder requested is not available the message IN USE is typed and another address is requested. Acceptable addresses that are also available are appropriately stored and NIH goes on to the next phase.

NIH then types the message ADDRESS OF NEXT INSTRUCTION and follows the same accepting and checking procedure as described above. The next worst (numerically highest) priority number is then automatically assigned. This priority number is automatically entered in the Program Location Map. Control is returned to the Utility Routine.

## Output Card

This routine dumps from the disk onto cards. Output Card reads a card containing the lower limit (columns 15-19) and upper limit (columns 22-26) and checks to see that the upper limit is not lower than the lower limit. When this condition occurs the message TRY AGAIN is typed and another request card is read.

When the limits are in the proper sequence a card is punched which contains the specified limits. The specified data area is then punched along with a card sequence number.

When the data area dump is completed, a check for more sector address request cards is made. The message C3 ON FOR REPEATS is typed when the last request has been consumated. Control is then transferred to the first of Output Card or returned through a call exit.

## Out Subroutine

OUT - The out subroutine allows the interruption of a program at some stage prior to completion. The out subroutine is simply included in the main program at suitable intervals. The subroutine has three operands: Use code, Program number, and the next statement to be executed. An invalid out statement will produce the message INVALID OUT STATEMENT.

When the out statement is valid, the subroutine will:

1. Check to determine if the present program has been consumated. If so, control is transferred to OUTMAIN.

2. Checks to see if C1 is on. If C1 is on, the program is stored in its present state of execution and control is transferred to OUTMAIN.

3. Checks to see if there is a record mark in core position 98. This condition will cause the program to be stored but execution will be continued immediately thereafter.

4. If none of the above conditions are found to exist, control is returned to the interrupted program.

## Out Main

OUTMAIN - OUTMAIN is basically a continuation of the Out Subroutine. When control is received from the Out Subroutine, the validity of the program number is checked. An invalid number will cause the message INVALID PROGRAM NUMBER or PROGRAM TAKEN to be typed, followed by NOT STORED. If the program has not been consumated, control is returned to it, otherwise, control is transferred to NEXT.

If tne program number is valid and there is no conflict of ownership, OUTMAIN then checks to see if the program has been completed. A completed program will cause the release of the specified cylinders and the priorities of all remaining programs are adjusted. The message PROGRAM CONSUMATED is typed. Control is transferred to NEXT.

If the program in execution is not consumated and not previously stored, OUTMAIN assigns the program the next available (worst) priority number and a check is made to see if cylinders are available. Should no cylinders be available for storage, the message NO OPEN CYLINDERS, NOT STORED is typed and control is returned to the interrupted program. If cylinders are available, the program is stored and the program's area on the Program Location Map is filled with the necessary information.

If the present condition is just an interruption of a previously stored program, the message PROGRAM NN INTERRUPTED is typed and control is transferred to monitor. Under the previously stated conditions, however, the program may be stored in its present form and execution continued immediately thereafter.

Tables

AVCYL - Available Cylinder Table

This table contains 31 two-digit numbers. The first is the cylinder number of the stored programs and areas. The other 30 are the numbers of the usable cylinders. A record mark indicates the end of the table. A flag on the units digit of any cylinder number indicates that the cylinder is in use. The table is constructed as: $\overline{2}6\overline{2}7\overline{2}8\overline{2}9\overline{3}0\overline{3}1\overline{3}2\overline{3}3\overline{3}435...\overline{5}5\overline{5}60$#

TIME 1 - This table (read time-one) contains the accumulated time and the hour of the last meter reading for each of the 200 users since the last initialization. It is constructed as: $\overline{X}$XXXXX$\overline{H}$HHH...XXXXXX is the accumulated time in hours and hundredth and HHHH is the hour of the last meter reading. The position in the table is determined by the use code. The first position is for use code zero, the second for code one, and the last for code 199.

TIME 2 - This table contains the use code plus the opening and closing meter readings for the last ten users. The table is constructed as follows with the rightmost user being the last user: $\overline{V}$VVVV#$\overline{0}$00000#$\overline{C}$CCCCC#$\overline{V}$VVVV#$\overline{0}$00000#$\overline{C}$CCCCC#.... Here VVVVV is the use code, 000000 the open meter reading, and CCCCCC the close meter reading.

PLM - Place Location Map. The PLM contains pertinent information on each interrupted program. The information is coded in groups as indicated below and the location of each group is determined by its program number. There are thirty groups, each consisting of five, five-digit fields as follows: $\overline{0}$0000$\overline{0}$0000$\overline{0}$0000$\overline{0}$0000$\overline{0}$0000.

The fields are interpreted as follows:

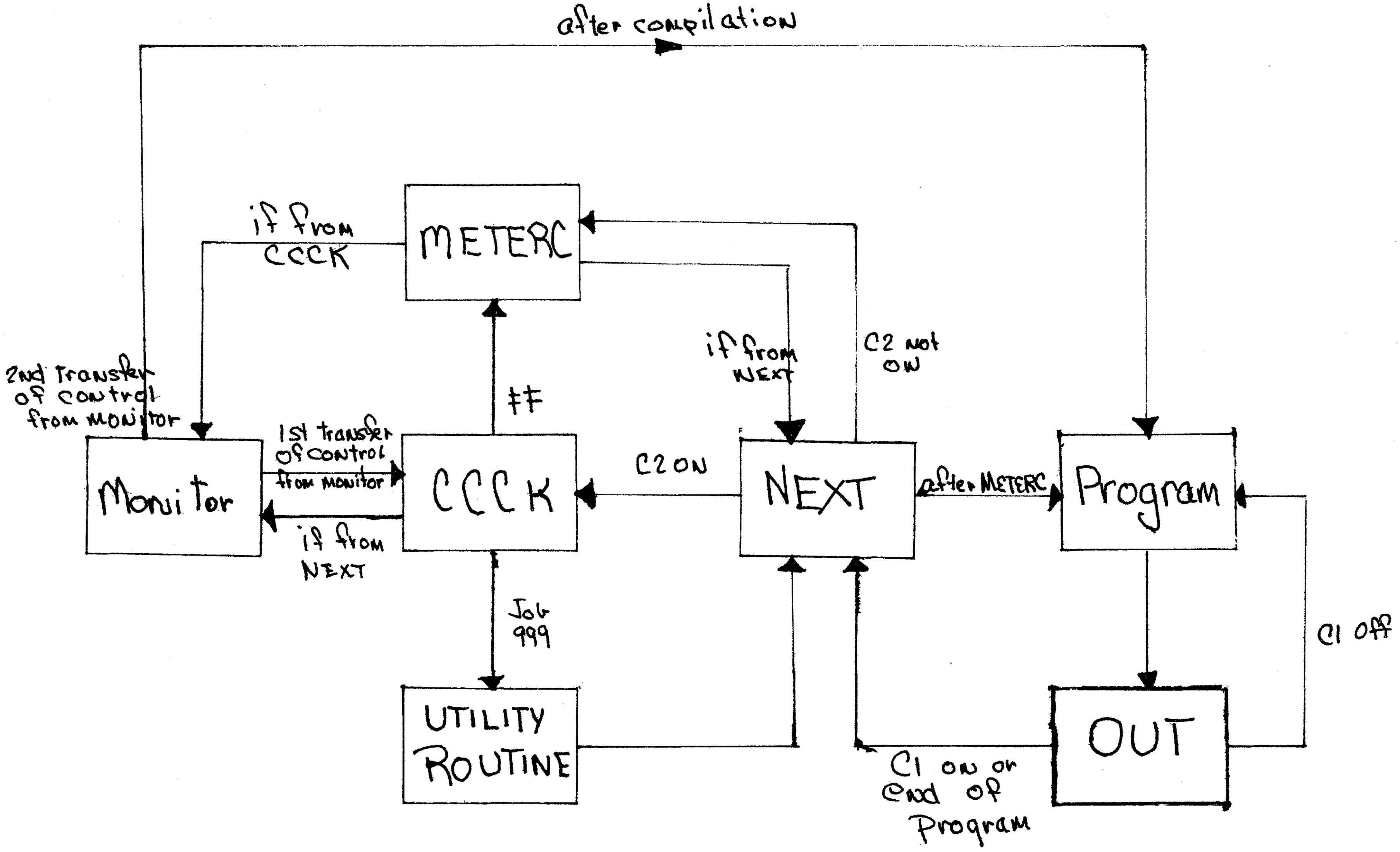
1. The first is the beginning sector of the cylinder used to store interrupted programs.

2. The second field is the beginning sector of the cylinder used to store the Fortran subroutines.

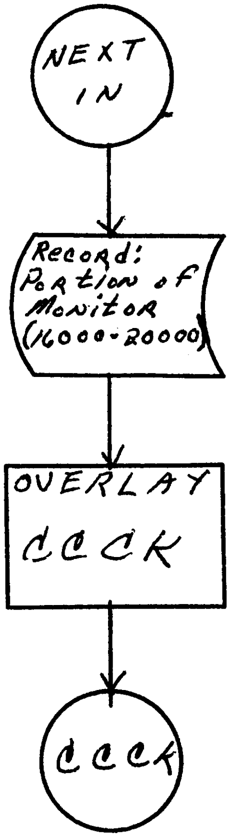
3. The third field is the address of the next instruction to be executed when the program is resumed.

4. The last is the priority of the programs. The programs are executed in order of ascending priority. A priority of zero indicates an inactive program.

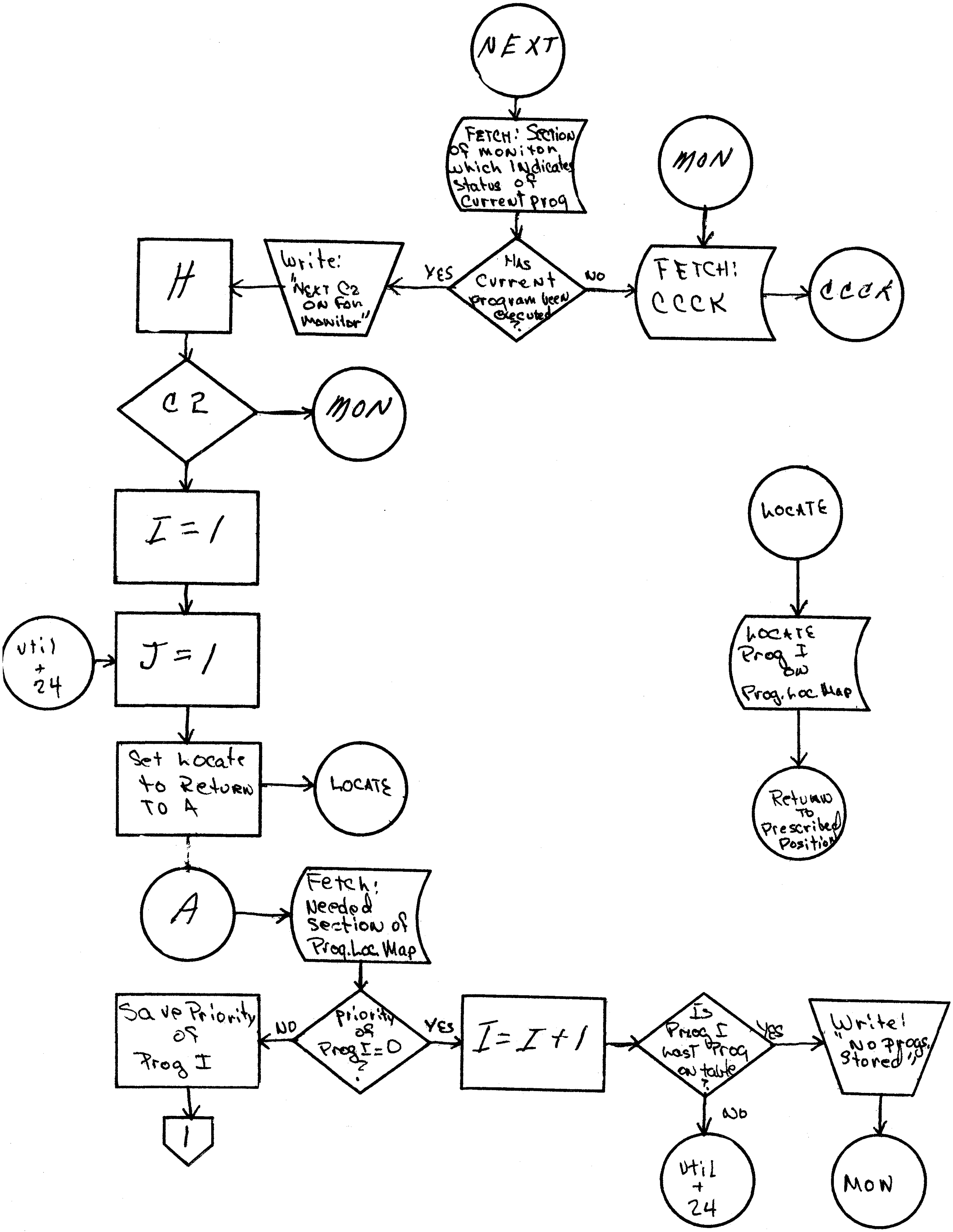
# General Flow Diagram

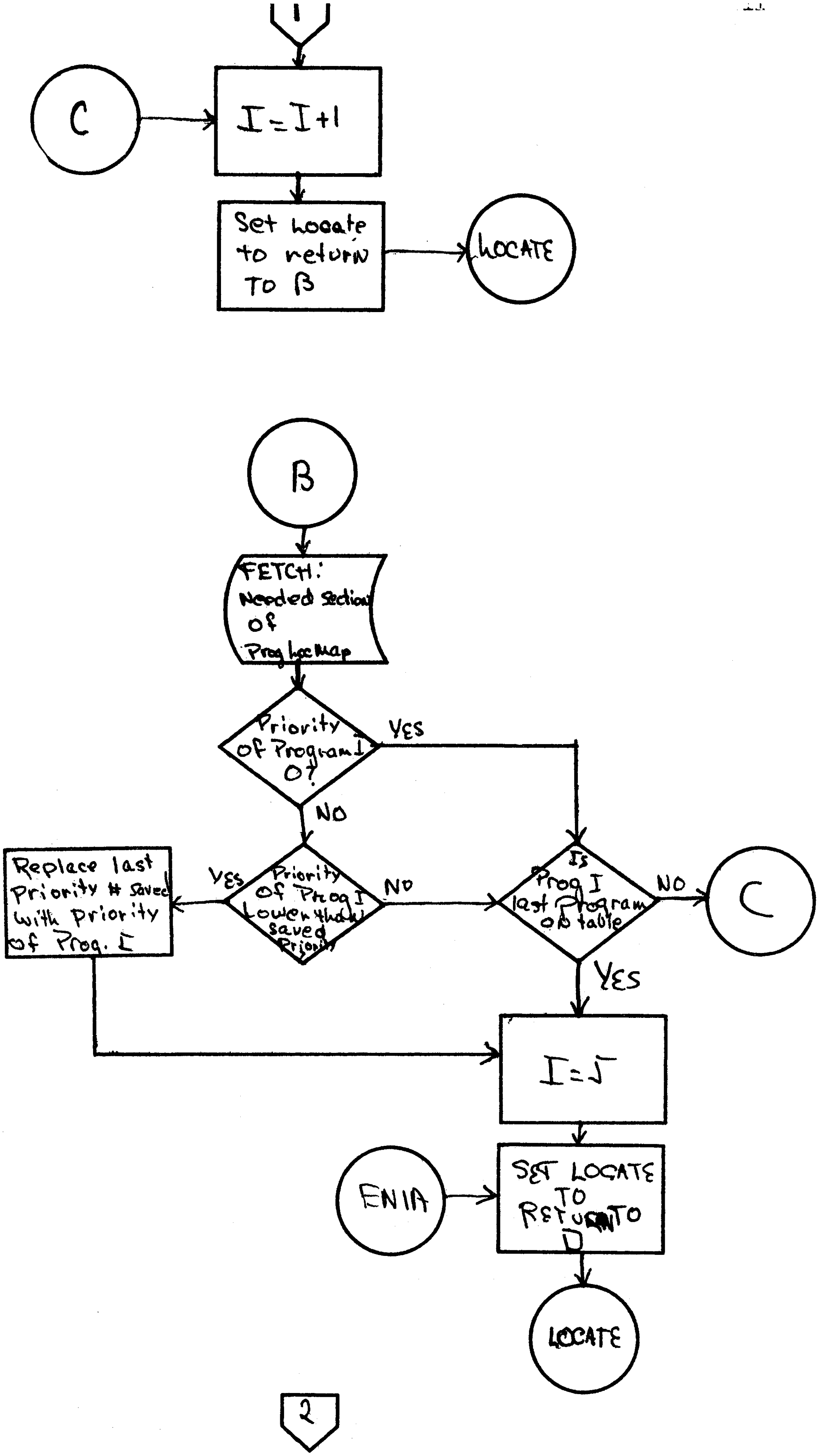


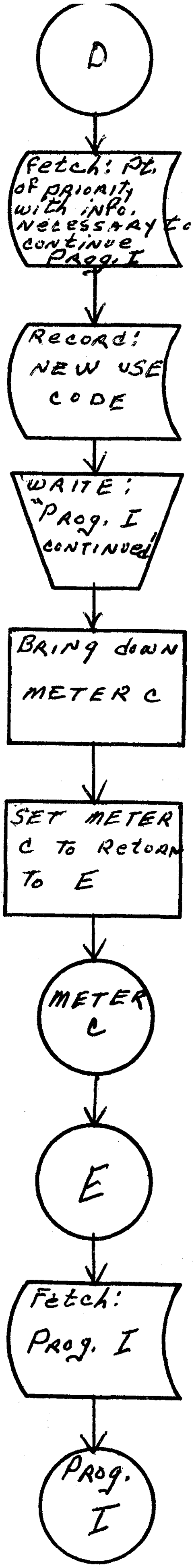
# NEXT IN


NEXT IN
Record: Portion of Monitor (16000-20000)
OVERLAY CCCK
CCCK

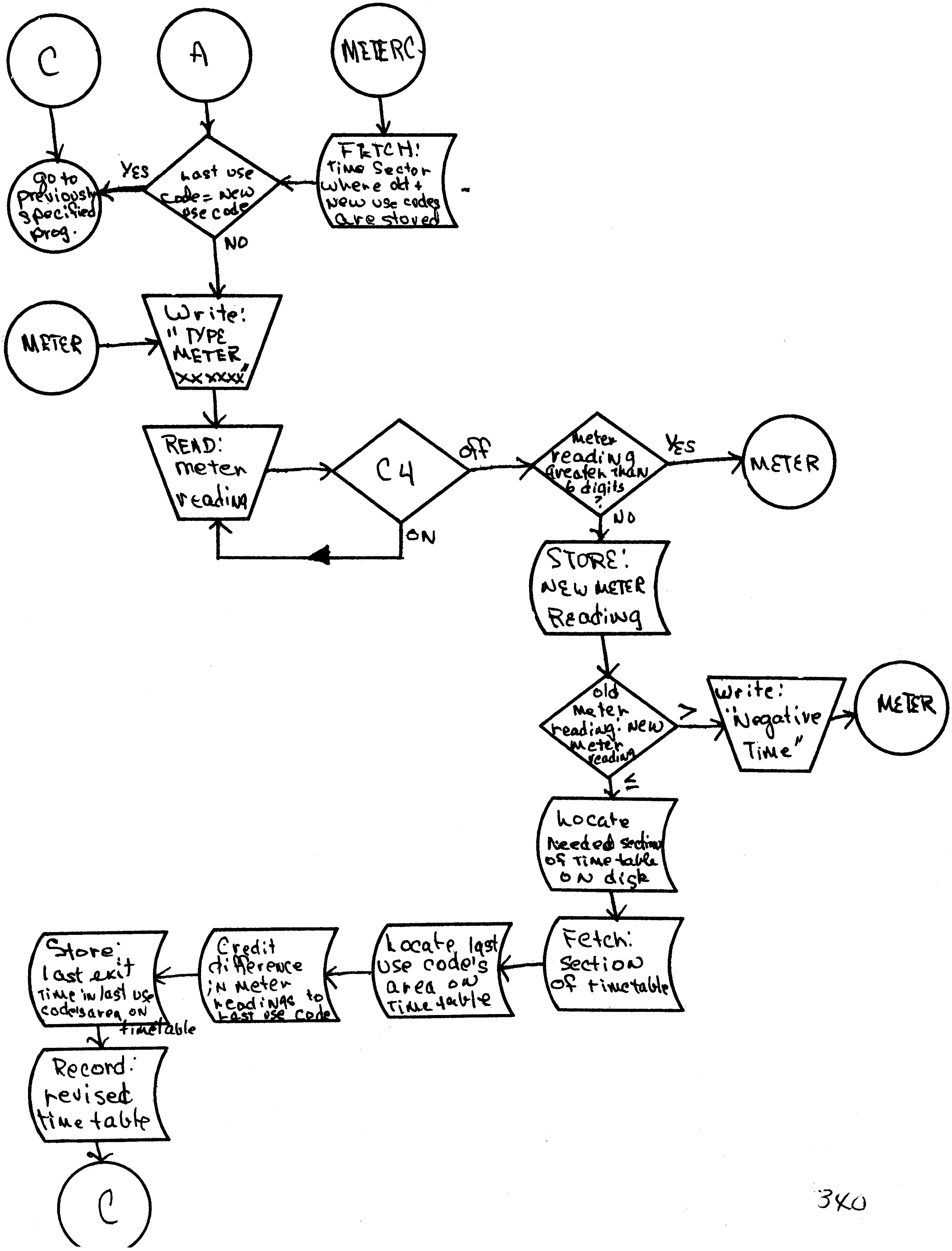
# NEXT


NEXT
FETCH: Section of monitor which Indicates status of Current prog
MON
Has Current program been executed?
YES
NO
Write: "NEXT C2 ON For Monitor"
H
FETCH: CCCK
CCCK
C2
MON
I = 1
LOCATE
Util + 24
J = 1
LOCATE Prog I on Prog. Loc. Map
Set Locate to Return To A
LOCATE
Return To Prescribed Position
A
Fetch: Needed section of Prog. Loc. Map
Save Priority of Prog I
NO
Priority of Prog I = 0?
YES
I = I + 1
Is Prog I last Prog on table?
YES
Write: "No Progs. Stored"
1
NO
Util + 24
MON

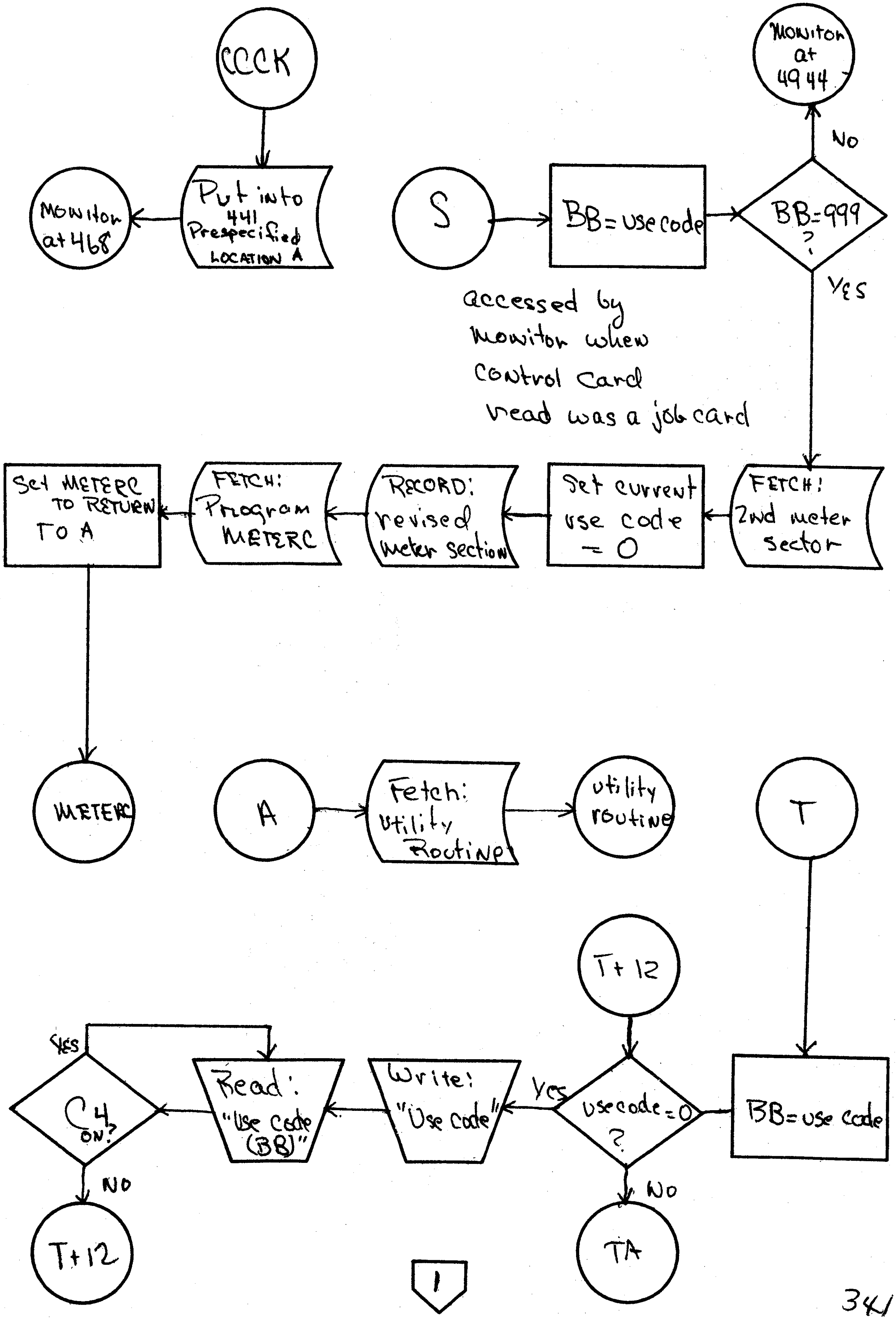

1
C
I=I+1
Set Locate to return To β
LOCATE
B
FETCH: Needed section of Prog Loc Map
Priority of Program I 0?
YES
NO
Replace last Priority # saved with priority of Prog. I
YES
Priority of Prog I Lower than saved Priority
NO
Is Prog I last Program on table
NO
C
YES
I=J
ENIA
SET LOCATE TO RETURN TO D
LOCATE
2


D
fetch: Pt. of priority with info. necessary to continue Prog. I
Record: NEW USE CODE
WRITE: "Prog. I continued"
BRING down METER C
SET METER C To Retorn To E
METER C
E
Fetch: Prog. I
Prog. I

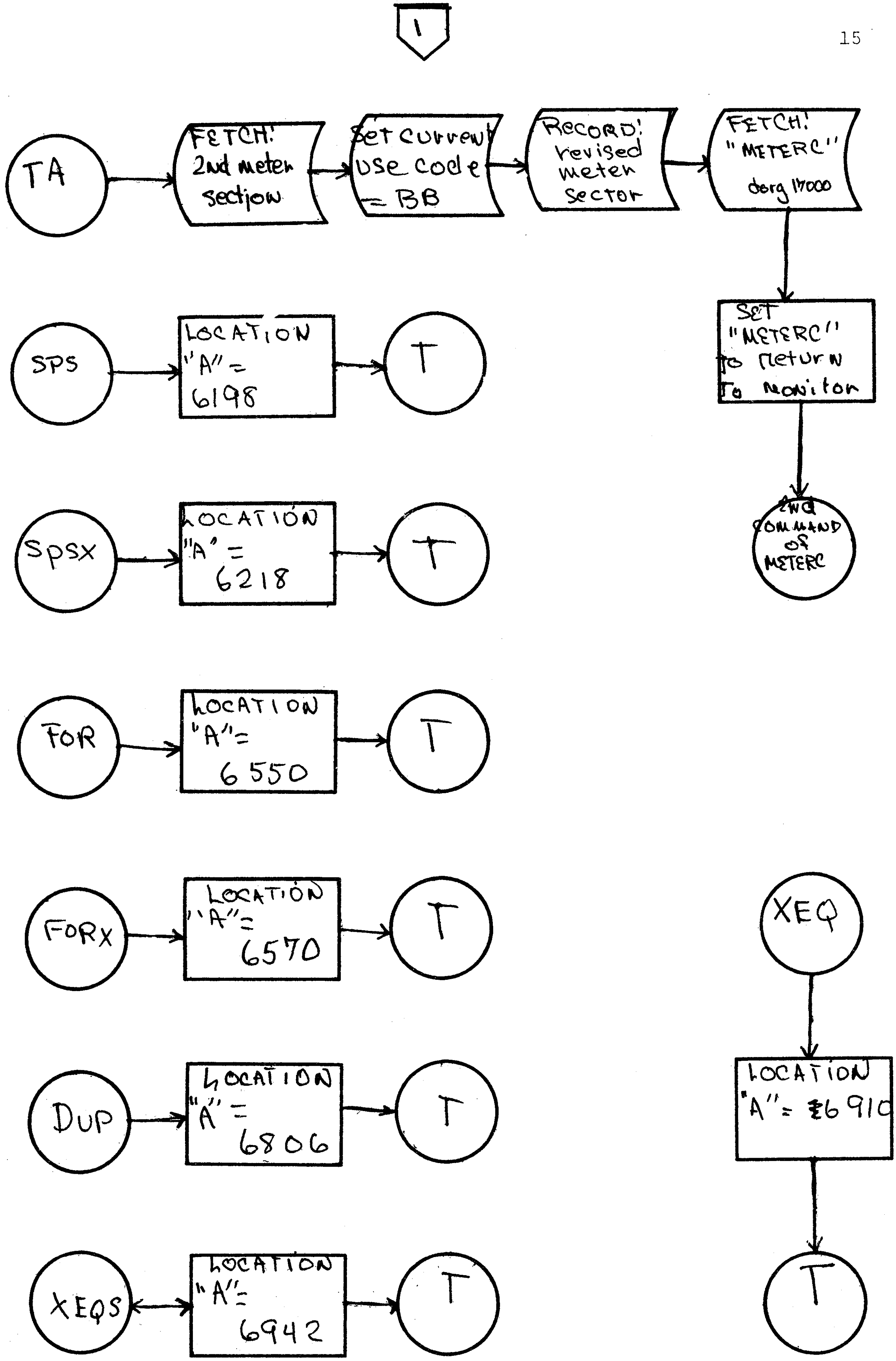
# METER C


C
A
METERC
go to previously specified prog.
YES
Last use code = New use code
NO
FETCH: Time Sector where old + New use codes are stored
METER
Write: " TYPE METER XX XXXX"
READ: meter reading
C4
ON
off
Meter reading greater than 6 digits ?
YES
METER
NO
STORE: NEW METER Reading
old Meter reading : New Meter reading
>
Write: "Negative Time"
METER
≤
Locate Needed section of Time table ON disk
Fetch: Section of timetable
Locate last use code's area on Time table
Credit difference in meter readings to Last use code
Store: last exit time in last use code's area on timetable
Record: revised time table
C

# C C C K

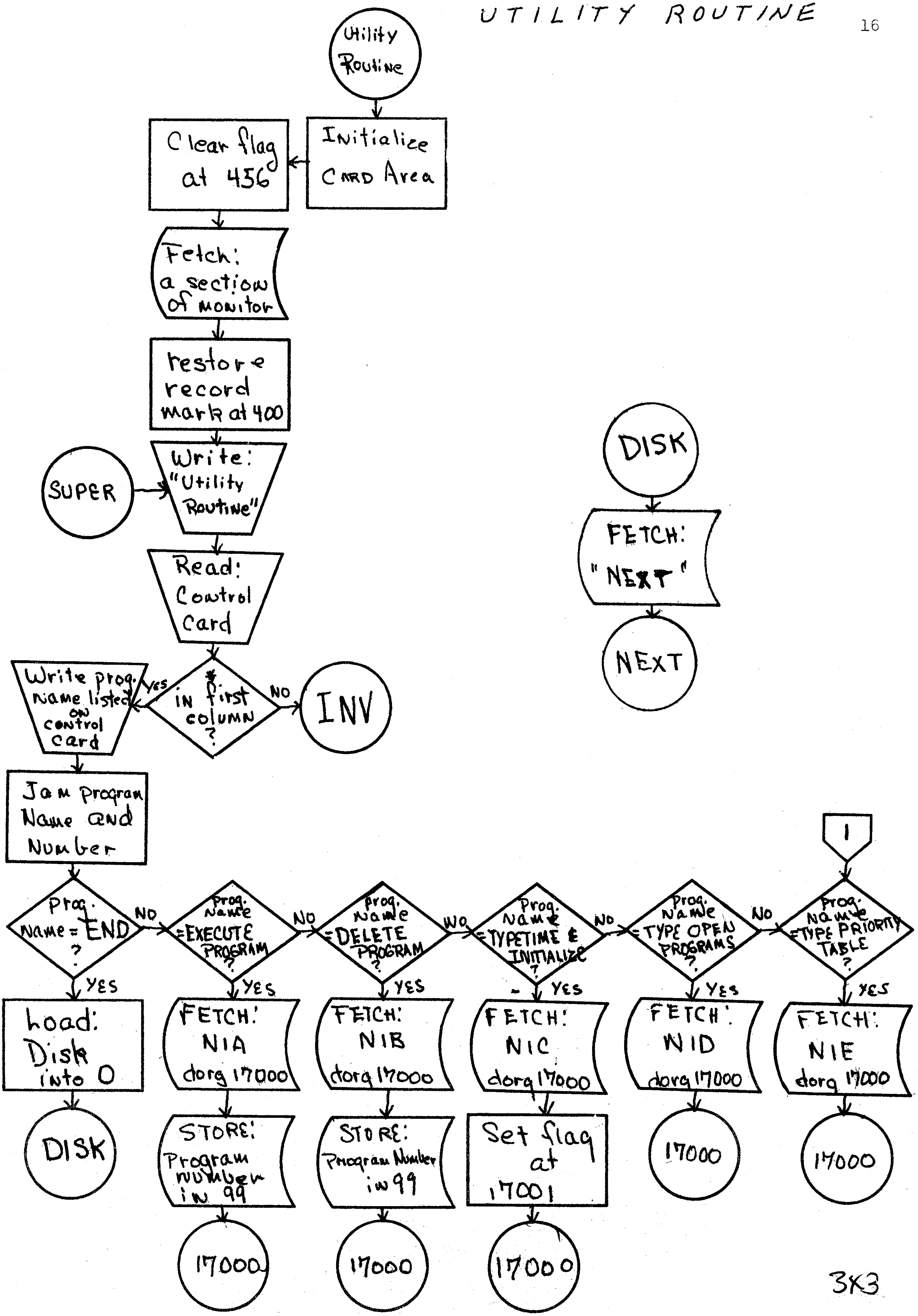


accessed by monitor when control card read was a job card

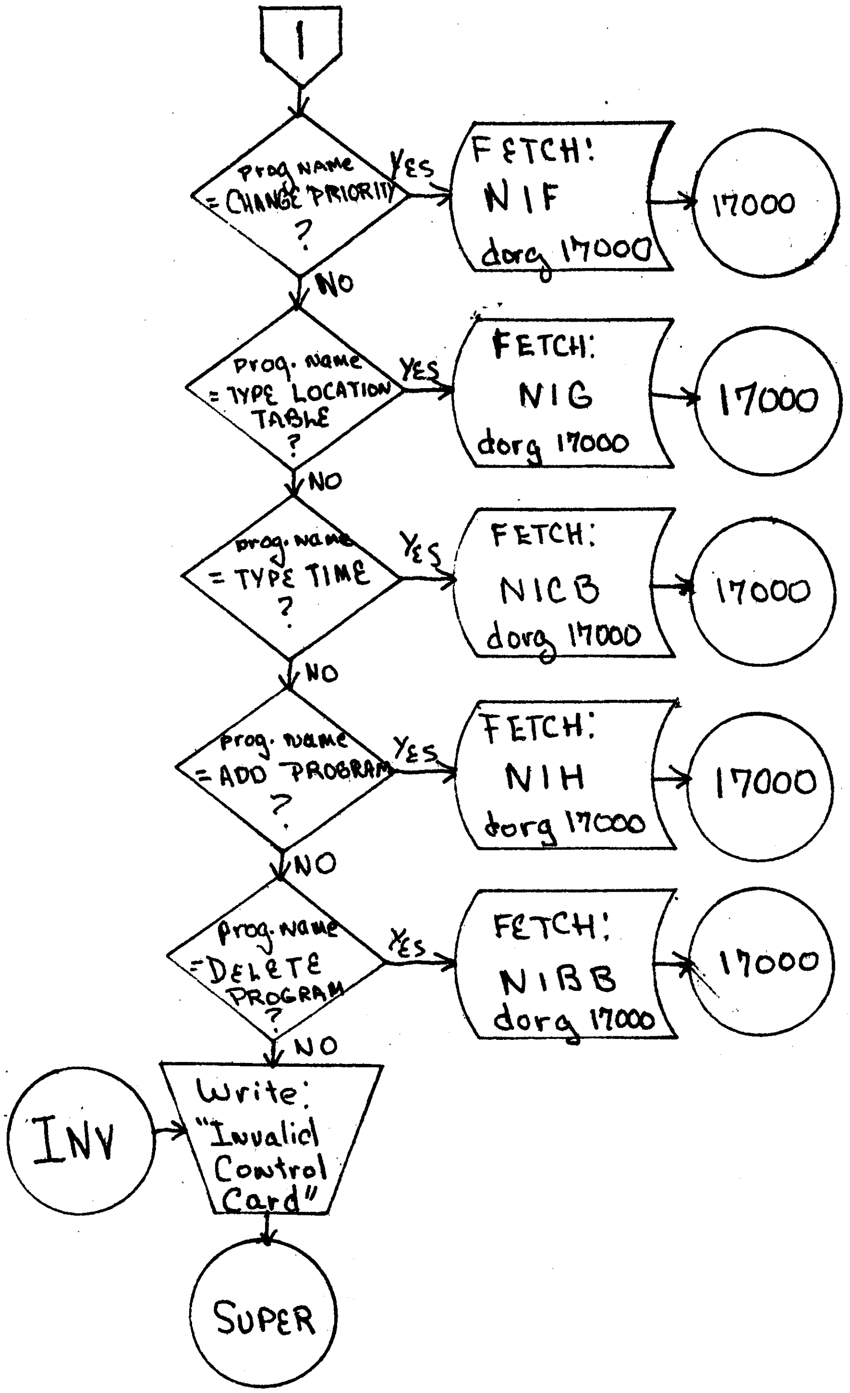

1
TA
FETCH!
2nd meter
section
Set current
use code
= BB
RECORD!
revised
meter
sector
FETCH!
"METERC"
dorg 17000
SET
"METERC"
to return
To monitor
2nd
COMMAND
OF
METERC
SPS
LOCATION
"A" =
6198
T
SpsX
LOCATION
"A" =
6218
T
FOR
LOCATION
"A" =
6550
T
FORX
LOCATION
"A" =
6570
T
XEQ
LOCATION
"A" = 6910
T
DUP
LOCATION
"A" =
6806
T
XEQS
LOCATION
"A" =
6942
T

3x2

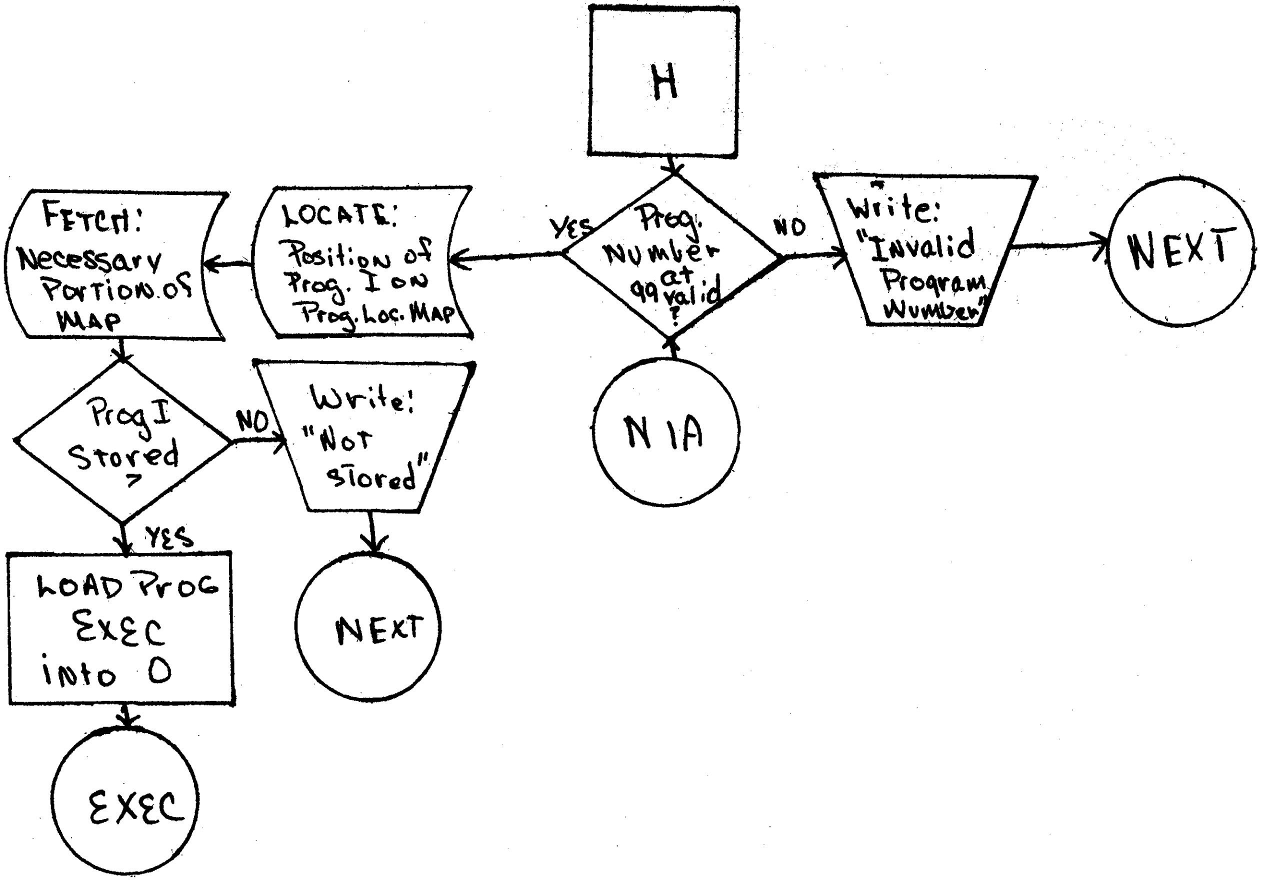
# UTILITY ROUTINE

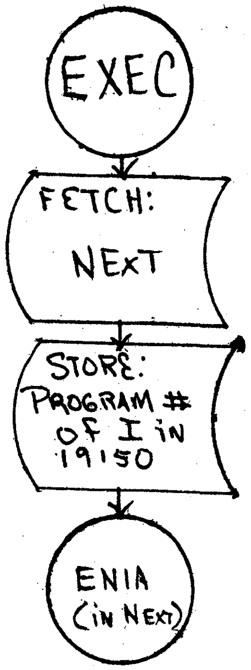


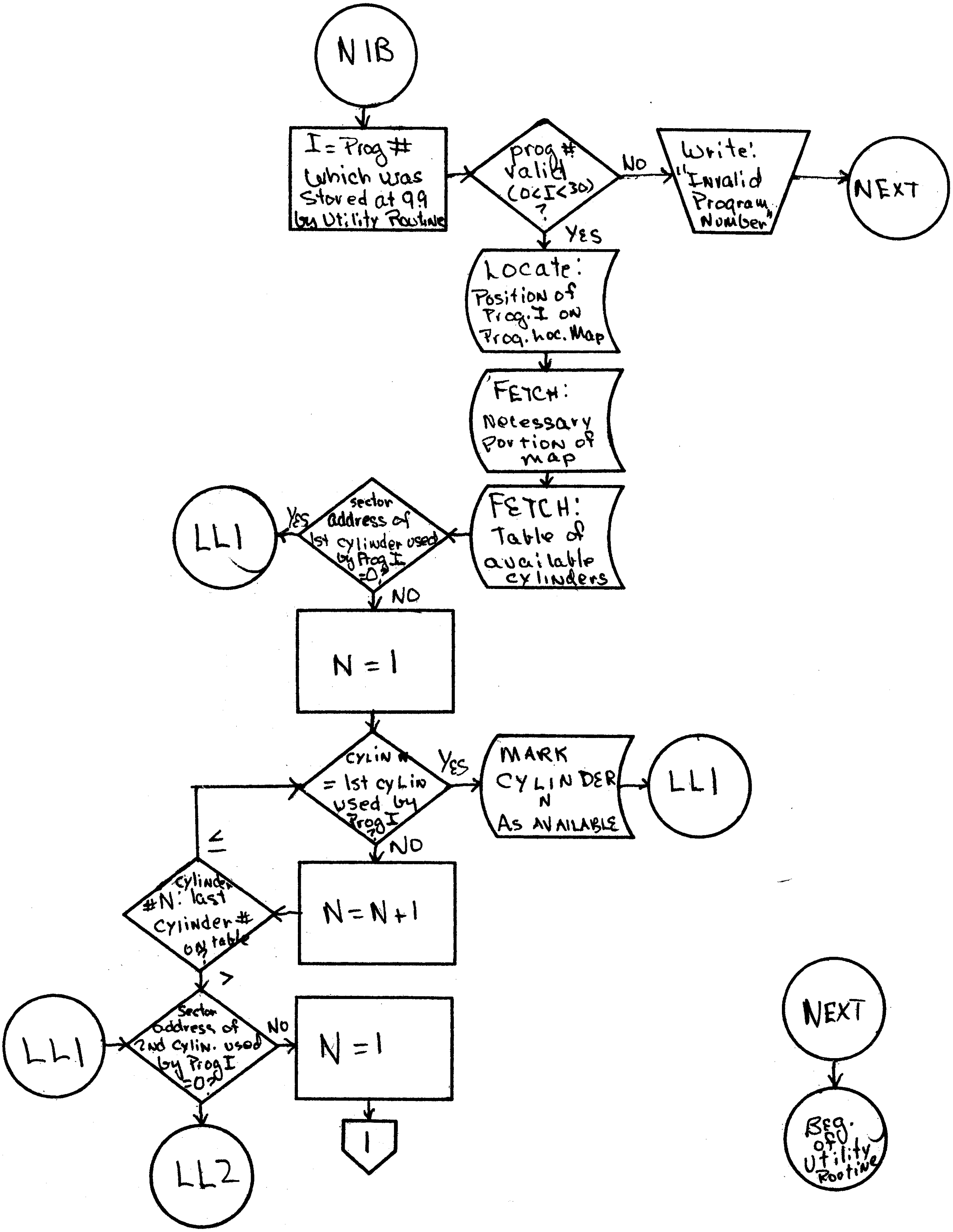
3X3

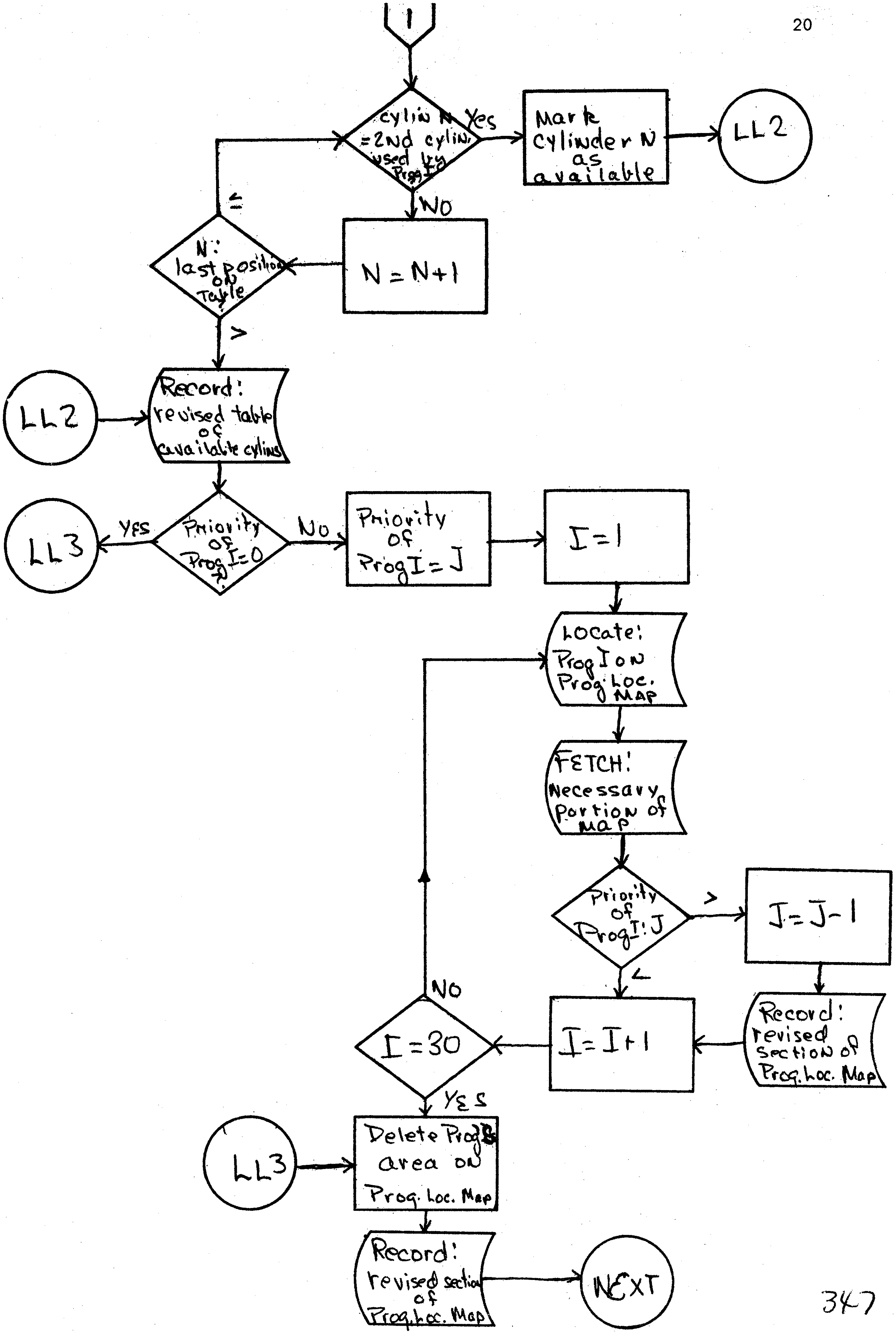

1
Prog NAME = CHANGE PRIORITY ?
YES
NO
FETCH: NIF dorg 17000
17000
Prog. name = TYPE LOCATION TABLE ?
YES
NO
FETCH: NIG dorg 17000
17000
prog. name = TYPE TIME ?
YES
NO
FETCH: NICB dorg 17000
17000
prog. name = ADD PROGRAM ?
YES
NO
FETCH: NIH dorg 17000
17000
prog. name = DELETE PROGRAM ?
YES
NO
FETCH: NIBB dorg 17000
17000
INV
Write: "Invalid Control Card"
SUPER

# NIA

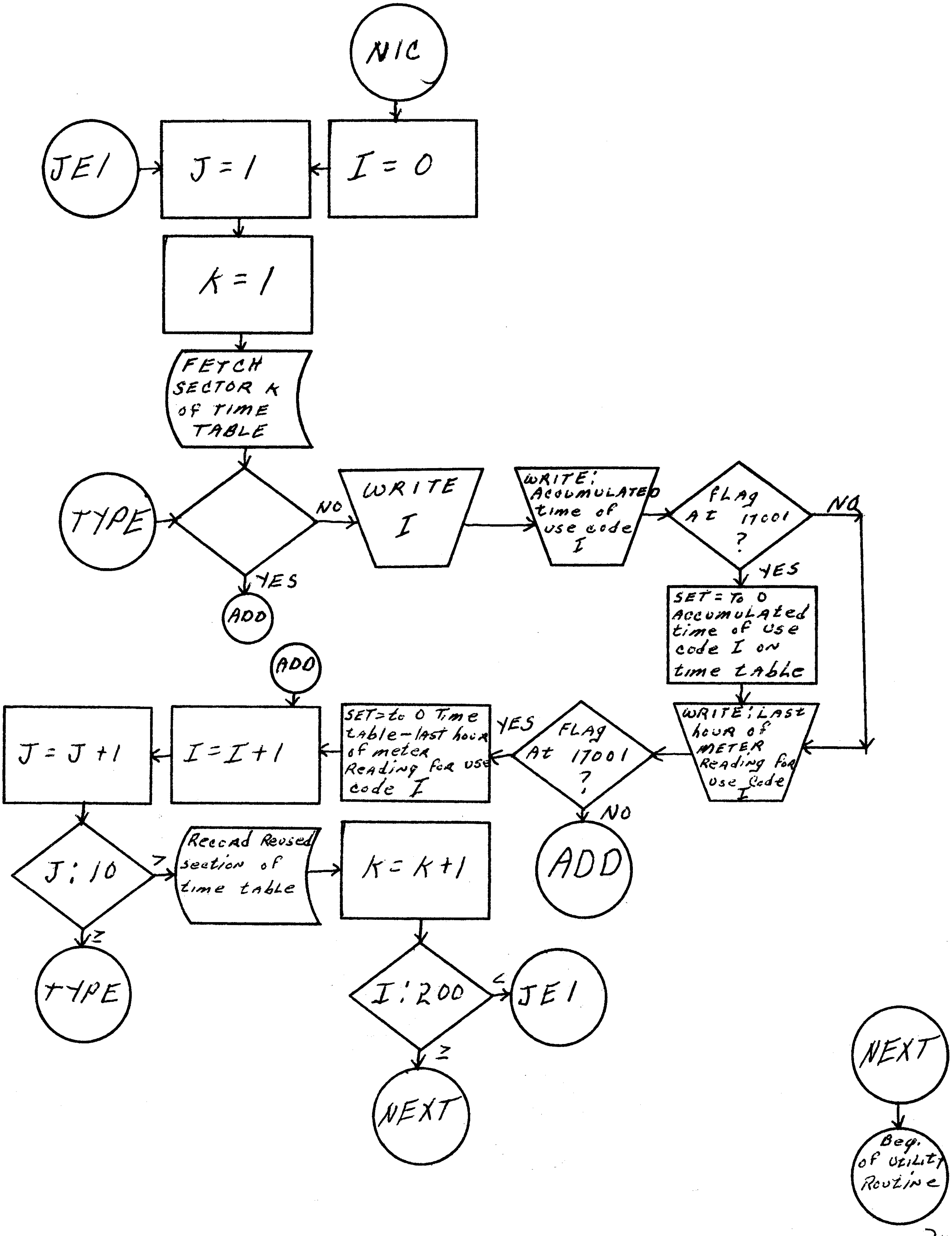

H
Prog. Number at 99 valid ?
YES
NO
Write: "Invalid Program Number"
NEXT
NIA
LOCATE: Position of Prog. I on Prog. Loc. MAP
FETCH: Necessary Portion of MAP
Prog I Stored ?
NO
Write: "Not stored"
NEXT
YES
LOAD Prog EXEC into 0
EXEC


EXEC
FETCH: NEXT
STORE: PROGRAM # of I in 19150
ENIA (in Next)

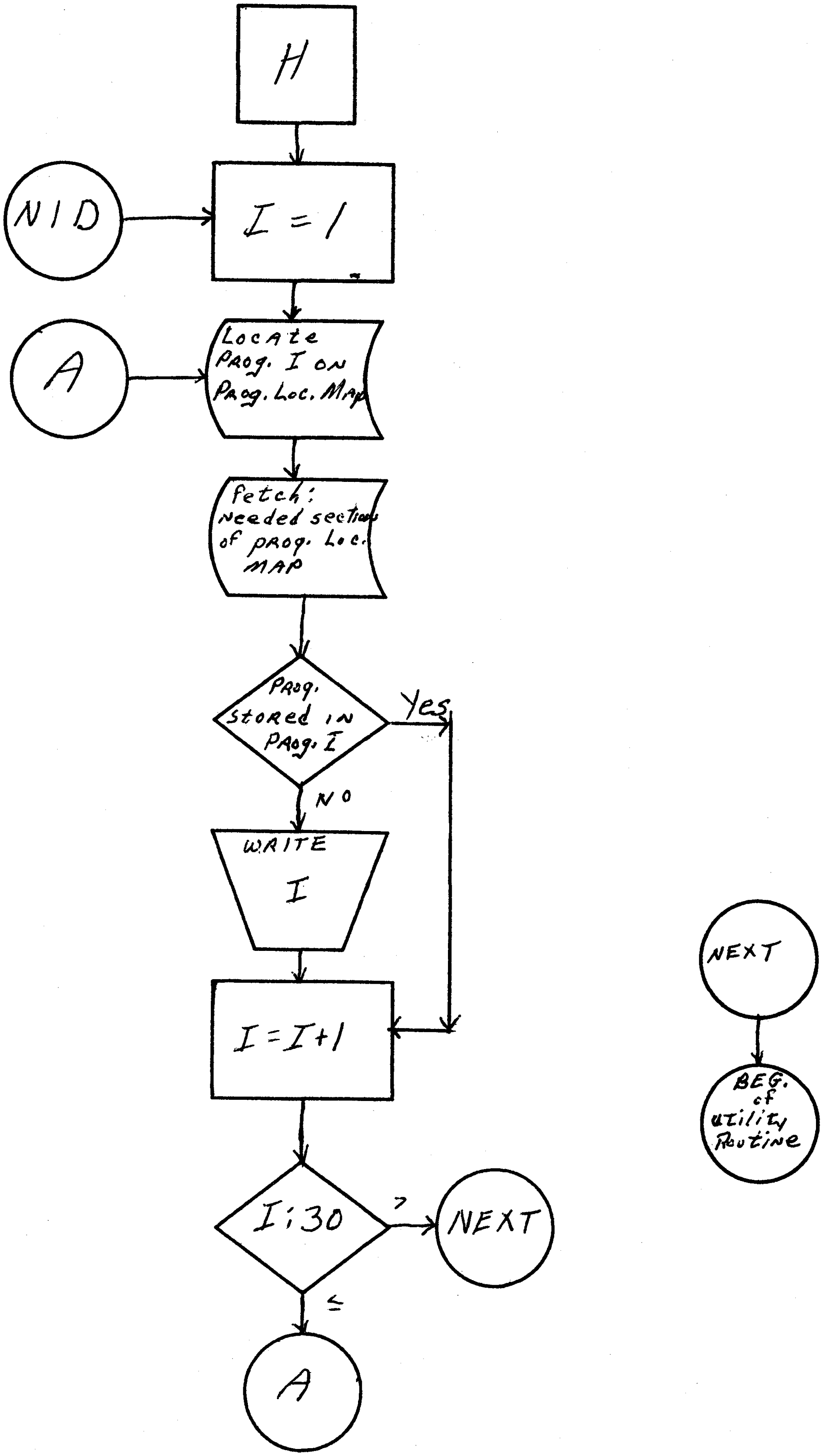

NIB
I = Prog # Which was Stored at 99 by Utility Routine
prog # valid (0<I<30) ?
No
Write: "Invalid Program Number"
NEXT
Yes
Locate: Position of Prog. I on Prog. Loc. Map
FETCH: Necessary Portion of map
FETCH: Table of available cylinders
sector address of 1st cylinder used by Prog I =0?
Yes
LL1
NO
N = 1
CYLIN # = 1st cylin used by Prog I ?
Yes
MARK CYLINDER N As AVAILABLE
LL1
NO
N = N + 1
Cylinder #N: last cylinder # on table
≤
>
LL1
Sector address of 2nd cylin. used by Prog I =0?
NO
N = 1
1
LL2
NEXT
Beg. of Utility Routine


1
cylin N = 2ND cylin used by Prog I
Yes
Mark cylinder N as available
LL2
NO
N = N+1
N : last position on Table ?
≤
>
LL2
Record: revised table of available cylins
LL3
YES
Priority of Prog I = 0 ?
No
Priority of Prog I = J
I = 1
Locate: Prog I on Prog. Loc. Map
FETCH: Necessary Portion of Map
Priority of Prog I : J
>
J = J-1
<
Record: revised Section of Prog. Loc. Map
I = I+1
I = 30
NO
YES
LL3
Delete Progs area on Prog. Loc. Map
Record: revised section of Prog. Loc. Map
NEXT

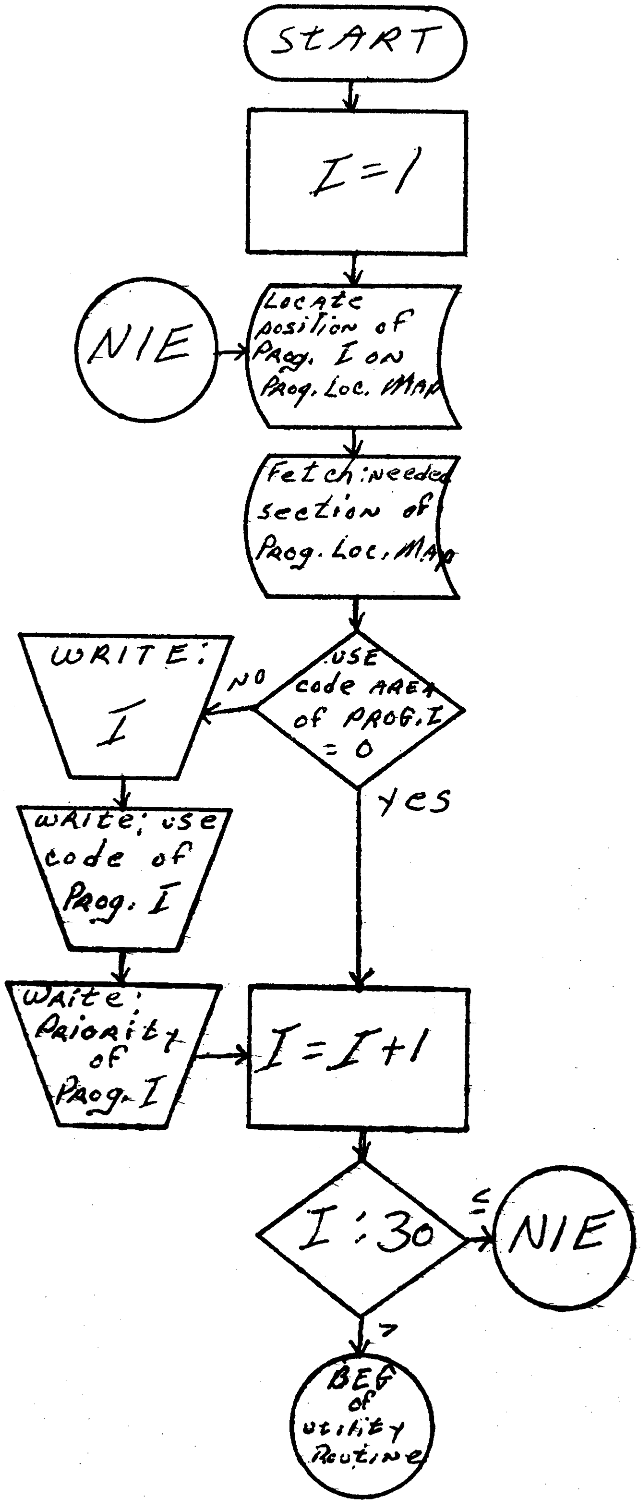
# NIC


NIC
I = 0
JE1
J = 1
K = 1
FETCH SECTOR K of TIME TABLE
TYPE
NO
YES
ADD
WRITE I
WRITE: ACCUMULATED time of use code I
FLAG AT 17001 ?
NO
YES
SET = To 0 Accumulated time of use code I on time table
WRITE: LAST hour of METER Reading for use code I
FLAG At 17001 ?
YES
NO
ADD
SET = to 0 Time table - last hour of meter Reading for use code I
ADD
I = I + 1
J = J + 1
J : 10
>
≥
TYPE
Record Revised section of time table
K = K + 1
I : 200
<
JE1
≥
NEXT
NEXT
Beg. of Utility Routine

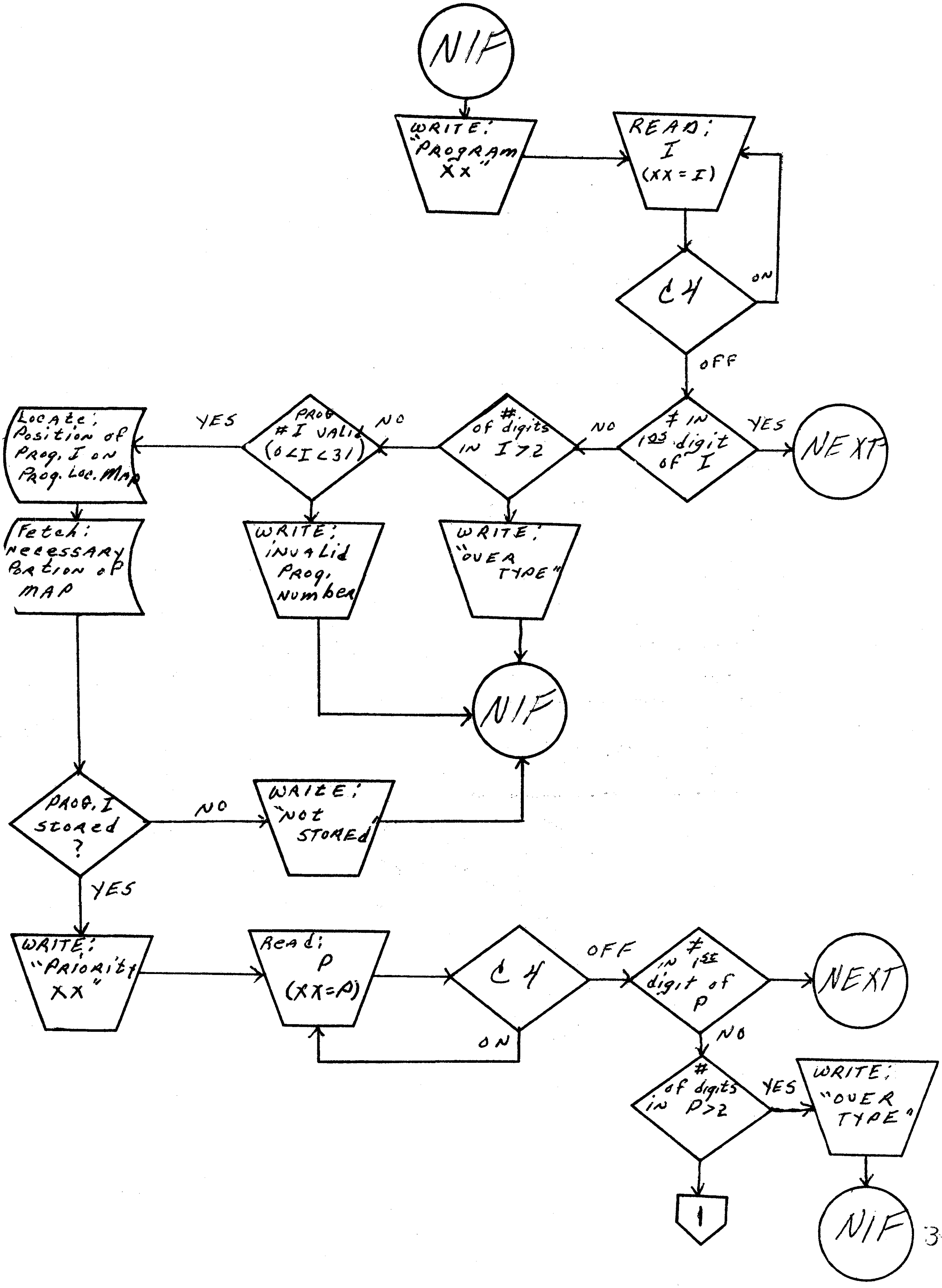
# NID

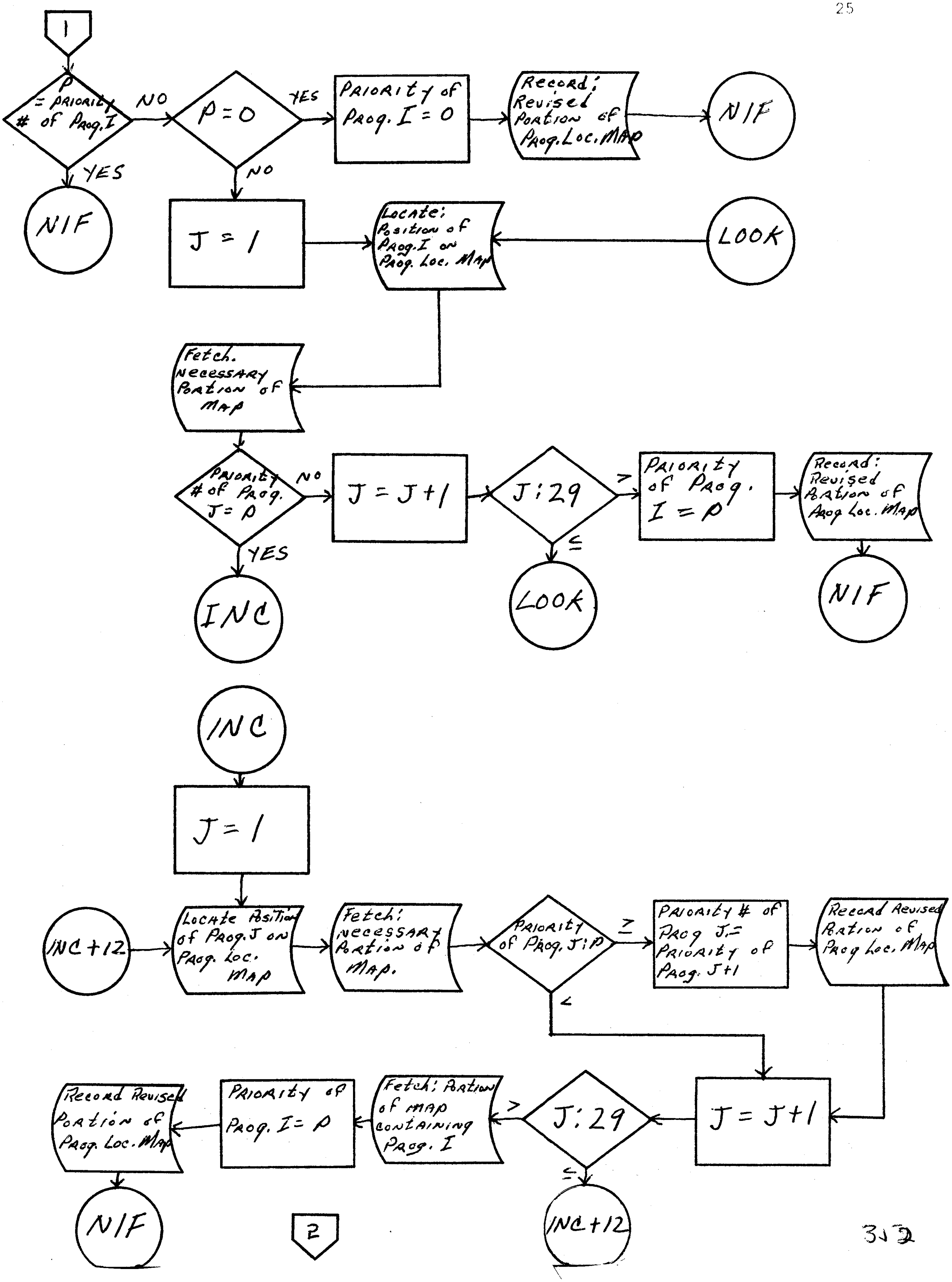

H
NID
I = 1
A
Locate Prog. I on Prog. Loc. Map
fetch: needed sections of prog. Loc. MAP
Prog. stored in Prog. I
Yes
NO
WRITE I
I = I+1
I ; 30
>
NEXT
≤
A
NEXT
BEG. of utility Routine

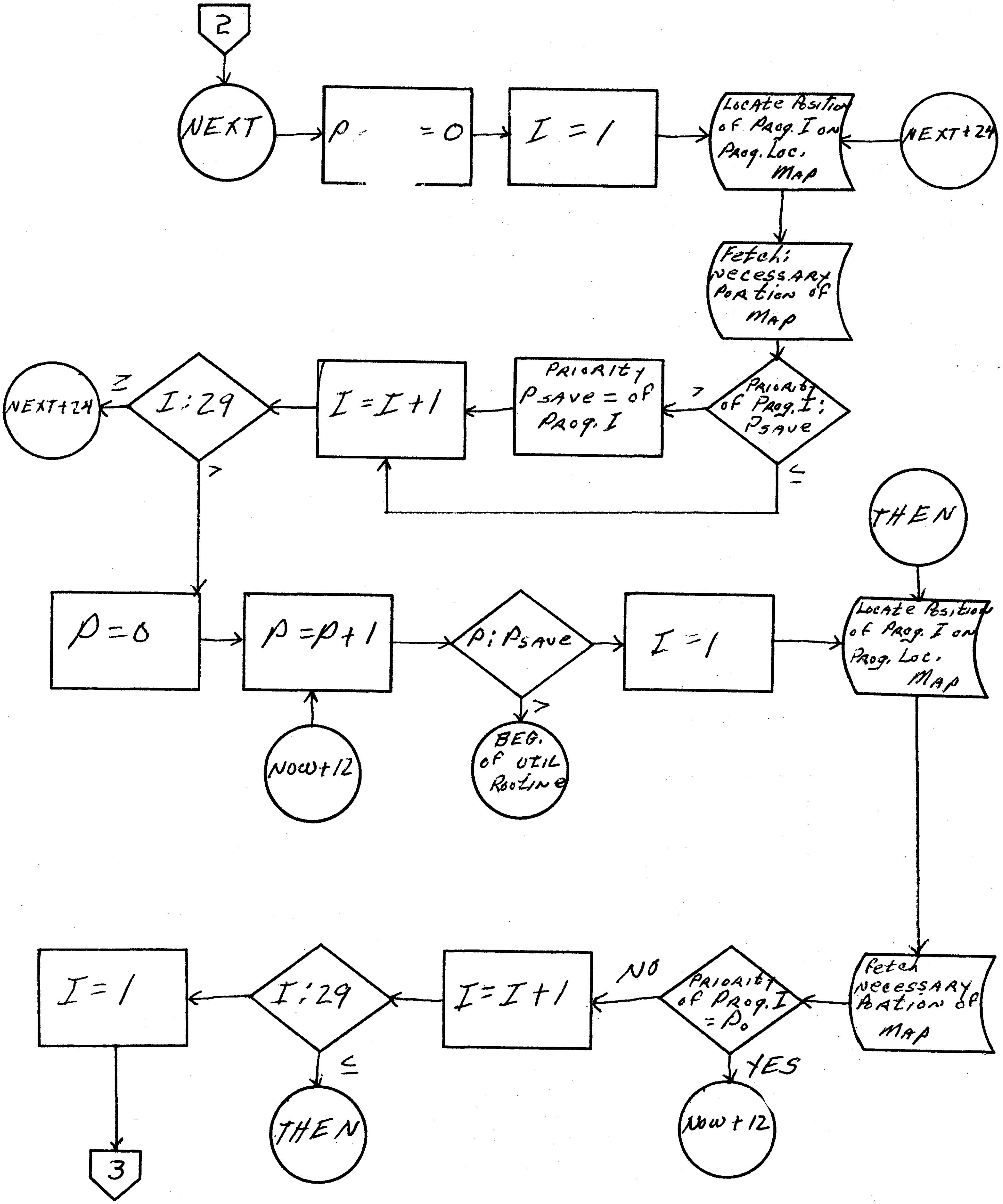
# NIE

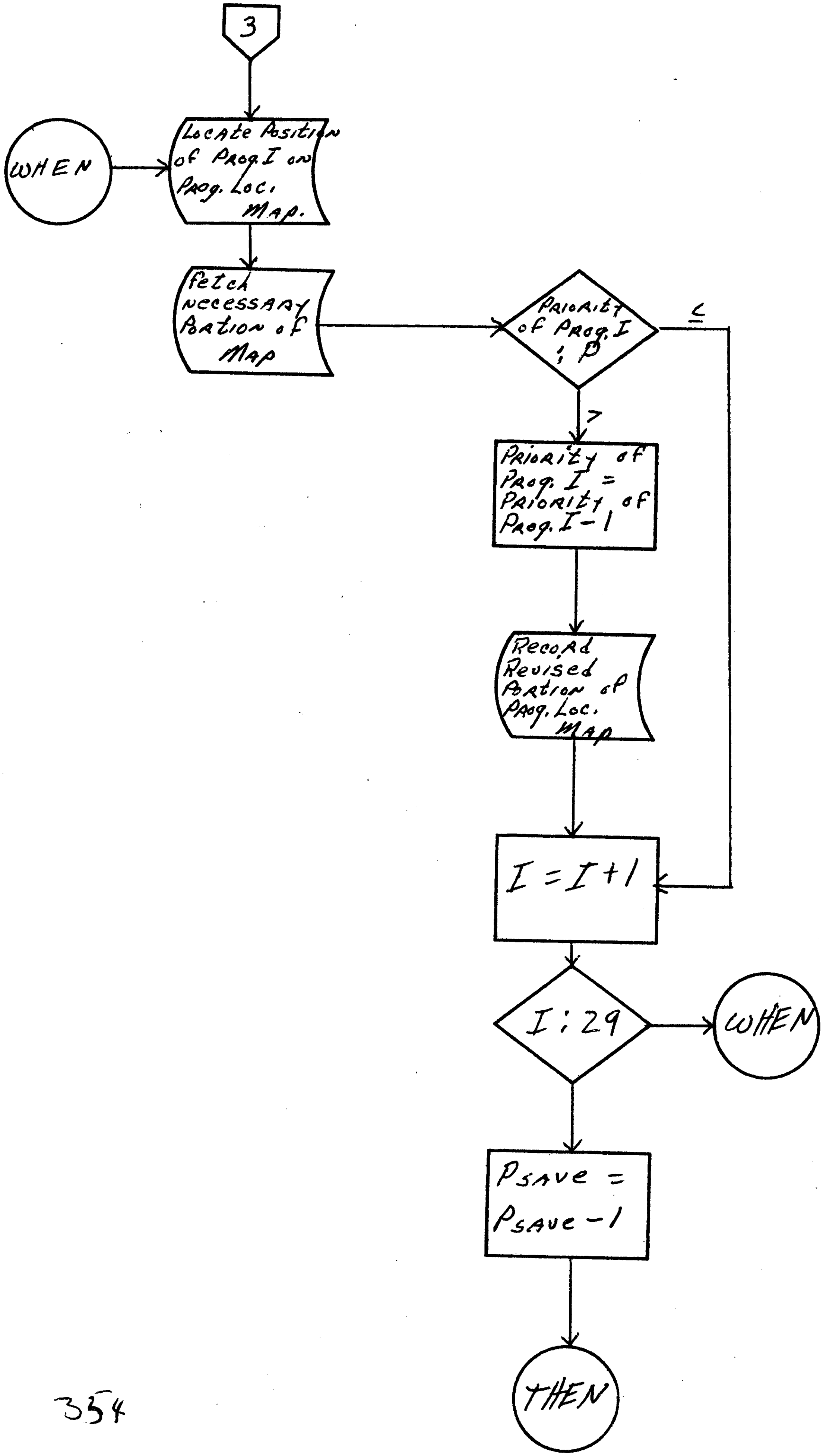

START
I=1
NIE
Locate position of Prog. I on Prog. Loc. MAP
Fetch: needed section of Prog. Loc. MAP
USE code AREA of PROG. I = 0
NO
yes
WRITE: I
write: use code of Prog. I
Write: Priority of Prog. I
I = I+1
I : 30
≤
NIE
>
BEG of utility Routine

NIF

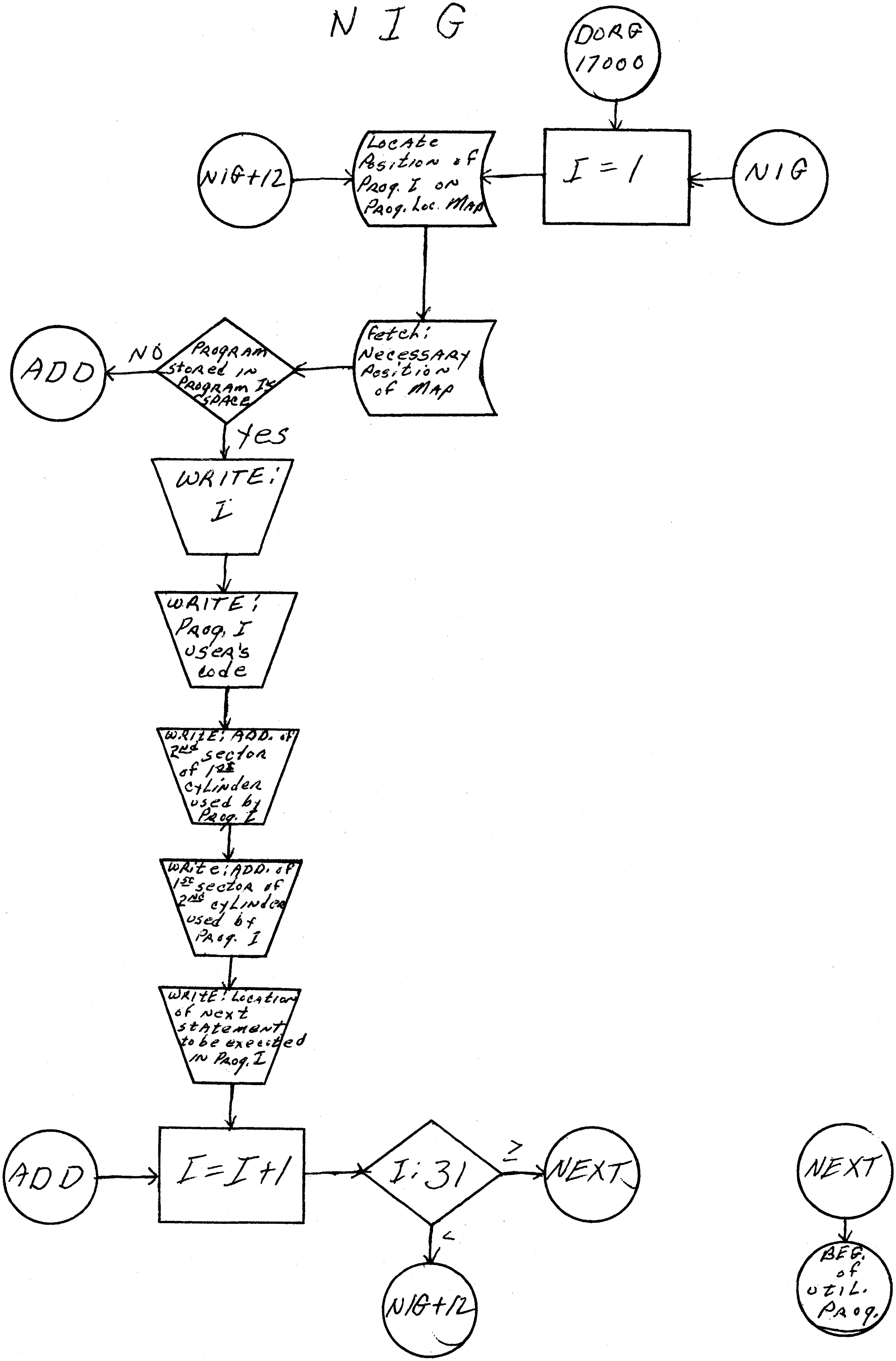

NIF
WRITE: "PROGRAM XX"
READ: I (XX = I)
CH
ON
OFF
# IN 1st digit of I
YES
NEXT
NO
# of digits in I > 2
WRITE: "OVER TYPE"
NO
PROG #I VALID (0<I<31)
WRITE: INVALID PROG. NUMBER
YES
Locate: Position of Prog. I on Prog. Loc. Map
Fetch: Necessary Portion of MAP
NIF
PROG. I STORED ?
NO
WRITE: "NOT STORED"
YES
WRITE: "PRIORITY XX"
Read: P (XX=P)
CH
ON
OFF
# in 1st digit of P
NEXT
NO
# of digits in P>2
YES
WRITE: "OVER TYPE"
1
NIF


1
P = PRIORITY # of PROG. I
NO
YES
NIF
P = 0
YES
NO
PRIORITY of PROG. I = 0
Record; Revised Portion of Prog. Loc. MAP
NIF
J = 1
Locate: Position of PROG. I on PROG. LOC. MAP
LOOK
Fetch. NECESSARY PORTION of MAP
PRIORITY # of PROG. J = P
NO
YES
INC
J = J+1
J : 29
>
≤
LOOK
PRIORITY of PROG. I = P
Record: Revised Portion of Prog Loc. MAP
NIF
INC
J = 1
INC+12
Locate Position of PROG. J on PROG. LOC. MAP
Fetch: NECESSARY PORTION of MAP.
PRIORITY of PROG. J : P
≥
<
PRIORITY # of PROG J = PRIORITY of PROG. J+1
Record Revised Portion of Prog Loc. MAP
J = J+1
J : 29
>
≤
INC+12
Fetch: Portion of MAP containing PROG. I
PRIORITY of PROG. I = P
Record Revised Portion of PROG. LOC. MAP
NIF
2

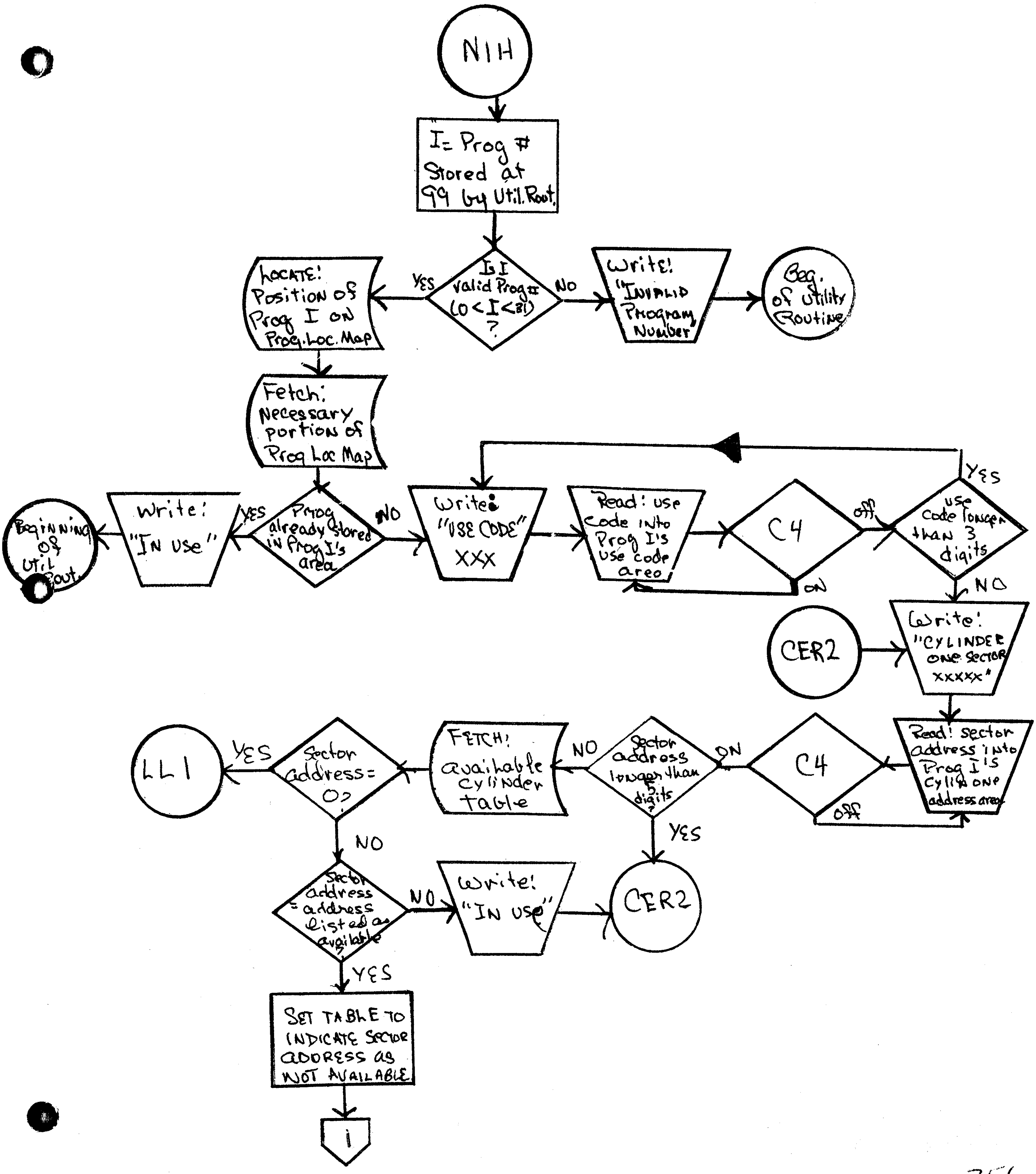

2
NEXT
P = 0
I = 1
Locate Position of Prog. I on Prog. Loc. Map
NEXT+24
Fetch: necessary Portion of Map
Priority of Prog. I : Psave
>
≤
Psave = Priority of Prog. I
I = I+1
I : 29
≥
NEXT+24
>
P = 0
P = P+1
P : Psave
>
BEG. of UTIL Routine
NOW+12
I = 1
THEN
Locate Position of Prog. I on Prog. Loc. Map
Fetch necessary Portion of Map
Priority of Prog. I = $P_0$
NO
YES
NOW+12
I = I+1
I : 29
≤
THEN
I = 1
3

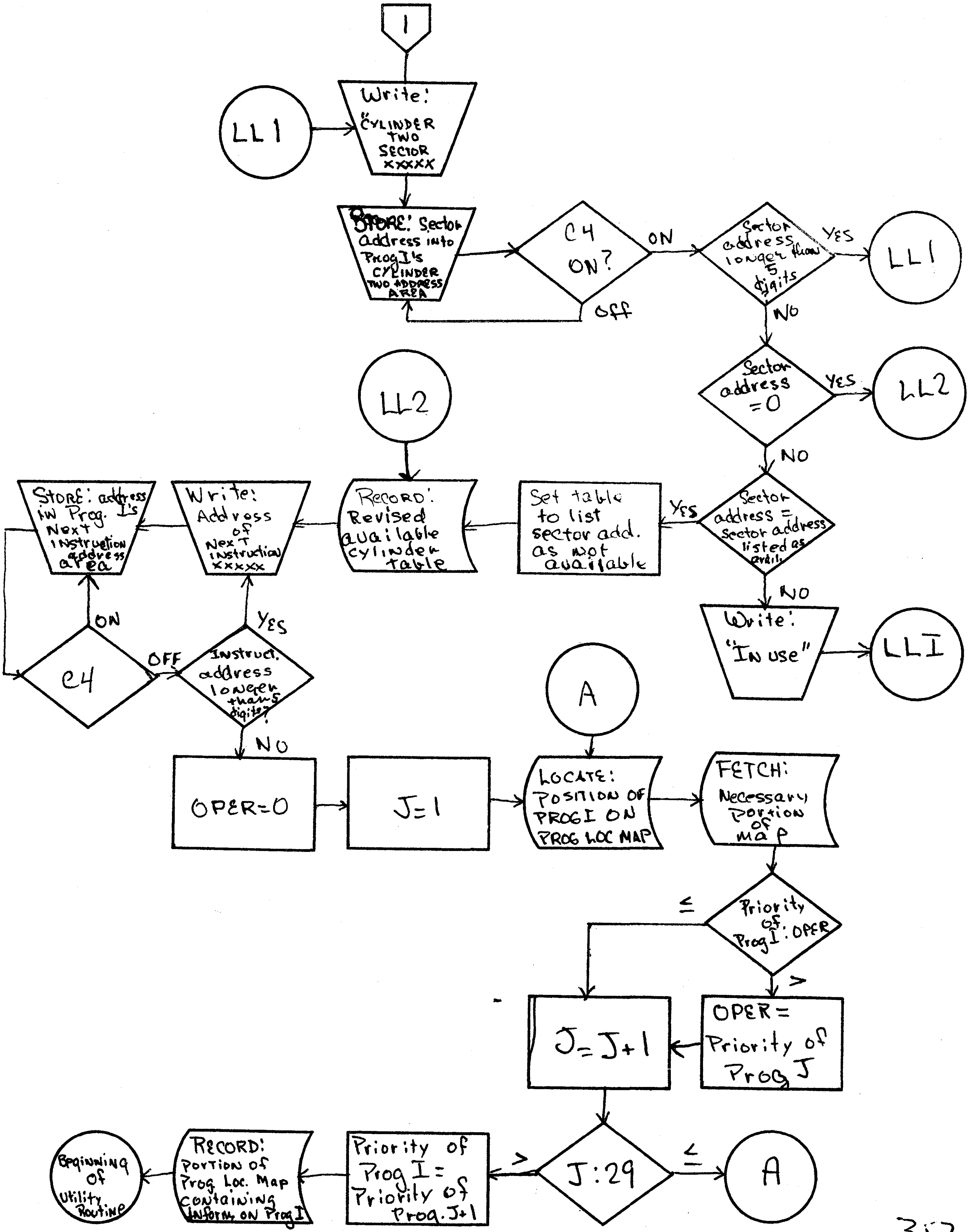

3
WHEN
Locate Position of Prog. I on Prog. Loc. Map.
Fetch necessary Portion of Map
Priority of Prog. I : P
≤
>
Priority of Prog. I = Priority of Prog. I-1
Record Revised Portion of Prog. Loc. Map
I = I+1
I : 29
WHEN
Psave = Psave -1
THEN

# N I G

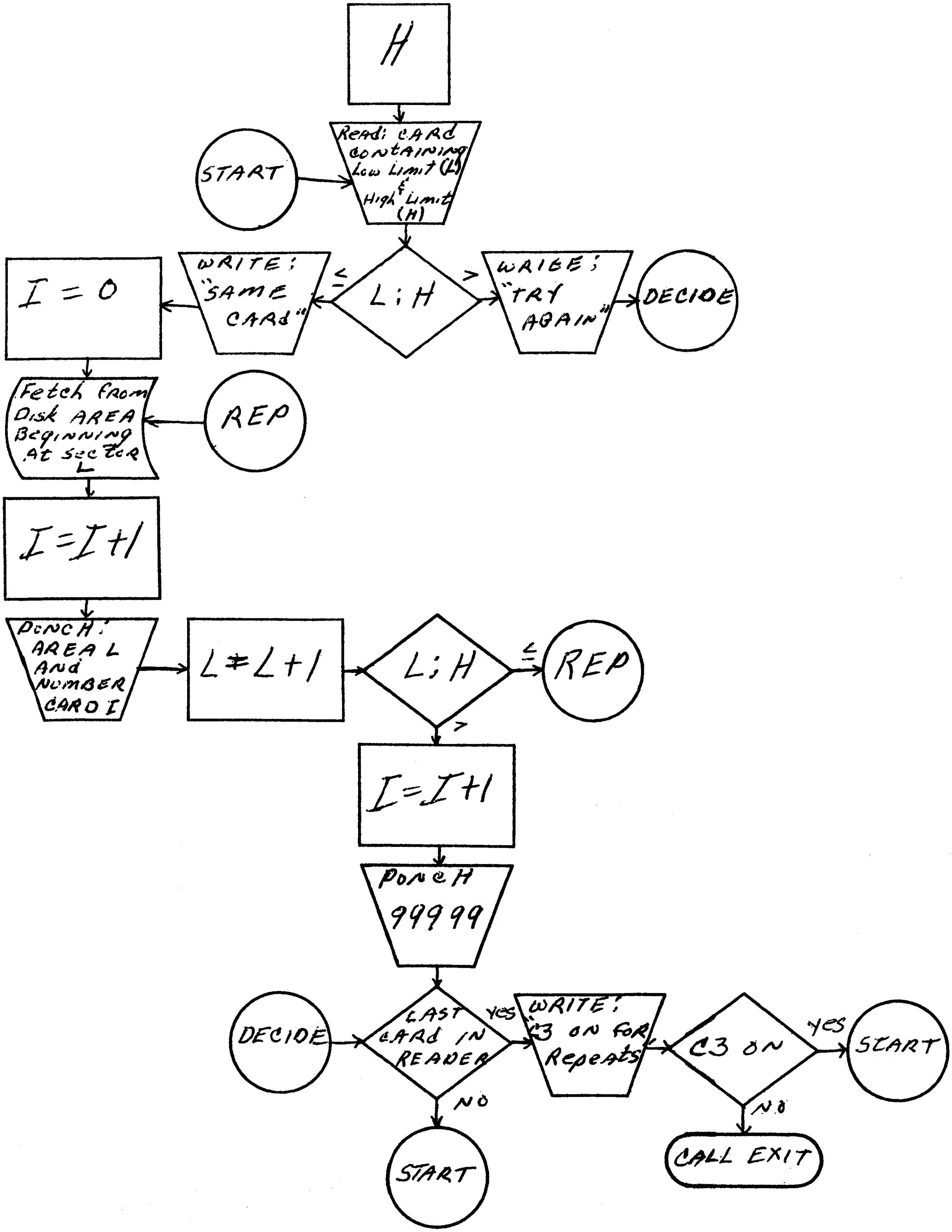

DORG 17000
I = 1
NIG
NIG+12
Locate Position of Prog. I on Prog. Loc. Map
Fetch: Necessary Position of Map
PROGRAM stored in PROGRAM I's SPACE
NO
ADD
Yes
WRITE: I
WRITE: Prog. I user's code
WRITE: ADD. of 2nd sector of 1st cylinder used by Prog. I
WRITE: ADD. of 1st sector of 2nd cylinder used by Prog. I
WRITE: Location of next statement to be execcted in Prog. I
ADD
I=I+1
I:31
≥
NEXT
<
NIG+12
NEXT
BEG. of util. Prog.

# NIH

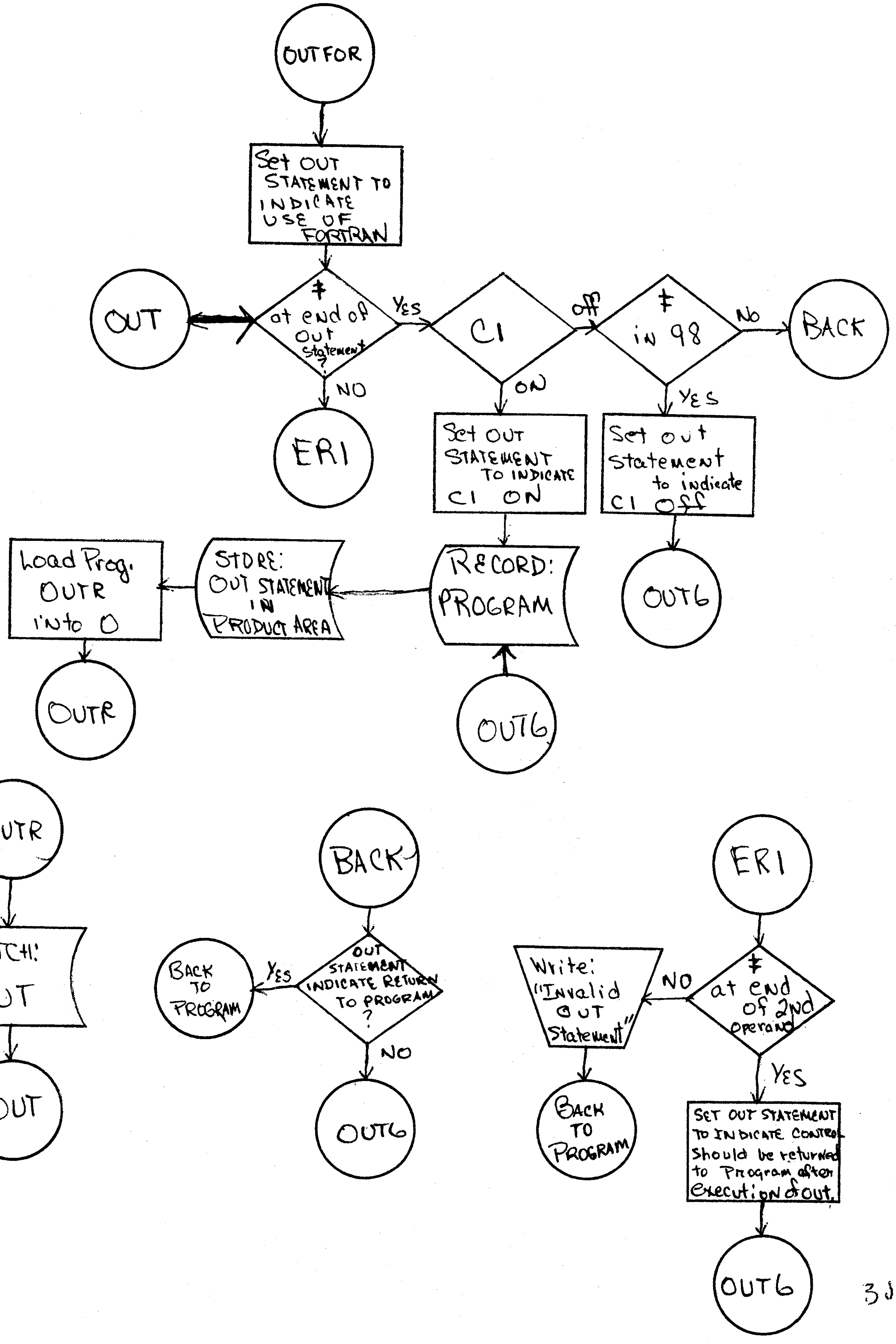

NIH
I= Prog # Stored at 99 by Util.Rout.
Is I valid Prog # (0<I<81) ?
YES
NO
LOCATE: Position of Prog I on Prog.Loc.Map
Write: "Invalid Program Number
Beg. of Utility Routine
Fetch: Necessary portion of Prog Loc Map
Prog already stored in Prog I's area
YES
NO
Write: "In use"
Beginning of Util Rout.
Write: "USE CODE" XXX
Read: use code into Prog I's use code area
C4
off
ON
use code longer than 3 digits
YES
NO
CER2
Write: "CYLINDER ONE SECTOR XXXXX"
Read: sector address into Prog I's cylin one address area
C4
ON
off
Sector address longer than 5 digits
NO
YES
CER2
FETCH: available cylinder table
Sector address = 0?
YES
LL1
NO
Sector address = address listed as available
NO
Write: "IN USE"
CER2
YES
SET TABLE TO INDICATE SECTOR ADDRESS AS NOT AVAILABLE
1


1
LL1
Write: "CYLINDER TWO SECTOR XXXXX
STORE: Sector address into Prog I's CYLINDER TWO ADDRESS AREA
C4 ON?
ON
Off
Sector address longer than 5 digits
YES
LL1
NO
Sector address =0
YES
LL2
NO
Sector address = Sector address listed as avail.
YES
Set table to list sector add. as not available
NO
Write: "In use"
LLI
LL2
Record: Revised available cylinder table
Write: Address of Next instruction XXXXX
Store: address in Prog. I's Next instruction address area
ON
C4
OFF
YES
Instruct. address longer than 5 digits?
NO
OPER=0
J=1
A
LOCATE: POSITION OF PROG I ON PROG LOC MAP
FETCH: Necessary portion of map
Priority of Prog I : OPER
≤
>
OPER = Priority of Prog J
J=J+1
J:29
≤
A
>
Priority of Prog I = Priority of Prog. J+1
RECORD: Portion of Prog. Loc. Map containing inform. on Prog I
Beginning of Utility Routine

# OUTPUT CARD

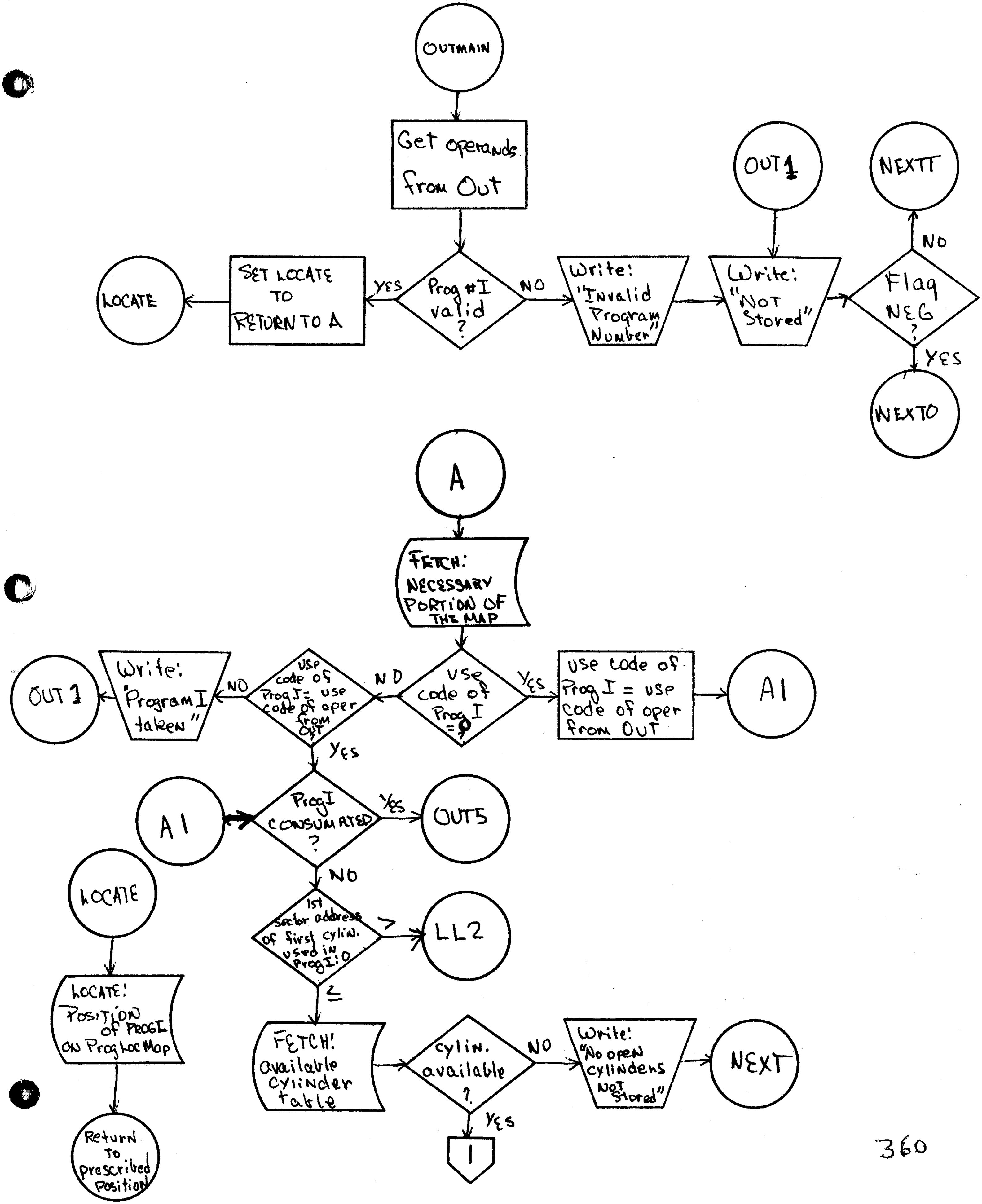

H
START
Read: CARD containing Low Limit (L) & High Limit (H)
L ; H
≤
>
WRITE: "SAME CARD"
WRITE: "TRY AGAIN"
DECIDE
I = 0
Fetch from Disk AREA Beginning At sector L
REP
I=I+1
PUNCH: AREA L And NUMBER CARD I
L=L+1
L ; H
≤
REP
>
I=I+1
PUNCH 99999
DECIDE
LAST CARD IN READER
YES
NO
START
WRITE: "C3 ON FOR Repeats"
C3 ON
YES
START
NO
CALL EXIT

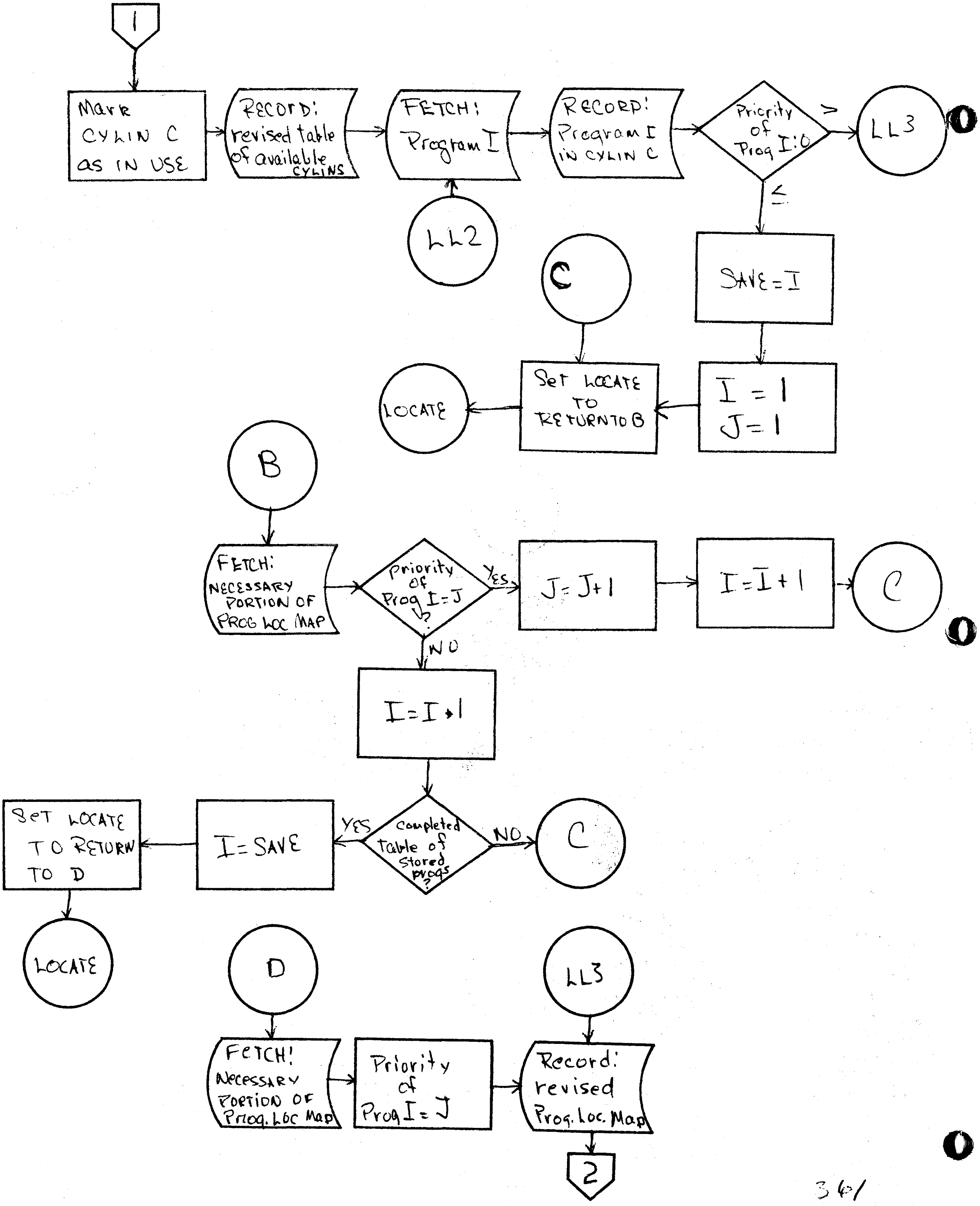
# OUT SUBROUTINE

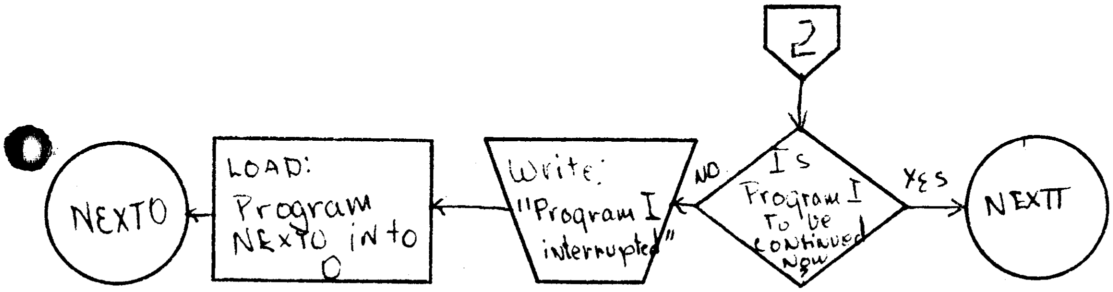


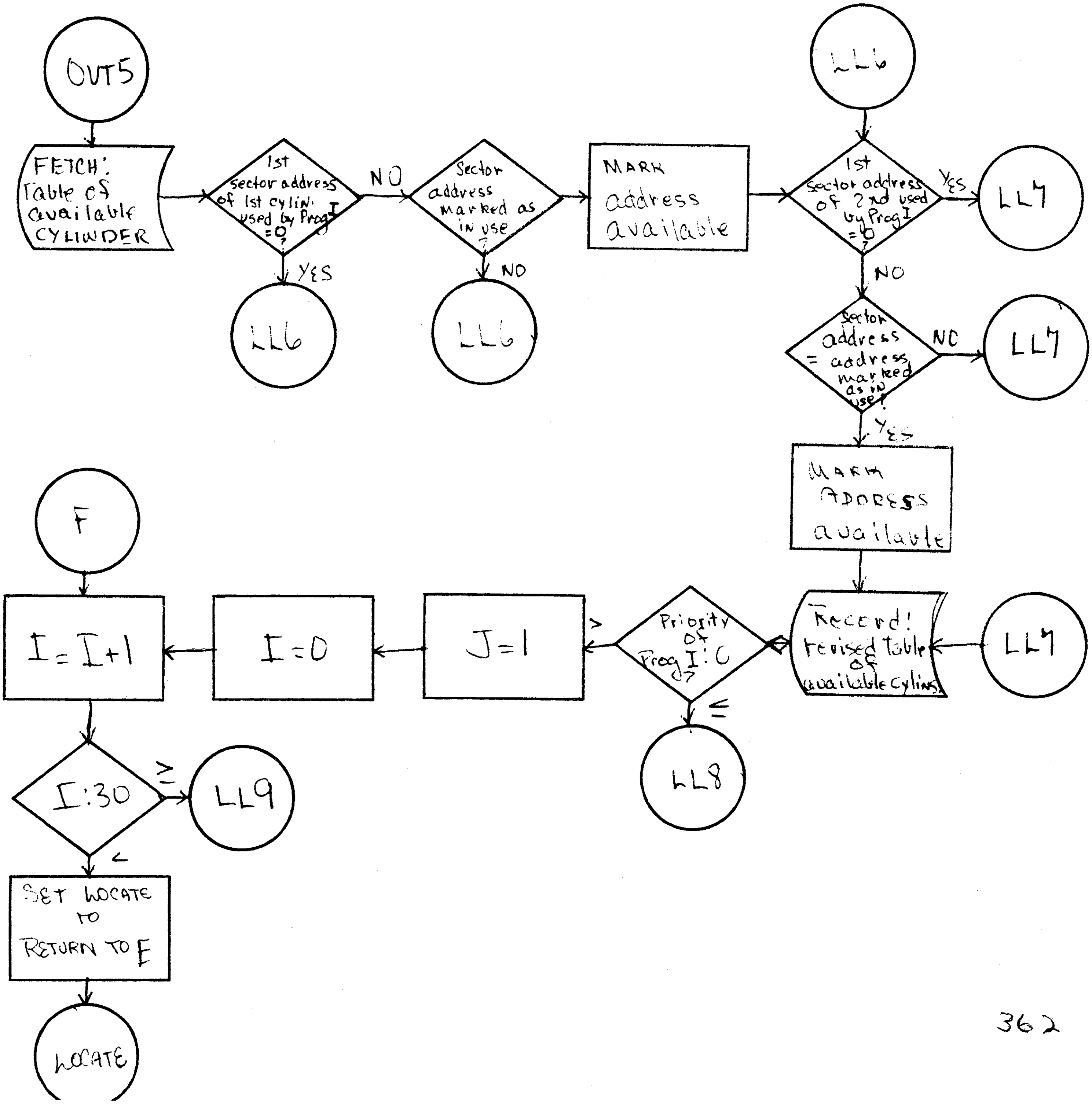
OUTFOR
Set OUT STATEMENT TO INDICATE USE OF FORTRAN
OUT
‡ at end of out statement ?
YES
NO
ERI
C1
off
ON
‡ in 98
No
BACK
YES
Set OUT STATEMENT TO INDICATE C1 ON
Set out statement to indicate C1 off
OUT6
RECORD: PROGRAM
STORE: OUT STATEMENT IN PRODUCT AREA
Load Prog. OUTR into O
OUTR
OUT6
OUTR
FETCH: OUT
OUT
BACK
OUT STATEMENT INDICATE RETURN TO PROGRAM ?
YES
BACK TO PROGRAM
NO
OUT6
ERI
‡ at end of 2nd operand
NO
Write: "Invalid OUT Statement"
BACK TO PROGRAM
YES
SET OUT STATEMENT TO INDICATE CONTROL Should be returned to Program after execution of out.
OUT6

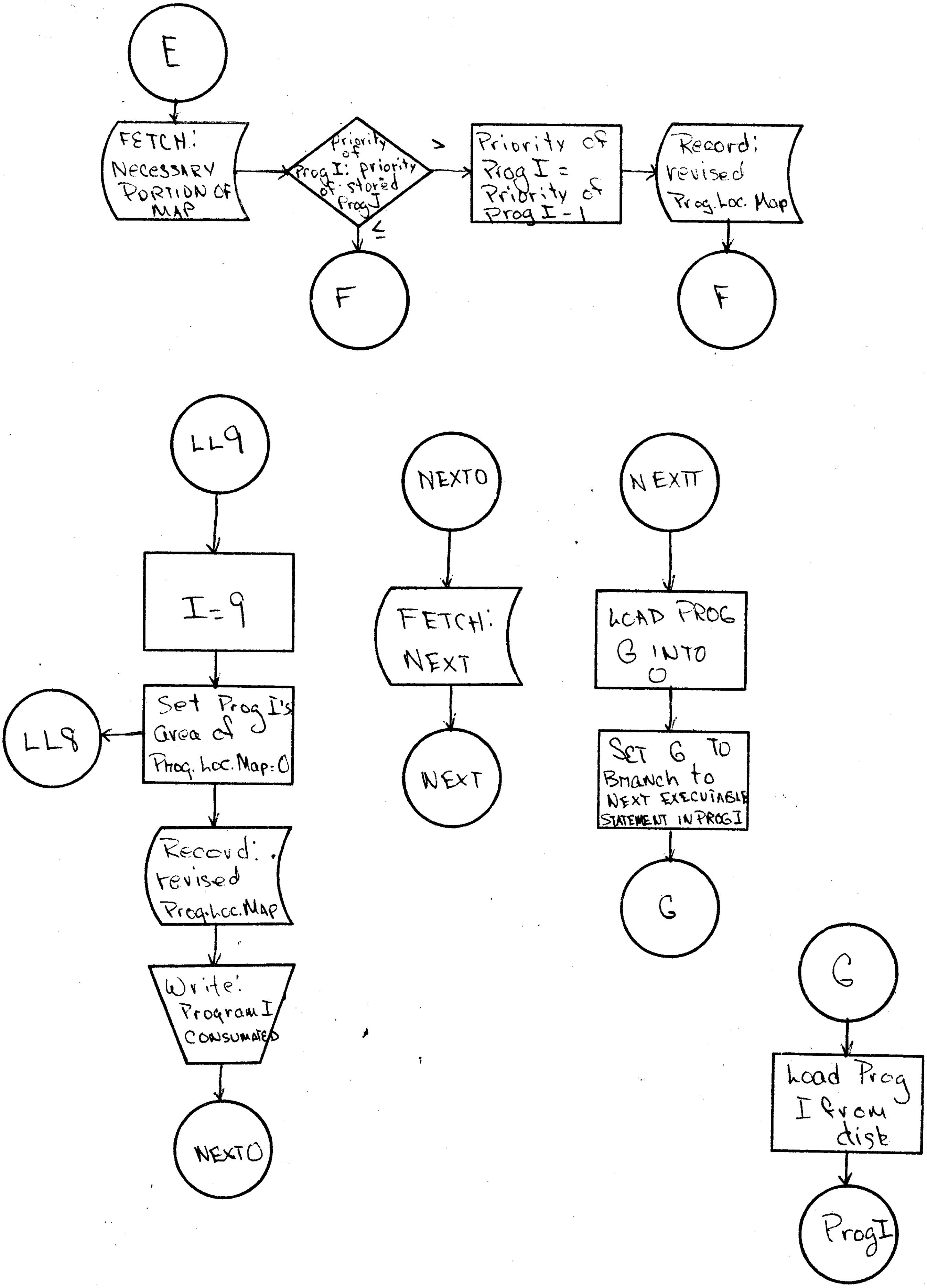
# OUT MAIN


OUTMAIN
Get operands from Out
Prog #I valid ?
YES
SET LOCATE TO RETURN TO A
LOCATE
NO
Write: "Invalid Program Number"
OUT1
Write: "Not Stored"
Flaq NEG ?
NO
NEXTT
YES
NEXTO
A
FETCH: NECESSARY PORTION OF THE MAP
Use code of Prog I =0 ?
YES
Use code of Prog I = use code of oper from OUT
A1
NO
Use code of Prog I = use code of oper from OUT
NO
Write: "Program I taken"
OUT 1
YES
A1
Prog I CONSUMATED ?
YES
OUT5
NO
1st sector address of first cylin. used in Prog I : 0
>
LL2
≤
FETCH! available cylinder table
cylin. available ?
NO
Write: "No open cylinders Not Stored"
NEXT
YES
1
LOCATE
LOCATE! POSITION of PROG I ON Prog Loc Map
Return To prescribed Position


1
Mark CYLIN C as IN USE
RECORD: revised table of available CYLINS
FETCH: Program I
LL2
RECORD: Program I in CYLIN C
Priority of Prog I:0
>
LL3
≤
SAVE=I
I = 1
J = 1
C
Set LOCATE TO RETURN TO B
LOCATE
B
FETCH: NECESSARY PORTION OF PROG LOC MAP
Priority of Prog I=J ?
YES
J=J+1
I=I+1
C
NO
I=I+1
Completed Table of Stored progs ?
YES
NO
C
I=SAVE
SET LOCATE TO RETURN TO D
LOCATE
D
FETCH! NECESSARY PORTION OF Prog. Loc Map
Priority of Prog I = J
LL3
Record: revised Prog. Loc. Map
2


2
Is Program I to be continued now
YES
NEXTT
NO
Write: "Program I interrupted"
LOAD: Program NEXTO into 0
NEXTO


OUT5
FETCH: Table of available CYLINDER
1st sector address of 1st cylin. used by Prog I =0?
NO
YES
LL6
Sector address marked as in use?
NO
LL6
MARK address available
LL6
1st sector address of 2nd used by Prog I =0?
YES
LL7
NO
Sector address = address marked as in use?
NO
LL7
YES
MARK ADDRESS available
Record: revised table of available cylins.
LL7
Priority of Prog I : 0?
>
≤
LL8
J=1
I=0
F
I=I+1
I:30
≥
LL9
<
SET LOCATE TO RETURN TO F
LOCATE


E
FETCH: Necessary Portion of Map
Priority of Prog I: priority of stored Prog J
>
≤
Priority of Prog I = Priority of Prog I - 1
Record: revised Prog. Loc. Map
F
F
LL9
I=9
Set Prog I's area of Prog. Loc. Map: 0
LL8
Record: revised Prog. Loc. Map
Write: Program I Consumated
NEXTO
NEXTO
FETCH: NEXT
NEXT
NEXTI
LOAD PROG G INTO 0
SET G TO Branch to NEXT EXECUTABLE STATEMENT IN PROG I
G
G
Load Prog I from disk
ProgI

# READF - A Free Format Read Subroutine For FORTRAN II-D

Presented by
Robert P. Bair
Elliott Company
Division of Carrier Corporation
Jeannette, Pennsylvania

1620 Users Group
Eastern-Midwestern Joint Conference
Americana Hotel
New York City, N. Y.

364

Free format input is the ability to read data without specifying its form or position in the input record. Free format may be either order-dependent or order-independent. Order-independent input allows a variable name and its value to be entered as data, so that any number of variables, in any order, may be given. This is the most flexible method of input, but it cannot be implemented in FORTRAN II because it requires a symbol table in core with which to identify the variable names. The MAD language uses this technique in its READ DATA statement. Order-dependent input requires a list of variables in the input statement, and the data then consists of values for these variables in the same order. The advantage over the standard fixed format input is that the amount of space between the data is variable, and more freedom is permitted in the form of the numbers. This type of free format may be used with the FORGO compiler by omitting the format number in the input statement, but it can be done in FORTRAN II only by a subroutine like this one.

READF is called like any other FORTRAN subroutine, and the list of variables to be read is given as arguments. The number of variables in the list must be odd, for reasons to be explained later. The first time it is called, the first input card is read and successive numbers are assigned to successive variables. When all the variables in the statement have been defined, the subroutine returns to the main-line program. The next time it is called, the scan of the first card is continued, and each number that is found is assigned to a variable. A new card is read only when the last one has been completely scanned, and any number of cards may be read in order to define all the variables listed.

Since there is no way to determine in a subroutine, the mode of any of the arguments in the CALL statement, the mode of all the arguments is assumed and only normalized floating point numbers are returned. READF actually stands for "READ Floating".

Numbers may be separated from each other by one or more non-numeric characters, except for a single decimal point. This is as flexible as it could possibly be made, and results in the ability to read and distinguish between any sequence of numbers that could be interpreted by a human. If two decimal points are found together, they will be used as a separating point between two numbers. Any number which extends to column 80 is assumed to end there. All alphabetic characters are ignored, except for E which may indicate an exponent. The only special characters which have an effect on the result are the decimal point, and the dash which indicates a negative number.

When a number is entered with an exponent, two separate numbers, the mantissa and the exponent, must be evaluated as one. This is a special case, and the following restriction is required to keep the numbers from being treated separately. The mantissa must be followed immediately by the letter E, which is followed by the exponent in one of the following possible forms:

+nn, -nn, nn, +n, -n, n

No blanks are permitted anywhere within the complete mantissa-exponent combination. If a number is followed by E, which in turn is not followed by one of the forms above, no exponent results. If the exponent form is correct, but more than two digits are given, an error message will be printed, and the third digit will start the next number.

The problems of communicating with the subroutine can best be explained by reviewing the linkage that is generated. The statement CALL READF (A) is compiled into the following:

```
BTM        -READF        ,*+11

DSA        A

DSC        1,0
```

where READF is a five digit field which contains the address of the first executable instruction in the subroutine. The constant, an unflagged zero, is required so that the next instruction starts in an even address. If there are two arguments in the CALL statement, two symbolic addresses of five digits each would be generated, and the constant would not appear. Thus any time that an odd number of arguments is given in the CALL statement, an unflagged zero is produced before the next instruction.

Since we would like to be able to give a variable number of arguments, we are faced with a decision between several alternate methods of indicating the number of arguments, or the length of the linkage. (1) The first argument may be an integer having a value equal to the number of arguments to follow. (2) The first floating-point variable in the list could be pre-set to the floating value equal to the number of arguments. (3) If an odd number of arguments is given, the unflagged zero which is automatically placed in the compiled linkage could be used to determine the number of arguments. (4) If the CALL statement were always followed by a PAUSE statement, a single instruction, halt, would be compiled at the end of the linkage to READF and could be recognized like the unflagged zero to signal the end of the arguments. The halt would then be by-passed.

Of these four methods for indicating the number of arguments, I have chosen the third, requiring that the number be odd, because it requires the least amount of extra effort on the part of the programmer. In many instances, there will be only one argument when READF is used in a DO loop to enter an array. The third method is certainly the most convenient here. In those cases where many different variables are to be read, any odd number can be specified in one statement. If an even number is desired, it can be broken up into two statements that both have an odd number of arguments. Because READF does not read a new card every time it is called, no information will be lost, and there is no difference whether one CALL statement is used or ten.

If the argument of a CALL statement is a subscripted variable, the 5-digit address generated after the BTM in the linkage is an indirect address. This must be allowed for in the subroutine when it determines the address of the argument. Another problem is that there is no way to tell from the compiled linkage whether or not any of the arguments are un-subscripted array names. If an array name is used, the address generated is simply that of its first element. The DIMENSION statement in a FORTRAN-written subroutine is necessary because of this ambiguity.

The subroutine's task of deciphering data is not a trivial one because of the many different forms a number may take. When considering a number like 002, one may form the rule that leading zeros are always ignored. But if this rule is followed, there is no way to enter a zero. Generally, the start of a number may be either a digit or a decimal point.

As READF scans from left to right across a number, it must adjust the exponent of the result, and transmit the digits, one at a time, into the mantissa of the result. The amount by which the exponent is adjusted for each digit depends on whether or not the digit is significant, and on its position in relation to the decimal point. The mantissa of the result is to be between 1 and 0, inclusive, so if the decimal point is preceeded by n significant digits, the final exponent will be equal to n. If there are n zeros after the decimal point before a significant digit, then the exponent equals -n.

READF is programmed in two loops. The first scans for a minus sign,digit or decimal point, which start a number, ignoring all other characters. If a minus sign is found, an indicator is set, which is cleared again unless a digit or decimal point follows immediately. When a digit is found, the second loop is started to test for significant digits, increment the exponent, build the mantissa, and test for a decimal point.

As each digit is placed into the mantissa of the result, the present value of K is added to the exponent which was initially zero. K is changed by the following rule: start with K equal to zero. When a decimal point is found, subtract 1 from K. Add 1 to K when the first non-zero digit is encountered. Thus, in the example 00123.45, the value of K and the exponent are both still zero until the 1 is found. K is then incremented to +1. When the decimal point is reached, the exponent will be +3, and then K is set back to 0, so that the remaining digits have no effect on it. In the number 0.0000823, K is set equal to -1 when the decimal point is reached, and the exponent is -4 by the time the 8 is reached. This is the first non-zero digit, so K is incremented to 0 and the exponent remains unchanged.

In order to produce floating point numbers which are already normalized, no leading zeros are transmitted into the mantissa. Transmission starts with the first non-zero digit, and proceeds for a number of digits equal to the mantissa length. If an exponent follows the number, it is simply added on to the exponent produced from the mantissa.

A routine within the subroutine advances the indirect address that points to the character being tests, sets flags to divide the input up into 2-digit fields, and tests for the record mark stored after the input area. If READF were converted to read paper tape, the routine would find the record mark made by the end-of-line.

READF is written as a variable-length subroutine, internally adjusting for the mantissa length being used in order to return floating-point results of the proper length. The value of the matissa length, f, is stored in the communications area of MONITOR. When READF is compiled, however, the header record must specify the value of f and k for which the subroutine is to be used. These figures are stored on the disk ahead of the program and are checked by the FORTRAN loader, phase 3, before the subroutine is loaded. The header record is all that restricts the subroutine to one value of f and k, and is the only thing that has to be changed when a different length is desired.

READF will greatly reduce the number of cards required for a given amount of data by permitting the numbers to be placed close together. It will also reduce the amount of time required to punch the cards, since there are such few restrictions on the card punch operator. Helpful comments, names of units, or instructions may be inserted anywhere to explain the data. Most important, there is no need to worry about an error resulting from something in the wrong card columns. If you can read it, so can READF, and it is well suited to every program from introductory FORTRAN to advanced applications.

READF is programmed in two loops. The first scans for a minus sign,digit or decimal point, which start a number, ignoring all other characters. If a minus sign is found, an indicator is set, which is cleared again unless a digit or decimal point follows immediately. When a digit is found, the second loop is started to test for significant digits, increment the exponent, build the mantissa, and test for a decimal point.

As each digit is placed into the mantissa of the result, the present value of K is added to the exponent which was initially zero. K is changed by the following rule: start with K equal to zero. When a decimal point is found, subtract 1 from K. Add 1 to K when the first non-zero digit is encountered. Thus, in the example 00123.45, the value of K and the exponent are both still zero until the 1 is found. K is then incremented to +1. When the decimal point is reached, the exponent will be +3, and then K is set back to 0, so that the remaining digits have no effect on it. In the number 0.0000823, K is set equal to -1 when the decimal point is reached, and the exponent is -4 by the time the 8 is reached. This is the first non-zero digit, so K is incremented to 0 and the exponent remains unchanged.

In order to produce floating point numbers which are already normalized, no leading zeros are transmitted into the mantissa. Transmission starts with the first non-zero digit, and proceeds for a number of digits equal to the mantissa length. If an exponent follows the number, it is simply added on to the exponent produced from the mantissa.

A routine within the subroutine advances the indirect address that points to the character being tests, sets flags to divide the input up into 2-digit fields, and tests for the record mark stored after the input area. If READF were converted to read paper tape, the routine would find the record mark made by the end-of-line.

READF is written as a variable-length subroutine, internally adjusting for the mantissa length being used in order to return floating-point results of the proper length. The value of the matissa length, f, is stored in the communications area of MONITOR. When READF is compiled, however, the header record must specify the value of f and k for which the subroutine is to be used. These figures are stored on the disk ahead of the program and are checked by the FORTRAN loader, phase 3, before the subroutine is loaded. The header record is all that restricts the subroutine to one value of f and k, and is the only thing that has to be changed when a different length is desired.

READF will greatly reduce the number of cards required for a given amount of data by permitting the numbers to be placed close together. It will also reduce the amount of time required to punch the cards, since there are such few restrictions on the card punch operator. Helpful comments, names of units, or instructions may be inserted anywhere to explain the data. Most important, there is no need to worry about an error resulting from something in the wrong card columns. If you can read it, so can READF, and it is well suited to every program from introductory FORTRAN to advanced applications.

# BIBLIOGRAPHY

1. Bailey, M. J., Barnett, M. P., and Futrell, R. P., "Format Free Input in Fortran". Communications of the ACM, 6 (October, 1963), 605-608.

2. IBM 1620 Monitor I System Reference Manual. White Plains, New York. International Business Machines Corporation. Form No. C26-5739-3.

```
ZZJOB                           31177001SYSTEM DEVELOPMENT
ZZFORX
      DIMENSION A(3)
      I=3
C     READF MUST HAVE AN ODD NUMBER OF ARGUMENTS
    2          CALL READF (A(1),A(2),B,A(I),C)
      PUNCH 1, A(1),A(2),B,A(I),C
    1     FORMAT (    5E16.8)
      GO TO 2
      END
TEST DATA FOR READF
9/4/65
FOLLOWING ARE TWO BLANK RECORDS WHICH PRESENT NO DIFFICULTY


THE NEXT LINE INDICATES VARIOUS METHODS OF SEPARATING NUMBERS
     1    2,3  4AND5  6+7-8/9+10      -11-12
ZERO CAN BE ENTERED SEVERAL WAYS
0    0000   0.0    .000     0.          0
NUMBERS CAN BE SURROUNDED BY LETTERS
INPUT13=.5UNLESS A .G. B+3.65+D
NUMBERS MAY BE ANY LENGTH, BUT ONLY F SIGNIFICANT DIGITS ARE USED.
1234567890123456788.9012345    0.00000000012345678901234
                    1234.5678901
USE A MINUS SIGN FOR NEGATIVE NUMBERS
-0.00567   -9843.67    -0    -598      -1-2.230 -0000.7
MINUS SIGNS NOT FOLLOWED BY A DIGIT ARE IGNORED
    - 1.   -1.    - .2    -.2    - 0.3    -0.3
PLUS SIGNS ARE IGNORED
+1      1     +.8    .8    +-9    -+9   (THE SIGN CLOSER TO THE
                                          NUMBER COUNTS)
TWO DECIMAL POINTS IN A ROW INDICATE THE END OF A NUMBER
1234..5678     1234. .5678    .123......56     .123. 56


READF CAN HANDLE NUMBERS WITH AN EXPONENT JUST AS WELL
THE EXPONENT MUST IMMEDIATELY FOLLOW THE NUMBER.     12.59E33
    6.123 E66    'IS TWO SEPARATE NUMBERS. SO IS   .9949E 2
NO BLANKS, PLEASE.   21.E-2    6E+30    829645E-45   .5648956E12    MANTISSA
CAN BE INTEGER.  21.45E9.1118E-6+3.E50-9         .000E25   0.00E70  ZERO
ALWAYS COMES OUT THE SAME      E23    1E23

A LEADING RECORD MARK CAUSES A RE-READ
Z   1111111
22222222

TWO RECORD MARKS CAUSE A CALL EXIT... BUT FIRST,
HERE IS AN EASY WAY TO ENTER AN ARRAY
A(4,5) = 4.23
ZZEND OF TEST 1
```

```
   .90000000E+01    .40000000E+01    .65000000E+02    .10000000E+01    .20000000E+01
   .30000000E+01    .40000000E+01    .50000000E+01    .60000000E+01    .70000000E+01
  -.80000000E+01    .90000000E+01    .10000000E+02   -.11000000E+02   -.12000000E+02
   .00000000E-99    .00000000E-99    .00000000E-99    .00000000E-99    .00000000E-99
   .00000000E-99    .13000000E+02    .50000000E+00    .36500000E+01    .12345678E+19
   .12345678E-09    .12345678E+04   -.56700000E-02   -.98436700E+04   -.00000000E-99
  -.59800000E+03   -.10000000E+01   -.22300000E+01   -.70000000E+00    .10000000E+01
  -.10000000E+01    .20000000E+00   -.20000000E+00    .30000000E+00   -.30000000E+00
   .10000000E+01    .10000000E+01    .80000000E+00    .80000000E+00   -.90000000E+01
   .90000000E+01    .12340000E+04    .56780000E+00    .12340000E+04    .56780000E+00
   .12300000E+00    .56000000E+00    .12300000E+00    .56000000E+02    .12590000E+35
   .61230000E+01    .66000000E+02    .99490000E+00    .20000000E+01    .21000000E+00
   .60000000E+31    .82964500E-39    .56489560E+12     21450000E+11    .11180000E-06
   .30000000E+51   -.90000000E+01    .00000000E-99    .00000000E-99    .23000000E+02
   .10000000E+24    .22222222E+08    .40000000E+01    .50000000E+01    .42300000E+01
```

# FORTRAN Compilation of CALL Statement

```
CALL  READF  (A)
```

is compiled into

```
BTM   -READF   ,*-11
DSA   A
DSC   1        ,0
```

where READF is the address of the subroutine

3) Determine the maximum operating rate of a machine when the flywheel size and motor size are specified.

4) Determine the maximum operating rate of a machine and the required flywheel inertia when the motor size is specified.

5) Determine the dynamic behavior of a machine when the flywheel inertia, motor size and operating rate are specified.

## DYNAMIC ANALYSIS

Analysis of the dynamic behavior of a machine involves determining motor speed as a function of machine position in the cycle. The approach used to determine speed versus position can be based on either solution of the differential equation of motion or on conservation of energy through a displacement interval. Because it is numerical and especially suitable for computer application, the procedure employing energy considerations is used in this analysis. The required energy relationships can be determined, but it is first necessary to consider numerical representation of the torque demand of a machine and of the speed-torque relationship of a motor.

### Torque Demand of a Machine

The torque requirement of a machine includes work loads, friction loads and inertia loads. By inertia loads are meant those due to acceleration and deceleration of machine components when the machine is operating at constant speed. The total torque demand of a machine can be determined at any position by summing the torque requirements of the various machine components at their respective phase angles.

For reciprocating machines the torque demand is not a linear function of machine angle. However, in this analysis, the torque will be represented by a series of straight-line functions. This approximation is made because, in most applications, the torque demand curve will not be defined to sufficient accuracy to warrant a more sophisticated method of representation. A typical torque demand curve is shown in Figure 2.

The torque demand curve determines the energy required by the machine when the motor shaft rotates through an angular increment, $\Delta\theta$ . By considering small increments the energy requirement, W( θ ), will be very nearly equal to the product of average torque and angular displacement. Thus,

$$W(\theta) = \frac{T(\theta_i) + T(\theta_f)}{2} \cdot \Delta\theta$$

376

## Speed-Torque Curve of a Motor

The operating characteristics of NEMA design B motors are such that with increasing torque and decreasing speed, the motor will stall at breakdown speed and continuous operation will be interrupted. Therefore, when considering the continuous operation of a machine, the operating range will include only that portion of the speed-torque curve above breakdown speed. A typical motor speed-torque curve is given in Figure 3. In this range the speed-torque curve can be represented by an exponential function, $y = Ax^E$, between breakdown speed and rated speed and by a straight-line between rated speed and synchronous speed. The energy supplied by the motor, through some interval of rotation, can then be developed as follows:

The equation for motor torque in the exponential function speed range is:

$$TQ(\theta) = \left[ TQ(R) - TQ(B) \right] \cdot \left[ \frac{N(\theta) - N(B)}{N(R) - N(B)} \right]^E + TQ(B)$$

where,

$$E = \frac{\ln \left[ TQ(B) - TQ(R) \right] - \ln \left[ TQ(B) - TQ(I) \right]}{\ln \left[ N(R) - N(B) \right] - \ln \left[ N(I) - N(B) \right]}$$

while motor torque in the linear speed range is given by:

$$TQ(\theta) = TQ(R) \cdot \frac{N(\theta) - N(R)}{N(R) - N(S)} + TQ(R)$$

where:

$TQ(\theta)$ = motor torque at angle $\theta$
$TQ(B)$ = motor torque at breakdown
$TQ(I)$ = motor torque at intermediate point
$TQ(R)$ = rated torque of motor
$N(\theta)$ = motor speed at angle $\theta$
$N(B)$ = motor speed at breakdown
$N(I)$ = motor speed at intermediate point
$N(R)$ = rated speed of motor
$N(S)$ = synchronous speed of motor

Either the exponential or linear equation is used to calculate motor torque depending upon whether speed, $N(\theta)$, is less or greater than rated speed, respectively. In either case, the energy supplied by the motor, $M(\theta)$, during an interval, $\Delta\theta$, is:

$$M(\theta) = \frac{TQ(\theta_i) + TQ(\theta_f)}{2} \cdot \Delta\theta$$

With the relationships developed for energy required and energy supplied over an increment of displacement, the procedure used for the dynamic analysis can be stated. Speed at successive increments in the cycle is calculated on the basis that the change of kinetic energy equals the energy supplied by the motor minus that required by the machine. Thus,

$$KE(\theta_f) = KE(\theta_i) + M(\theta) - W(\theta)$$

where,

$$KE(\theta) = 1/2 \cdot J \cdot N(\theta)^2$$

and,

$J$ = total mass moment of inertia referred to the motor shaft

By applying this basic equation at successive increments, the kinetic energy, and consequently motor speed, can be determined at each point in the cycle. To obtain a continuous solution, an iterative technique is used, whereby the speed at the beginning of the cycle is replaced by that at the end of the cycle. When the motor speed at the end of the cycle converges to that at the beginning, the speed versus position data will describe the dynamic behavior of a machine.

## EVALUATION OF DYNAMIC BEHAVIOR

Speed fluctuation and maximum slowdown can be calculated directly from the results of the dynamic analysis. The dynamic behavior, for a particular flywheel and motor (or machine operating rate), is evaluated by comparing these calculated values to specified design limits. In addition to providing acceptable dynamic behavior, a motor must be sized such that its temperature rise during normal operation does not exceed the limits of its insulation. Because the ability of a motor to dissipate heat at a given maximum temperature is constant, the criterion of temperature rise is controlled by the amount of heat produced in the electrical windings of the motor. Thus, in order to meet this temperature rise condition, it is necessary that the heat produced by the motor during a cycle of operation does not exceed that which would be produced at rated output.

The heat energy produced in an electric circuit per increment of time, dt, is:

$$HEAT = I^2 \cdot R \cdot dt$$

Since the resistance, R, in a motor circuit is essentially constant during continuous operation, the heat created, and consequently temperature rise, can be compared on the basis of root mean square current. The requirement for no overheating is then:

$$\text{Rated Current} > I_{rms} = \sqrt{\sum_{n=1}^{N} (I_n^2 \cdot \Delta t_n)/T}$$

where

$I$ = motor current
$\Delta t$ = time increment
$T$ = time per cycle
$N$ = number of increments per cycle

## PROGRAM LOGIC

For a given motor and flywheel size and machine operating rate, the program calculates motor speed for each displacement increment in the cycle. Speed fluctuation, maximum slowdown and rms current are then calculated to describe the dynamic behavior as given by the speed versus displacement calculations. The dynamic behavior is evaluated by testing these values against specified design limits. The sequence for testing these criteria is given in the Simplified Logic Diagram of Figure 4.

Necessary changes are first made to obtain acceptable dynamic behavior of the machine. These changes are made after testing maximum slowdown and speed fluctuation of the motor. When speed fluctuation is excessive, additional flywheel inertia is required to provide a more nearly constant speed. If the program input specifies that the flywheel inertia cannot be changed, motor size will be increased to reduce the speed fluctuation. Where maximum slowdown from rated speed is excessive, but speed fluctuation is not, additional horsepower is required. In this situation, motor size is increased and flywheel inertia is reduced to a value such that maximum allowable speed fluctuation exists.

It should be noted that the type of analysis required, as specified in the program input, selects either motor horsepower or maximum machine operating rate as a quantity to be determined. In an analysis to determine the maximum operating rate of a machine, the motor size is fixed, and horsepower changes, as previously specified, are replaced by inverse changes in operating rate. This in effect changes the power requirements of the machine in place of changing the required power necessary from the motor. After each change in component size or machine operating rate, the dynamic analysis and all subsequent calculations and tests are repeated.

After a flywheel and motor size (or machine operating rate)has been determined that will satisfy the dynamic requirements, the rms current effect on motor heating is an additional requirement to be satisfied.

Flywheel inertia and machine speed, if a program variable, is changed by increments as specified in the program input. The program also provides that flywheel inertia can be limited to given minimum and maximum values.

When the computer has determined the motor and flywheel requirements of a given machine, it will list motor speed, percent speed variation from rated speed, torque demand, motor torque and motor current for each degree of machine angle. This data is also given in 10 degree increments in an output summary which includes motor size, flywheel inertia, machine speed and other design information.

Input necessary for use of the program analysis includes torque demand of the machine, operating characteristics for the range of motors being considered, machine inertia and program control data.

The program is written in Fortran II and requires a 1620 system with 40,000 units of memory and a 1311 disk drive.

## BIBLIOGRAPHY

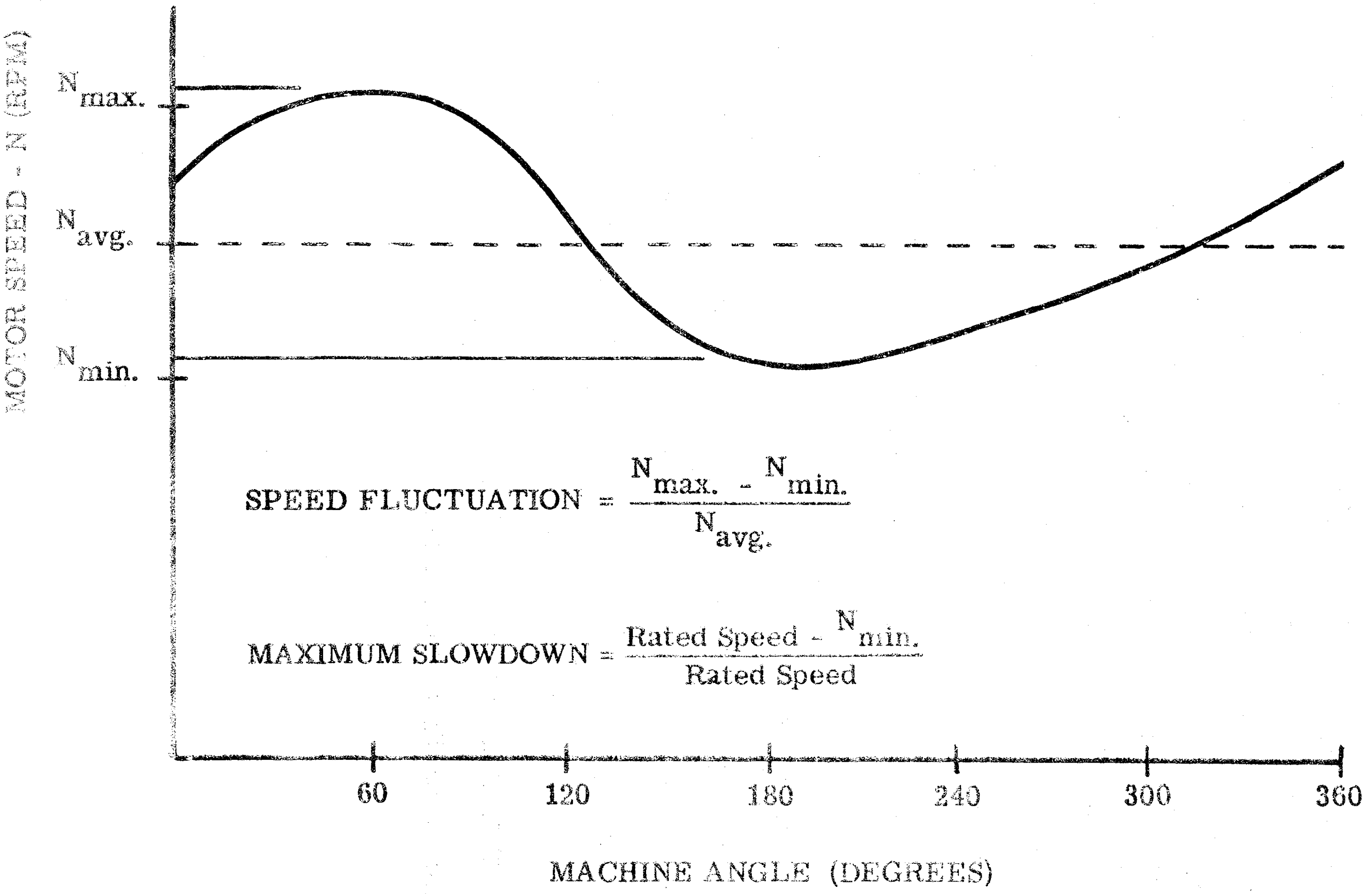
F. R. Crossley, "Dynamics in Machine," The Ronald Press Company, 1954.

C. C. Libby, 'Motor Selection and Application," McGraw Hill Book Company, 1960.

M. F. Spotts, "Flywheel Machine," Machine Design, March 28, 1963.

CONTINENTAL CAN COMPANY, INC.

Corporate Equipment Engineering



$$\text{SPEED FLUCTUATION} = \frac{N_{max.} - N_{min.}}{N_{avg.}}$$

$$\text{MAXIMUM SLOWDOWN} = \frac{\text{Rated Speed} - N_{min.}}{\text{Rated Speed}}$$

FIGURE I - MOTOR SPEED vs. MACHINE ANGLE [illegible]

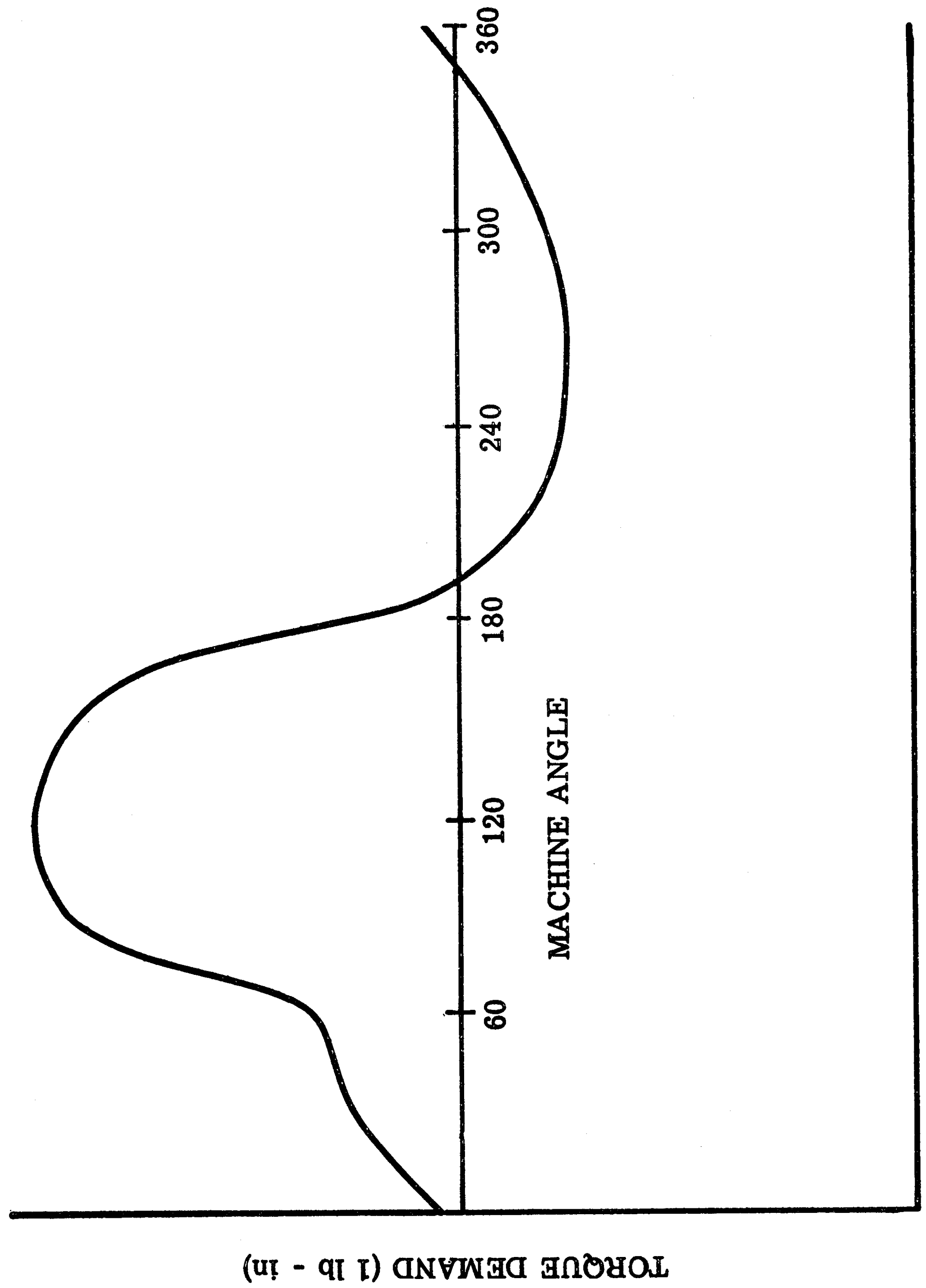


FIGURE 2 - TYPICAL DRAWING PRESS TORQUE DEMAND CURVE

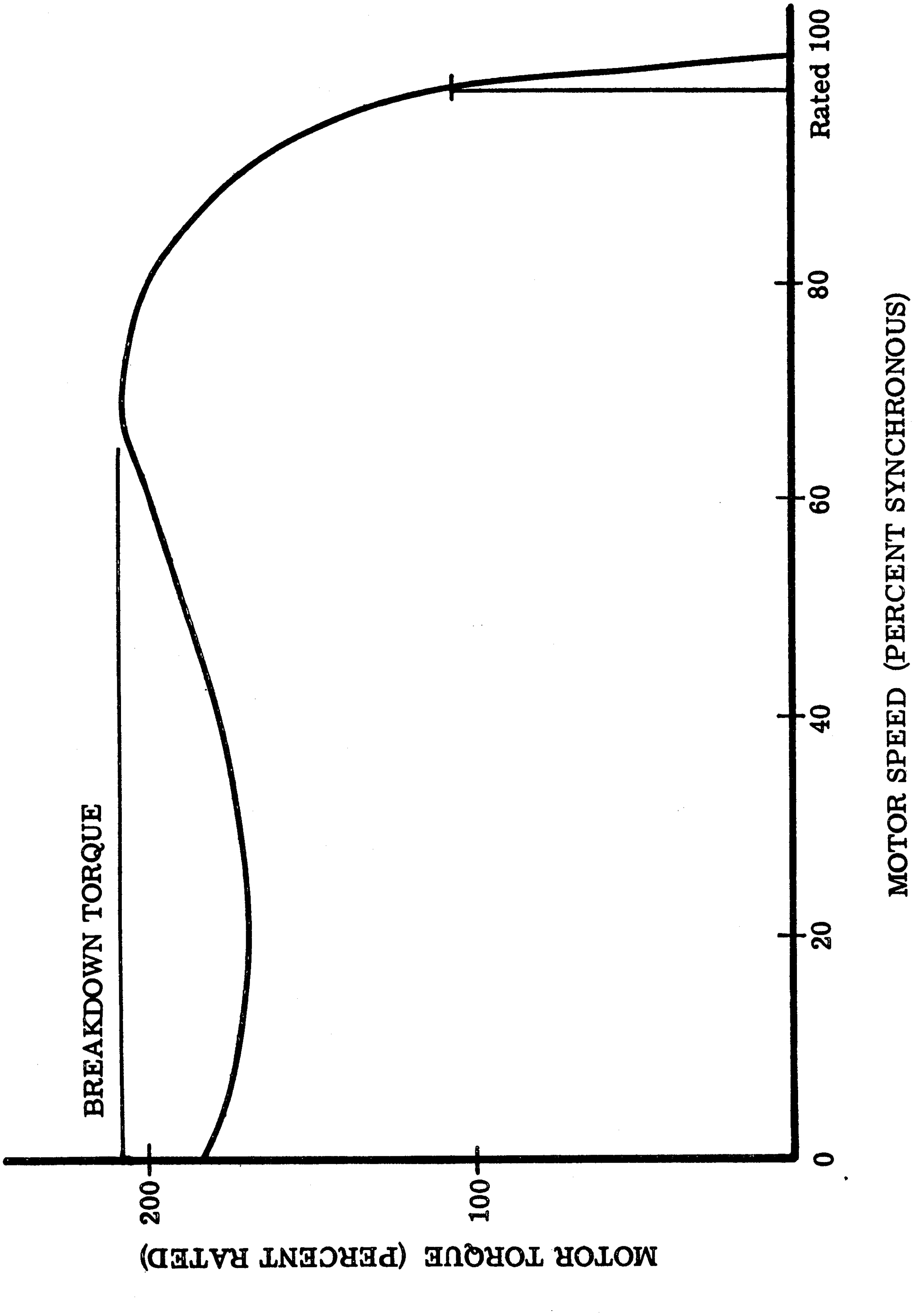


FIGURE 3 - TYPICAL NEMA-B SPEED TORQUE CURVE

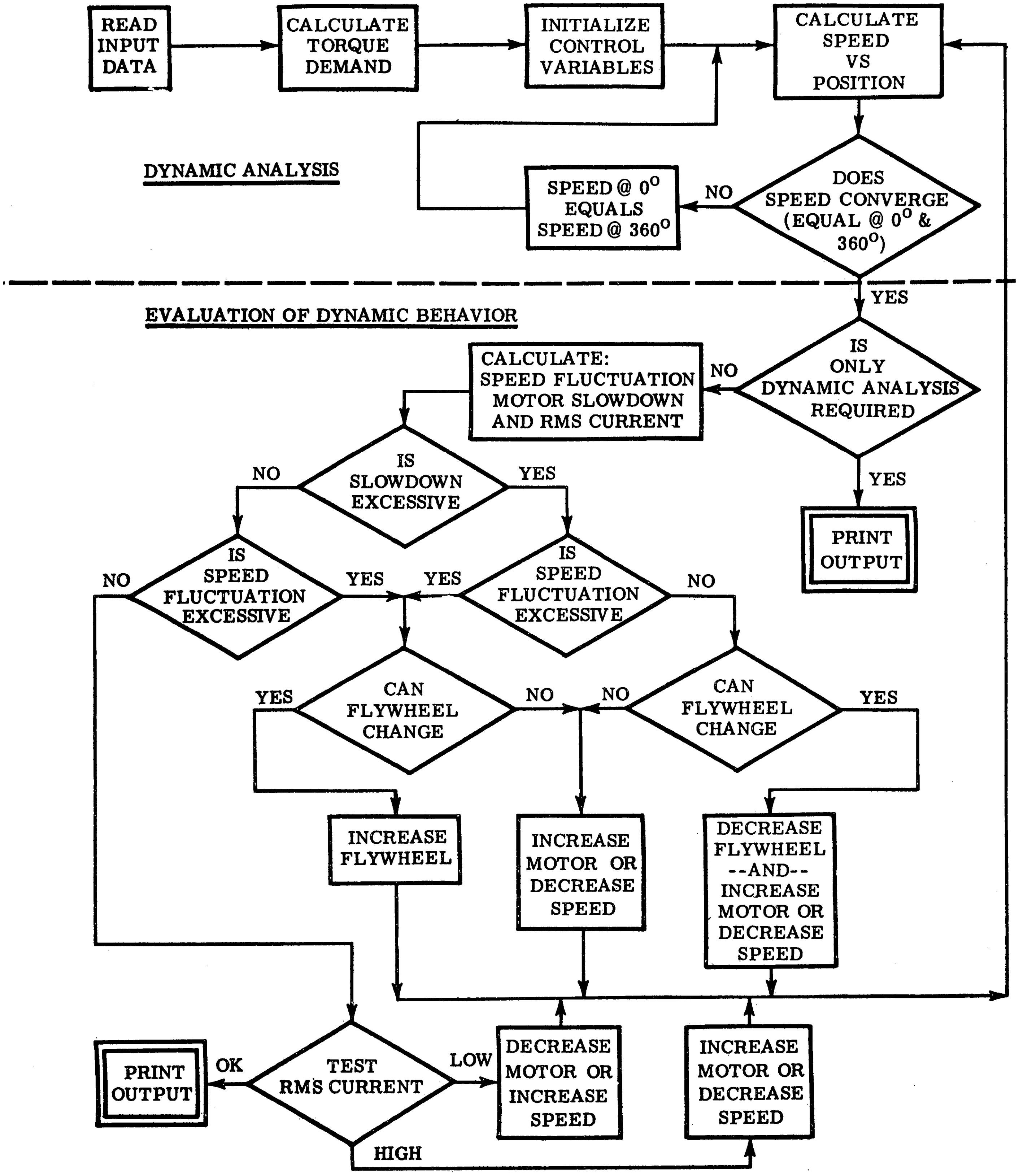

READ INPUT DATA
CALCULATE TORQUE DEMAND
INITIALIZE CONTROL VARIABLES
CALCULATE SPEED VS POSITION
DYNAMIC ANALYSIS
SPEED @ 0° EQUALS SPEED @ 360°
NO
DOES SPEED CONVERGE (EQUAL @ 0° & 360°)
YES
EVALUATION OF DYNAMIC BEHAVIOR
CALCULATE: SPEED FLUCTUATION MOTOR SLOWDOWN AND RMS CURRENT
NO
IS ONLY DYNAMIC ANALYSIS REQUIRED
YES
PRINT OUTPUT
NO
IS SLOWDOWN EXCESSIVE
YES
NO
IS SPEED FLUCTUATION EXCESSIVE
YES
YES
IS SPEED FLUCTUATION EXCESSIVE
NO
YES
CAN FLYWHEEL CHANGE
NO
NO
CAN FLYWHEEL CHANGE
YES
INCREASE FLYWHEEL
INCREASE MOTOR OR DECREASE SPEED
DECREASE FLYWHEEL --AND-- INCREASE MOTOR OR DECREASE SPEED
PRINT OUTPUT
OK
TEST RMS CURRENT
LOW
HIGH
DECREASE MOTOR OR INCREASE SPEED
INCREASE MOTOR OR DECREASE SPEED

FIGURE 4 SIMPLIFIED LOGIC DIAGRAM

# ACTIVE NETWORK ANALYSIS

Alonzo F. Adkins and
R. H. Seacat
Electrical Engineering Department
Texas Technological College
Lubbock, Texas

At the present time, there are two topological methods that are very useful in analyzing passive electrical networks. One of these methods was developed by W. S. Percival in 1953.[1] The other method was developed by Feussner in 1903.[2,3] Both methods have been improved upon by investigators in circuit theory at Texas Technological College.[3] Recently, E. D. Merkl has extended both Feussner's method and Percival's method to include active networks containing vacuum tubes and transistors.[4] In this paper, we shall use Merkl's extension of Percival's method to a program for analyzing electrical networks containing dependent current sources.

## A BRIEF REVIEW OF PERCIVAL'S METHOD APPLIED TO PASSIVE NETWORKS

The equilibrium equations for an electrical network being solved on the nodal basis can be put into the form

$$[I] = [Y][V], \qquad (1)$$

where [I] is a column matrix with n rows, each element being an independent current source, [Y] is an n by n admittance matrix, and [V] is a column matrix with n rows. The column matrix [E] contains the unknown node voltages that are to be calculated. If all current sources are removed except the one connected to the $i^{th}$ node, the voltage at the $k^{th}$ node can be expressed as

$$E_k = I_i \frac{\Delta_{ik}}{\Delta} . \qquad (2)$$

[1] W. S. Percival, "Solution of Passive Electrical Networks by Means of Mathematical Trees," Journal Institute of Electrical Engineers, London, 1954, Volume 101, Part IV, pp. 258-264.

[2] W. Feussner, "Uber Stromverzweigung in Netzformigen Leitern," Annalen der Physik, 1902, Fourth Series, Volume 9, pp. 1305-1329.

[3] R. H. Seacat, "A Method of Network Analysis Using Residual Networks, " (Dissertation, Texas A & M University).

[4] E. D. Merkl, "Topological Methods Applied to Active Networks," (Dissertation, Texas Technological College).

Percival has shown that the principal determinant, $\Delta$, can be found from the graph of the network. The expanded value of the principal determinant is given as,

$$\Delta = \Sigma \text{ tree edge admittance products over all trees.} \qquad (3)$$

Usually a network is so complicated that it is difficult to enumerate all trees of the network. However, the following relation can be used to expand a complicated network into less complicated networks,

$$\Delta = Y_x \Delta_x + \Delta_{x'}. \qquad (4)$$

This relation states that the principal determinant of a network containing an admittance $Y_x$ is equal to $Y_x$ multiplied by the determinant of the network with $Y_x$ shorted plus the principal determinant of the network with $Y_x$ open.

The numerator $\Delta_{ik}$, or the numerator for the voltage between any two nodes, may be found in the following manner: Let the input terminals of the network be 1 and 1', and the output terminals be 2 and 2'. Find two non-touching paths, one from terminals 1 to 2 and the other from terminals 1' to 2'. All edges used in the two paths are multiplied together and all nodes used in the two paths are shorted together. The trees of the remaining graph are found and the product of the tree edges and the path edges is found. This process is repeated for all possible paths and the products are formed into a sum. Finally, the whole procedure is repeated with the paths going from terminals 1 to 2' and 1' to 2. The second results are subtracted from the first to give the expanded value of

$\Delta_{ik}$.

## THE EXTENSION OF PERCIVAL'S METHOD

Percival's method may be applied to active networks using a modified set of rules. For the principal determinant, the following rules apply:

Rule 1. A dot is placed over the element through which the control current is flowing. A dot over an element is just a shorthand way of writing, "this element should be multiplied by $(1 \pm \alpha)$." Whether or not the dot signifies multiplication by $(1 + \alpha)$ or $(1 - \alpha)$ depends on the direction of the dependent source.

Rule 2. Each element in parallel with the dependent current source, not dotted, will have an x placed over it. The significance of the x is to cancel the effect of the dot if a product of an x and a dot is made. If a product contains no dotted element the effect of the x is ignored. If an undotted element is shorted across the dependent current source during graph reduction, this element is also "x"ed. If the expansion of Δ is determined without graph reduction, the dotted elements of any tree having any combination of undotted elements shorting the dependent current source are ignored.

Rule 3. The product of two or more dotted elements produce the same results as if only one of the product members were dotted.

Rule 4. If the principal determinant is expanded by graph reduction, no dotted or "x"ed element can be removed from the graph.

In order to calculate the numerator, the following rules are applied to Percival's method:

Rule 1. The value of the dependent current source is replaced by an independent current source that has a value equal to α times the independent current source. The network is then treated as a passive network with two source currents ($I_1$ and $\alpha I_1$).

Rule 2. A dot is placed over each element used to write the control current in terms of the independent current source. Again, the dot means "multiply this element by ($1 \pm \alpha$)."

Rule 3. When determining the terms for $\alpha I_1$, a dotted element that is not in parallel with the independent current source is treated as if the element was not dotted. A dotted element that is in parallel with the independent current source gives a value of zero.

Rule 4. The product of an x element with a dotted element cancels the effect of the dot as in Rule 2 for the principal determinant.

Rule 5. All terms involving multiplication by α that do not contain the element through which the control current flows are dropped.

## PROGRAM DESCRIPTION

A program which is based on Merkl's contribution to active network topology has been written. The program is centered around a straightforward tree finding algorithm. By definition, a tree of a network is a combination of branches of the network meeting two restrictions:

1. All nodes of the network are connected by the branches of a tree.

2. The branches of a tree cannot form a closed loop.

Using the definition of a tree, the trees of a network may be enumerated as in the following example. Consider a complete four node network (i.e., a four node network with branches connecting every node). Let the nodes of the network be numbered 1, 2, 3, and 4. Then a particular branch of the network can be referred to by the two node numbers that the branch is connected to. For example, branch 3, 4 refers to the branch connecting nodes 3 and 4. Since a tree must connect all nodes of a network, each tree of the network must contain at least one branch connected to node 1. Therefore, separate the trees of the network into four groups, one group containing branch 1, 2, one group containing branch 1, 3, and one group containing branch 1, 4. Now find the trees for each group by "growing" branches to the already existing branches until the tree is complete. The group of trees containing branch 1, 2, may be further divided into second groups by "growing" a branch onto branch 1, 2. This results in the following second groups for the group 1, 2.

| | |
|---|---|
| 12 | 13 |
| 12 | 14 |
| 12 | 23 |
| 12 | 24 |

Now, the trees may be completed by "growing" third branches on each second group as follows:

| | | |
|---|---|---|
| 12 | 13 | 14 |
| 12 | 13 | 24 |
| 12 | 13 | 34 |
| 12 | 14 | 23 |
| 12 | 14 | 43 |
| 12 | 23 | 24 |
| 12 | 23 | 34 |
| 12 | 24 | 43 |

In order to prevent redundancy, in enumerating the trees, the branches for each nth group start with a branch that has not been previously used in the n-1 group.

Now the procedure is repeated for each of the original groups resulting in the listing of the trees of the network as follows:

| | | |
|---|---|---|
| 12 | 13 | 14 |
| 12 | 13 | 24 |
| 12 | 13 | 34 |
| 12 | 14 | 23 |
| 12 | 14 | 43 |
| 12 | 23 | 24 |
| 12 | 23 | 34 |
| 12 | 24 | 43 |
| 13 | 14 | 32 |
| 13 | 14 | 42 |
| 13 | 32 | 34 |
| 13 | 32 | 24 |
| 13 | 34 | 42 |
| 14 | 42 | 43 |
| 14 | 42 | 23 |
| 14 | 43 | 32 |

For a network that does not have branches connecting all nodes, only one additional restriction must be added to the algorithm. The restriction is as follows: Before a branch can be "grown" in order to form a tree, the branch must exist.

Notice that listing the trees in this manner allows the expressions for the summation of the tree edge products to be easily factored. For example, the denominator for the complete four node network can be written as:

$$\begin{aligned}\Delta = {} & 12(13(14 + 24 + 34) + 14(23 + 43) + 23(24 + 34) \\ & + 24(43)) + 13(14(32 + 42) + 32(34 + 24) + 34(42)) \\ & + 14(42(43 + 23) + 43(32))\end{aligned}$$

A FORTRAN program has been written that will produce Δ in this factored form for a passive network. This program was further extended for use on active networks. The extension consisted of several routines for checking each tree according to the previously mentioned rules that apply to the principal determinant of an active network. The program inserts the (1 ± α) factor in appropriate locations in the expression for Δ.

A program that determines the numerator functions for arbitrary node voltages of a network has also been written. This program uses a "path finding" routine which determines all paths from the input terminals to the output terminals. Each time a set of paths are found, the path edge branches are inserted into the numerator function, and a "short code" is stored for each element in the path. A modified tree finding routine then determines the additional terms for the particular set of paths. This procedure is iterated for all possible paths from the input terminals to the output terminals. Then, the entire operation is repeated with the active source as the input.

# SIMULATION OF A RADIO-DISPATCHED TRUCK FLEET

By
John W. Sawyer
Wake Forest College

This paper describes a study made for the R. J. Reynolds Tobacco Company, which arose in response to a seemingly simple question: "How many trucks should we be operating?"

An analysis of what was involved in such a question brought out the following information.

The Engineering and Construction Shops operate a fleet of trucks which can be used to haul either materials or crews of workmen. When a truck is needed, the responsible person phones the dispatcher, giving the location and nature of cargo. The dispatcher notes this information, as well as the time, in his log, and dispatches a truck, by radio, as soon as one is available. When the truck completes the trip the driver radios the dispatcher and notifies him of availability. The truck routes are variable, similar to the operation of taxicabs.

If no truck is available to handle a request, in general, no penalty is attached for handling materials only; however, if a crew of workmen is waiting to be transported, their idle time at hourly wages imposes a definite financial penalty on the company.

Hence, the problem boils down to balancing the number of trucks against the cost of idle crews.

A simulation of the system was proposed to circumvent various difficulties connected with actual physical change. For example, under physical variation of the number of trucks it would take considerable time to note the effect of various factors; the present system would be in turmoil; the cost of adding and deleting trucks for experimentation would be prohibitive; and answers would not be valid at different levels of demand other than that existing at the time of physical variation.

Data was obtained from the dispatcher's log, truck trip tickets, and by visual observation of the operation. From these sources it was possible to obtain the following data:

1. The time required to service requests for transportation.
2. Frequency and distribution of trip times.
3. Magnitude of requests for truck service.
4. Frequency and distribution of requests.
5. Amount of delay in dispatching trucks (truck wait).

6. Amount of idle time (crew wait) resulting from truck wait.
7. Cost of idle time resulting from truck and crew wait.
8. Cost per man-hour of lost time.
9. Cost per hour of truck operation.
10. Present number of trucks being operated.

Analysis of considerable data led to several observations concerning truck requests:

1. There is a definite pattern, varying every half-hour for the 19 half-hour periods of the day.
2. The number of requests for trucks falls into an approximate normal distribution for each period.
3. Requests occur at random within each period.
4. The pattern of requests by time period does not vary significantly from one day of the week to another day of the week; nor does one week vary significantly from another week--that is, one day is like any other day, free of cyclic patterns.
5. The length of trips is random and independent of the time of day.
6. A truck wait resulted in a crew wait in 12.3% of the cases in which the truck wait occurred.

After these preliminary studies, a computer program was designed (see flow chart) to simulate the actual conditions encountered in providing truck service. It simulates as many days as desired, with any given number of trucks, building up normal variations in demand while maintaining randomness. The number of trucks could be varied at will to determine the optimum fleet. The simulation could be carried out at the rate of four minutes of computer time for each full day of simulation.

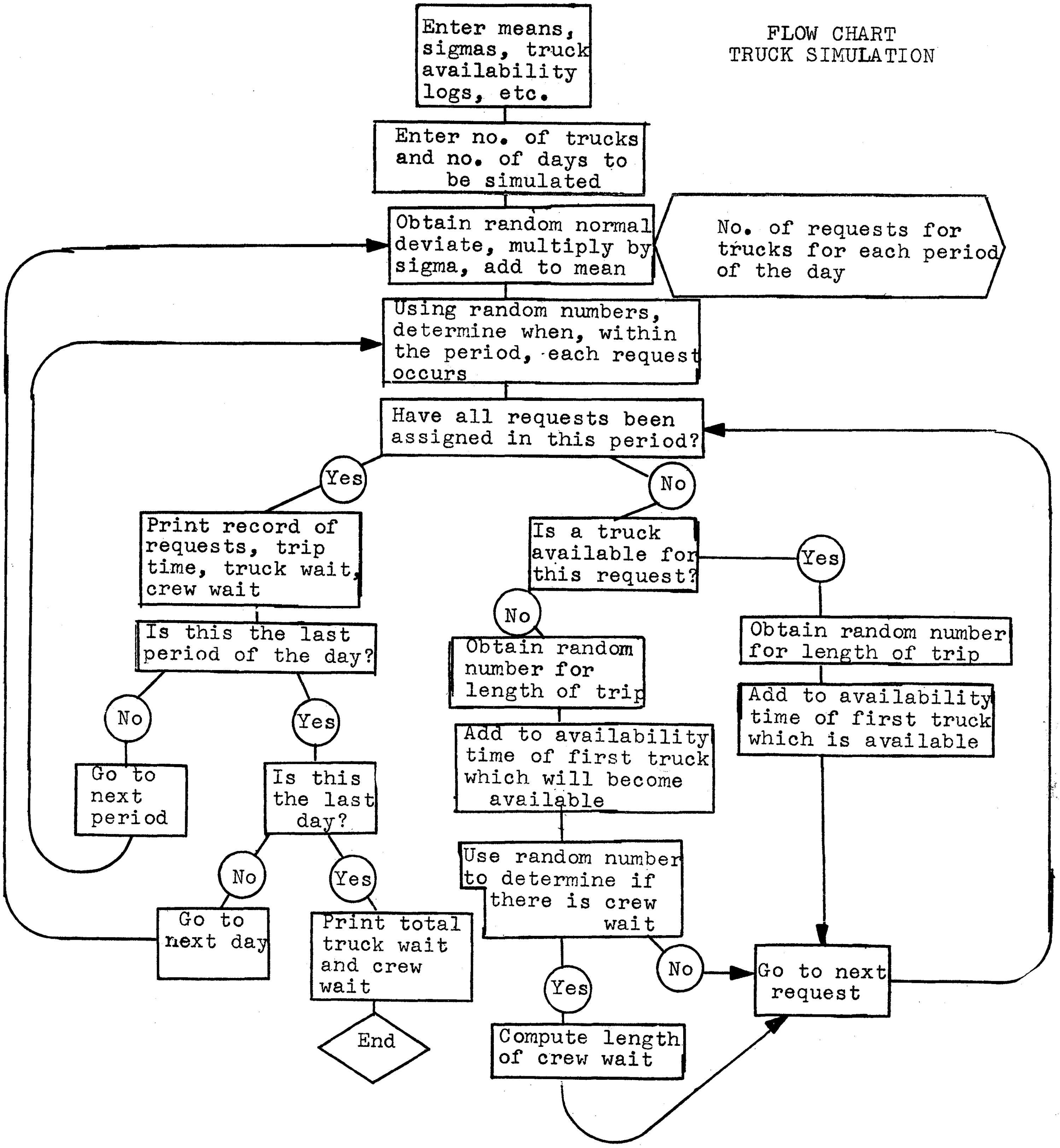
The output included the number of trucks, the number of days of simulation, a log of requests and trip times, the minutes of truck wait, and the minutes of crew wait. Crew wait time plotted against the number of trucks resulted in a nice exponential curve. This process was repeated for various levels of demand for truck service.

Crew wait cost was calculated and was balanced against the cost of adding additional trucks. Finally, a simple operating curve was given to the Engineering and Construction shops, expressing the optimum number of trucks as a function of the average number of requests for truck service per day. This, in effect, gave a break-even point at which to buy another truck.

This simulation was run first in 1959, and has been re-run twice since that time simply for verification and comparison purposes. It has been found that the predicted crew wait time at various levels of request has been very close to the actual crew wait, and, except for slight modification to reflect changing costs, the operating curve is still the authority for adding trucks.

Management has been extremely happy with the results, and several additional simulations have been requested as a result of this satisfaction.

FLOW CHART
TRUCK SIMULATION


Enter means, sigmas, truck availability logs, etc.
Enter no. of trucks and no. of days to be simulated
Obtain random normal deviate, multiply by sigma, add to mean
No. of requests for trucks for each period of the day
Using random numbers, determine when, within the period, each request occurs
Have all requests been assigned in this period?
Yes
No
Print record of requests, trip time, truck wait, crew wait
Is this the last period of the day?
No
Go to next period
Yes
Is this the last day?
No
Go to next day
Yes
Print total truck wait and crew wait
End
Is a truck available for this request?
No
Obtain random number for length of trip
Add to availability time of first truck which will become available
Use random number to determine if there is crew wait
Yes
Compute length of crew wait
No
Yes
Obtain random number for length of trip
Add to availability time of first truck which is available
Go to next request

# Simulation of Automobile Traffic

Dr. Phyllis Fox

Dr. Frederick Lehman

Newark College of Engineering

Presented to the 1620 Users Group Meeting, New York City, October 8, 1965

## Introduction

We have been using our 1620 Model II computer at Newark College of Engineering to simulate car following (no passing). In particular we are studying the rear-end accident situation with the hope of being able to suggest promising prevention devices and measures.

With a mathematical - decision making type of model we can generate accidents and near accidents at will. The cost of this method of study is much less than a field investigation where reliable data is very difficult to obtain. To have practical significance the model must well represent the real situation and yield accident rates similar to those found from motor vehicle accideñt records.

The principal computer work which has been done in the area of automobile simulation falls either in the area of urban traffic network simulation with emphasis on the intersection problem, or in the area of the highway interchange problem. In our study, on the other hand, we are emphasizing the individual car + driver situation, hoping to understand the fine structure of driver behavior by concentrating on the simple car-following, nopassing, single-lane driving situation, as might be found for example in a tunnel. Traffic models of this particular situation have been postulated and studied by several investigators, but the research has focussed on the steady-state aspect of throughput of traffic flow. We are stressing the transient, accident-causing, exceptional situation.

We are devoting considerable care to structuring our model around the human characteristics and behavior of the drivers. We have explored the literature for results of current research in human reaction times, perception activity, sensitivity, response, etc., so that we can use appropriate distributions for our parameters, and see how our model behaves under reasonable changes of these parameters.

## Mathematical Model

In Fig. 1 are shown three cars from a platoon of cars traveling in the single-lane situation. The middle car, car n, is at position $X_n$ and has velocity $V_n$. The behaviour of car n is influenced primarily by the relative velocity between itself and car n-1 ahead of it. If car n is closing in too fast on car n-1 the relative velocity gets negative and car n then tends to slow down; if the relative velocity becomes positive on the other hand, car n might be expected to accelerate. Roughly speaking then acceleration (or deceleration) response is proportional to relative velocity

$$\frac{d^2 x_m}{dt^2} = K \left( V_{m-1} - V_m \right)$$

with some delay on the part of the response. The early models were based on this simple equation, which can be made to fit relatively calm driving situations.

Obviously however, there are a great many other factors contributing to driver response. What he does, depends not only on relative velocity, but relative spacing, on his own absolute velocity, on his individual characteristics such as reaction time, perceptiveness, sensitivity of response and certainly many other factors. The model we are now using incorporates many of these features. It stems from the work of Herman and others at General Motors, and in particular follows the model of L. Edie at the Port Authority of New York. The equation for the model is shown in Figure 2. It postulates that the acceleration (or deceleration) response is directly proportional to individual velocity and relative velocity, and inversely proportional to the square of the spacing between the cars. The factor $\alpha$ , the sensitivity factor, determines the degree of the response, and T represents the response reaction delay time. T includes perception time, decision time and response time. An equation of this form has been shown to fit actual driving data very well.

Computer Implementation

We consider it likely that as our research progresses we may want to program our simulation as a list-processing program with property lists representing the characteristics for each vehicle + driver. However for the initial exploration of our model we have used Fortran, in particular the Kingston Fortran for the disc. We consider it an excellent Fortran system. We have used its plotting routine to considerable advantage, because, having no On-line printer, we plot data on our 407 from the punched output of the Kingstran plot. Though the process is a bit time and card consuming, we find it more useful than even a plotter would be for us, since we can get ten plots on the same graph.

In our computer representation certain characteristics such as car position, velocity and acceleration are represented as two dimensional vectors, the first subscript being the car number and the second the time step. Other characteristics, such as individual preferred velocity, desired headway, and reaction time are one-dimensional vectors. The latter parameters, do not change every time step, but may change asynchronously during a computer run; for instance reaction time is made to depend on the driving situation the driver has just experienced.

At a given time step each car of the platoon is considered in sequence, its acceleration determined and then integrated to give velocity and distance. The integration scheme is quite simple, of error ( $\Delta t^2$ ), and for $\Delta t$ =.1, it takes 4.4 seconds of computer time to compute the behavior of one car for one second of real time. Thus, for a 5 car platoon, for example, 22 seconds of 1620 time are required to compute one second of actual time. This gives us a sort of slow-motion look at what is happening. Of course for simulation of large urban traffic networks this penalty ratio would be disastrous, but for our purposes it is not too bad considering that accidents happen pretty fast. Each case is run in a range of 10-20 seconds of real time.

Our program fits into 40K of core storage allowing for 20-car platoons which is larger than we wish to consider so far.

Results

Our program is set up to test out the effect of various maneuvers of the lead car on the cars following. In Fig 3 is shown a typical maneuver we have used to experiment with. Initially all the cars of the platoon are going along in this case at 68 ft. spacing and 48 ft./sec. velocity (about 32 m.p.h.). Then the lead car decelerates at 8 ft./ sec./sec. for 3 seconds to a new velocity of 24 ft/sec.

When we first tried out this model we were getting an unrealistic number of rear-end collisions, and we realized that our lack of realism was due to not letting a car look two cars ahead. What we had was a "foggy" model where the drivers could see only one car ahead and might indeed have collisions at the rate we were experiencing.

We have now expanded the equation to include both the car ahead and the car ahead of it, and the way the revised simulation is behaving seems quite realistic. The rate at which accidents occur depends, as one might expect, on the general reaction time of the driver population, and on their degree of response, and of course on the presence of exceptional drivers - either speed demons or vague old ladies. Fig 4 shows the results of a run where the drivers all had a desired headway keeping them too close to the car ahead. There was a collision between cars 7 & 8. Fig. 4 which was plotted from cards on our 407 shows the velocity profile of the various cars in the platoon before the collision took place. In general for position plots we plot only relative car position, but in Fig. 5 we have translated these data back into absolute positional notation in order to show the propagation of the lead car disturbance and its culmination in a collision between cars 7 and 8.

## Future Research

We plan to expand our model considerably and incorporate many more factors, especially a more detailed portrayal of human driving behavior. We will explore the multi-dimensional parameter space to see which factors seem most important in accident causation, in the hope that we may contribute some useful knowledge to the field of accident prevention.

The important aspect of model validation is always in our minds, and we will check our model against any real-world experiments we can find. We are encouraged already to note that our model gives a stable driving situation for those values of parameters, e.g. reaction time, obtained from experimental sources, and that collisions occur as we deviate from such values.

Acknowledgements We would like to acknowledge our thanks to the Public Health Service of the U.S. Dept. of Health, Education and Welfare for their support of this work under Grant Number AC 00236-01, and to the Research Foundation of Newark College of Engineering, which supported the research in its initial stages.

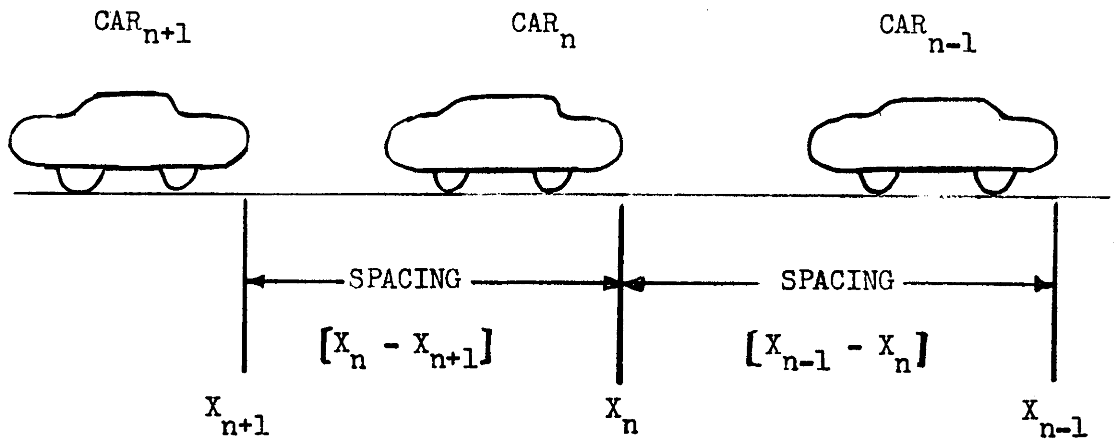


FIGURE 1. CAR FOLLOWING SITUATION

$$\text{ACCELERATION} = \left.\frac{d^2X_{n+1}}{dt^2}\right)_{t+\text{reaction time}} = \alpha\, V_{n+1} \left.\frac{[V_n - V_{n+1}]}{[X_n - X_{n+1}]^2}\right)_t$$

$V_n$ : VELOCITY OF n$^{\underline{th}}$ CAR

$X_n$ : POSITION OF n$^{\underline{th}}$ CAR

FIGURE 2. CAR FOLLOWING EQUATION

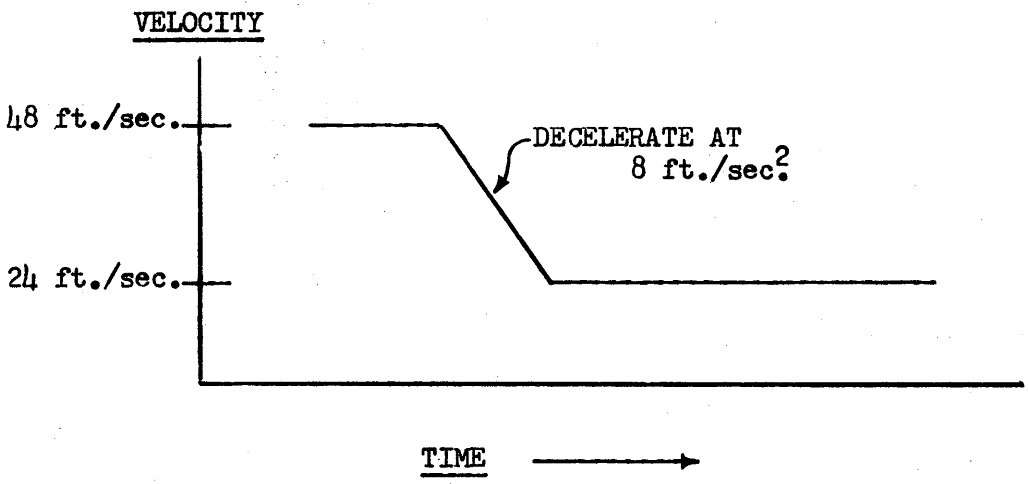


FIGURE 3. TYPICAL LEAD CAR MANEUVER

x03

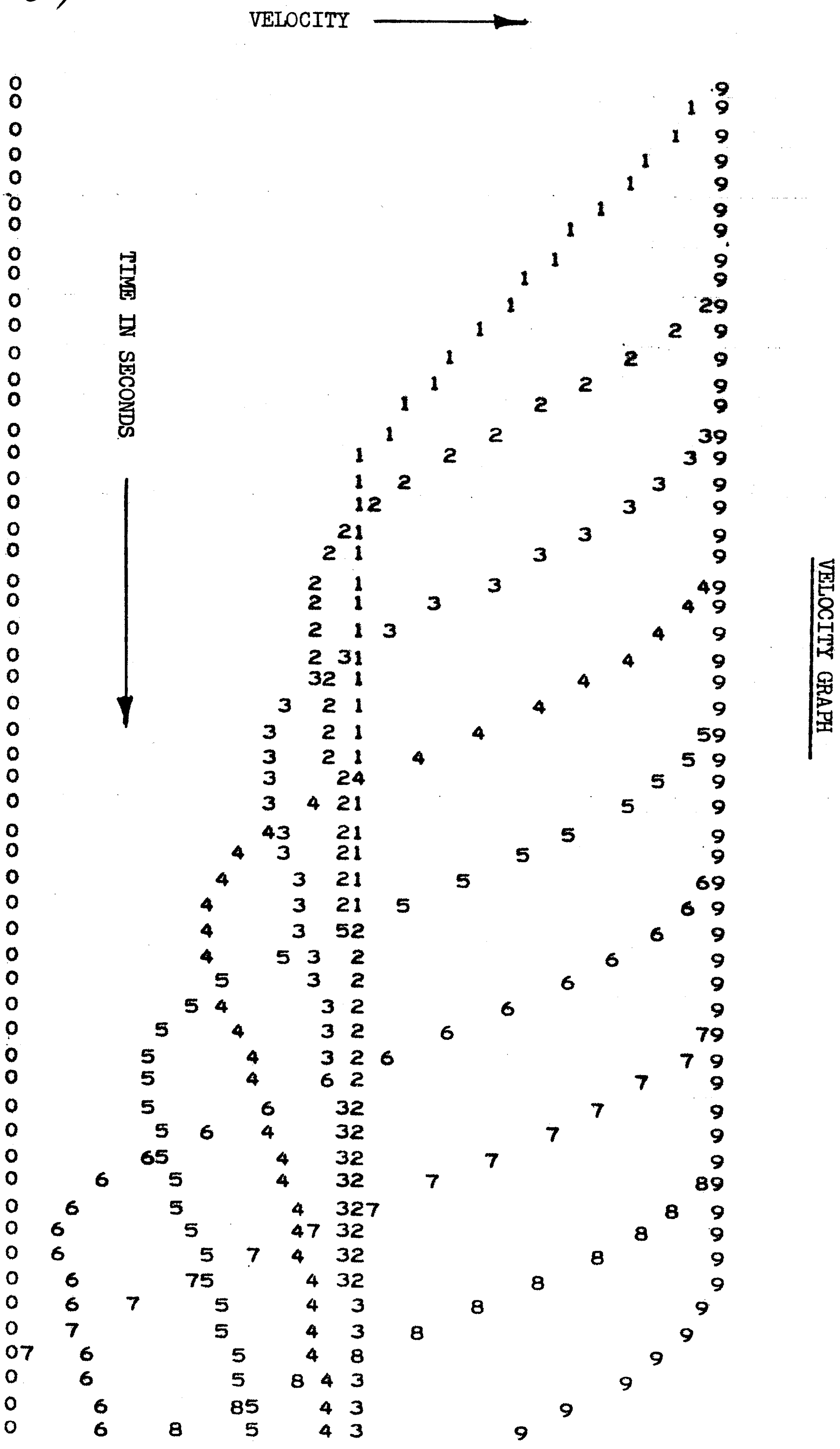


Fig. 4

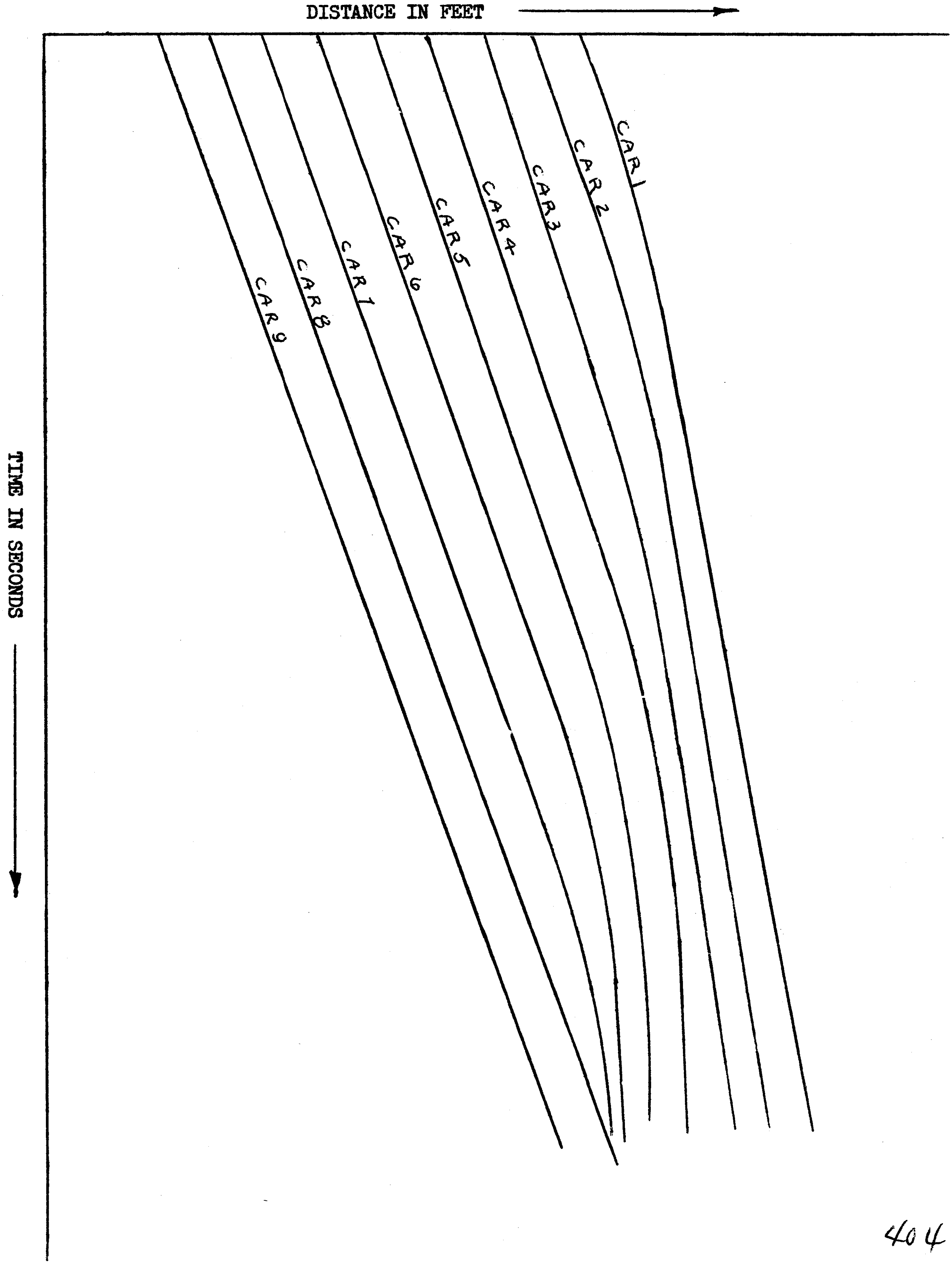
DISTANCE IN FEET
TIME IN SECONDS
CAR 1
CAR 2
CAR 3
CAR 4
CAR 5
CAR 6
CAR 7
CAR 8
CAR 9

Fig. 5

# DATA PROCESSING

# AT

# INDIANA STATE UNIVERSITY

Presented Oct. 8, 1965
New York City, N. Y.

405

# CONTENTS

# DATA PROCESSING AT INDIANA STATE UNIVERSITY

by Noel T. Smith and John T. Kline

## INTRODUCTION

Indiana State University is a multi-purpose, state supported, coeducational institution located in Terre Haute, Indiana. The University occupies a campus area of more than 50 acres in the heart of the city, and a 10-acre plot in the suburban section, where the University Lodge is located. Campus facilities include 14 academic and administrative buildings and twelve residence halls. Two additional residence halls and one new general classroom building will be in use in the spring of 1966.

Both undergraduate and graduate courses of study are offered. Degrees are granted in the fields of teacher education, liberal arts, and professional-vocational curricula. Practical arts programs are available to students who desire specialization in fields that may or may not lead to a degree.

The expanded enrollment in our University has brought greater complexity of operation and planning to the machine record and Computer Center section. Our University enrollment has increased by 90% since the machine record section was installed in 1958.

Since July, 1963 a 1620 system has been in operation. Although still card/disk oriented, the 1620 computer with a 1443 printer has added greatly to our overall system. The system is to be expanded to include two disk drives in 1966 and further expanded to an IBM 360 Model 40-disk-tape-data cell system in September, 1966. Also, scheduled for delivery in February, 1966 is a disk/printer card IBM 1130.

This presentation is centered around our present 1620 system and shows areas of use of the 1620. The areas covered include admissions, test score reporting, pre and post registration reports, mid-term high school principal's report, grade reporting, and the Evansville extension campus system.

Although Indiana State University has a complete 1620 payroll system, including machine corrections, check writing, bank reconciliation, quarterly, semi-annual and annual reporting of PERF-STRF, social security, credit union, foundation etc., the system is not discussed in this paper. The acquisitions and order accounting system for the ISU library is also by-passed. Information concerning these two systems may be obtained by contacting the ISU Computer Center.

Our sincere thanks goes to Mr. Robert Wiseman, Assistant Director of the Computer Center at ISU. Without his cooperation this paper would not have been possible.

## ADMISSIONS

After a student has been accepted for admission to Indiana State University the Admissions office sends the student's master code sheet (Exhibit 1), an information page filled out by the student upon application, to the Computer Center for keypunching. The cards punched are kept on file for use during the students stay at ISU.

## STUDENT MASTER

The student master (Exhibit 3) contains coded information pertinent to many reports during a semester. For example: student number and name, current classification, the curriculum, sex, birth date, permanent identification number, church preference, home county and state (as well as high school county and state), school enrolled, first major, second major, minor and area major, marital status, transfer status, and advanced standing status.

Using the student master, many statistical reports can be made about the composition of a student body during any semester.

## STUDENT ADDRESS

This is a name and address card used for such reports as the grade report which must be mailed to many students. The name and address card also has some housing information pertaining to such things as non-resident housing, married housing, on or off campus housing. The classification and sex is also kept on the address card. (Exhibit 4)

## PARENT ADDRESS

A parent address card is punched on entering freshman only. This card is used to mail the parents certain information during the first semester.

## ADMISSIONS REPORTS

Applications for admission to the comming fall semester usually begin during the last half of the current fall semester. In order to help the admissions office keep a better record of admissions and to formulate statistical reports concerning admissions, the Computer Center keeps the freshman masters separate from returning student masters until mid-term of the fall semester. Each week applications processed for the last week are punched and statistical reports updated and a list of applicants provided. Once a month all applicant's masters are sorted into alphabetical order and a list of applicants to date is generated. Statistical reports are generated covering such topics as: State and County frequency counts, frequency counts of the proposed majors and minors, and a count of total admissions to date.

## TEST SCORE SERVICES

Along with each application for admission there is a test score sheet sent to the Computer Center. These sheets are used to punch a test score card for each entering freshman. The test score card consists of SAT math and verbal scores, ACE scores, or ACT scores. High school rank in class and a converted percentile based on the size of the high school are kept in the card.

The test score cards are used for predicted index. The predicted index is based mainly on the SAT scores and converted high school rank in class. A list of all test scores is then prepared for distribution to the counselors and a list of predicted indexes and test scores sent to the Deans of the various schools on campus. (Exhibit 25) The predicted index is then used to prepare a statistical report on predicted indexes. The average predicted index and percentiles which a predicted index represent are made available to the counselors. After the first semester a comparison is made between the students predicted index and his actual first semester index. Presently the index of correlation is .63. (Exhibit 13)

## MASTER SCHEDULE

Approximately two months before the beginning of a semester the master course schedule is prepared by the department chairmen and sent to the Computer Center. The master course card is punched and listed. (Exhibit 27) A feasibility study is run with the course master cards as data. The feasibility study is a computer check to make sure that two classes are not scheduled at the same time for the same room or the same instructor for two classes at the same hour. The list is then sent back to the department chairmen for approval. Small changes are often made, such as adding courses or switching times, etc. (Exhibit 15)

## GRADE CARD/CLASS TICKET

After the master course card has been punched and the schedule approved, the master course card is then used to generate grade cards (Exhibit 7) and class tickets. Punched in the master course card is the number of students that will be allowed to take this course, course instructor, room, time, building, course number, department, and a master course number. The grade card is generated by the computer which puts department, course number, master course number, building, room, time, course description, and credit hours in the grade card. The number of grade cards generated for one class is determined by the maximum number punched in the course master. Class tickets are generated in much the same way except instructors name is used instead of course description. (Exhibit 29) (Exhibit 15)

## MASTER CARDS/PERMITS/ID/ENCUMBRANCE

In preparation for registration an ID card and a permit to register is made for each student. The ID card and the permit to register are merged behind the master card. An encumbrance list is circulated among the various departments on campus and returned to the Computer Center. An encumbrance card is punched (Exhibit 2) and the permit to register is replaced by the encumbrance card. At registration time the student must pick up his master, permit, and ID before he registers for any classes. If he has an encumbrance card it must be cleared before he is issued his permit to register. A student is not allowed to enter the registration area without a permit to register. (Exhibit 16)

## PACKET PREPARATION

Presently at ISU, registration for classes is accomplished by filling in several cards with the information requested. This group of cards is known as the packet. (Exhibit 31) The packet consists of the following cards: Registrar's card form 2, Registrar's card form 3, Business office fee, Academic Dean's Scholarship, IBM Information, Student Personnel, Housing Information, School of Business (for business majors only), School of Education (for education majors only), Automobile registration, Church preference, Bluebook information (Student directory), and Extended Services (evening or Saturday students only). These packets are made available to the students before registration. They may fill them out before registration time. (Exhibit 14)

## SECTIONING

Sectioning is the process of merging the cards with their appropriate class ticket and separating the groups into the various departments which they represent. These groups of cards are then taken to the registration area. At the registration area a student proceeds to the department table to obtain a grade card and class ticket according to his proposed class schedule. After the student has received all of his class cards he goes through the fee line where all of the cards are checked for missing information or missing cards, and pays his fees. After the fee line all the cards are collected except class tickets and ID card.

## POST REGISTRATION CARD HANDLING

After registration the cards are collected and sent to the Computer Center to be distributed. Using the 514 the student number is interspersed gang-punched in all the student's cards. The cards are then sorted into alpha-order, separated, and sent to the appropriate offices. (Exhibit 17) The master, IBM information, and grade cards are kept by the Computer Center.

## HOURLY EQUATED REPORT

The Hourly Equated Report (Exhibit 24) is a report concerning classification, sex, and the number of hours a student is taking this semester. The report has subtotals by general classification (part-time and full-time) and is separated by sex into the following catagories: freshman, sophomores, juniors, seniors, graduates, advanced graduates, special students and auditors. The grand total line provides the total number of freshman, sophomores, etc. attending ISU this semester. ( Exhibit 18)

## INSTRUCTOR LOAD REPORT

This report is a count of students in each class. Using this report we can show how many students in any department are being taught by the same professor. The report also provides room utilization information. It consists of counts separated by instructor, department, subject, room, time, and building. At the end of the semester it is this report, along with other information, that is used to schedule final exams. (Exhibit 12) ( Exhibit 18)

## CLASS LISTS

After late registrations have been completed (approximately 10 days after registration) the Computer Center generates a preliminary class list (Exhibit 11) to be sent to the instructor. The class list is a three copy report. The instructor uses the first copy as an attendance record, the second copy is sent back to the Computer Center at mid-term grade reporting, the third copy is kept for reference by the registrar's office. After mid-term and the deadline for all drop and adds the Computer Center generates a similar single copy class list for the instructor to use for final grade reporting. The class list report consists of a heading containing such information as instructor, building, room, time, etc. The main body of the report is a list of the students attending that particular class and their classification. The classification on this list is meant to be a help to the instructor in his evaluation of the student during the course, and a guide to mid-term reporting. (Exhibit 41)

## OFFICIAL ENROLLMENT REPORT

The Official Enrollment Report (Exhibit 30) is similar to the hourly equated report in that it pertains to the number of students in various catagories. It is with the Official Enrollment Report that the official full time student equivalent enrollment is figured. This is based on 15 hours as a full time load. A student taking 10 hours would only be 2/3 of a student in this report. The report is separated by sex, classification, and area major. If necessary this report can be run as a state analysis or county analysis in respect to full time students. (Exhibit 41)

## RESIDENCE REPORT

The Residence Report (Exhibit 39 ) is a report showing how many students we have from each county within Indiana, each state, and country. Each count is separated by sex and classification. A subtotal by residency in Indiana and non-residency is provided by this report. (Exhibit 41 )

## MAJOR AND MINOR REPORTS

The Major and Minor Reports are mainly for the benefit of the department chairmen. They consist of a count separated by classification for each area major and minor and each specific major and minor offered at ISU.

## MID-TERM GRADE REPORTING

At mid-term a grade report is sent out for each entering freshman, and deficiencies sent out for the upper classmen. One week prior to mid-term the Computer Center generates a color coded card for each freshman and upperclassman in each class. (Exhibit 33) These cards are sent to the appropriate instructors. The instructor records the mid-term grade on this card and returns it to the Computer Center. The grades are then punched in all the freshmen cards and. failing grades are punched in the upperclassmen cards. The mid-term grades are sent out on the same form as final grades with a comment "mid-term" printed on the form. The final grade report program is used and a mid-term index card punched for freshmen. (Exhibit 42)

## HIGH SCHOOL PRINCIPAL'S REPORT

After mid-term at ISU there is a meeting of high school principals from the various Indiana high schools represented on campus. In order that the principals may be informed of their former student''s acheivement, the Computer Center generates two reports for them. The first report is a list of classes attended by each student from a particular high school. Punched output form this program is a summary card to be used in the next program. The second report is a statistical report showing the high school principal how his school is doing in comparison with other Indiana schools. (Exhibit 34) This report is prepared by forming an ogive curve of indices of all the entering freshmen from Indiana at mid-term and plotting a point along this curve to represent the particular high school in question. With this report the high school principal can tell the percentile rank of his school in comparison with other Indiana high schools. All plotting is done on the 1443. (Exhibit 20)

## FINAL GRADE REPORTING

Since final grade reporting is one of the most important functions of the Computer Center with regard to student record processing, the process will be explained in more detail than other reports in this paper. The following description closely follows the flow chart (Exhibit 35-36) of the grade reporting system and explains it in detail.

Explanation of the flow-chart:

A All card files are merged together in alphabetical sequence by student number.

B Computer prints grade reports and updates student index file. Probation cards for failing students. Store used cards.

C Grade reports to students and school officials.

D Output: New student index cards and probation report writing cards.

E New index cards to back in file for report use.

F Probation report cards are sorted by school.

G Probation reports are printed and sent to the Deans of the schools.

H Probation cards are stored.

Explanation of card files.

After completion of step A (shown by the flow chart), the input to the computer consists of multiple card groups; one per student.

There are five main card types involved in student grade processing, they are:

Student Grade Card-These are the cards the students picked up and turned in at registration time. At grade reporting time the returned class lists are used as data to punch the grades into these cards. (Exhibit 7)

Student Index Card-The student index file is a continuous file maintained and updated each semester by the Computer Center. This file has the complete scholastic history and current status of each student. Such coded information as housing, social organizations, transfer hours, major areas of study, sex, and classification is kept on this card.

Student Required Index Exception Card-The purpose of this card file is to automate the detailed processing of students having scholastic problems. Because the student's academic progress is of utmost concern to the University, careful monitoring and guidance techniques are essential. The Required Index Card allows the proper school authorities to carefully supervise the progress of a student. By submitting a probation form to the Computer Center, a school official may stipulate exactly what scholastic level of achievement must be met. This is done by stating what grade point average the student must earn, either on a cumulative or semester basis. This information is then entered into the student's card group and allows the computer to analyze the student's work accordingly.

If the student fails to meet this requirement, the computer will generate a card from which a complete scholastic report can be written and sent to the appropriate official. The card is labeled "exception" because in the absence of this card the computer will use the standard catalog required index schedule to analyze the student.

Student Name and Address Card-The name and address card allows for automatic addressing of the student's grade report.

Comment card-The comment card allows a school official a maximum of two lines of comment on the student grade report. Through the use of a comment code, the comment may be printed if the student fails to meet a specified grade condition. It is also possible to print the comment under any conditions.

Output:

Two main cards are generated by the computer during grade reporting. They are: the updated Index Card and an Action Card. The Index Card is held and used for next semester's grades, the Action Card is used to generate reports to be distributed to counselor's, registrar's office and department chairmen.

After grade reporting the index cards are used to generate the Scholastic Acheivement Report. (Exhibit 37)

The Scholastic Acheivement Report gives information such as number of people in a given group, total hours, total points, and average grade point ratio for this group. The groups are by sex, area major, sorority pledges, sorority actives, complete sororities, fraternity pledges, fraternity actives, complete fraternities, residence halls and residence halls by floors.

## Regional Campus

The Computer Center's activities in admissions and registration at the regional campus is a pilot study. It is hoped it will lead to a more completely automated system than is presently used on the main campus.

The following is a detailed explanation of the regional campus flow-charts.

| Flow-Chart# Block# | Exhibit Number | Explanation |
|---|---|---|
| 1 A-1 | 1 | The Application for Admission form sent from the Admissions office to the keypunch section with status (accepted-rejected) and number (perm-alpha) marked on it. This becomes the start of a student disk record. |
| 1 A-2 | | The Applications are punched as received. |
| 1 B-5 | | The Application for Admission is returned to the Admissions office. |
| 1 A-3 | 2 | A "Permit Denied" card is punched on all applications denied with a code of reason denied. |
| 1 A-4 | | A "name and address" card is punched on admissions denied for notification of student, H. S. and others. |
| 1 A-5 | | The "denied" name and address is filed. (File #8) |
| 1 B-1<br>1 C-1 | 3 | A "Student Master" card is punched on accepted admissions, then filed. |
| 1 B-2 | 4 | A "name and address" card is punched on accepted admissions for future use and notification, then filed. |
| 1 B-3 | 5 | A "permit to register" is punched for each accepted admission. |
| 1 B-4 | 6 | An "I. D." card is prepared. |

| Flow-Chart# Block# | Exhibit Number | Explanation |
|---|---|---|
| 1 C-4<br>1 C-5<br>1 D-5 | 8 | Permits, ID's, and denied are all put together in alpha order to become file for registration. As students arrive to begin registration, they pick up permit, ID, and registration form (Exhibit 8) and student goes to advisor. |

In this registration system responsibility for sectioning is left with the departments. Each course and each section of a course in the catalog is given a code number. The descriptive information of the course is fed into the computer at point 2 D-2. This will be explained later. Each department, prior to registration is given a tally sheet (Exhibit 19) to be used at registration time. These sheets are marked as each student enrolls and closing or opening sections is the department's option, and nothing more than signing the registration form is necessary. The department also assigns the code number to the registration form. This eliminates some error conditions. After this form is completed by the student, fees are paid, housing checked and the registration form is collected.

| Flow-Chart# Block# | Exhibit Number | Explanation |
|---|---|---|
| 1 D-4<br>1 C-3<br>1D-3 | | Registration is completed by completion of the registration form. The courses are listed in the proposed class schedule area and departments or others in charge of sectioning initial and put course codes on the sheet. After the schedule is completed, the registration form is returned to the keypunch section. |
| 1 D-3<br>1 D-1<br>1 E-1<br>1 D-2<br>1 E-2 | 10 | A course request card is punched with student number and the course codes from the registration form, then filed (File #3) The registration/is then filed-(File #9) form |
| 2 A-2<br>2 A-3<br>2 A-4<br>2 B-2<br>2 B-1<br>2 B-3 | | Course cards, address cards, and master cards are sorted (alpha) and merged from file #1, 2, 3. Unmatched are hand checked to establish error and refiled (File #4) |

| Flow-Chart#<br>Block# | Exhibit<br>Number | Explanation |
|---|---|---|
| 2 D-2<br>2 C-1<br>2 C-2<br>2 D-1<br>2 E-1<br>2 F-1 | | A computer program with master catalog on disk is run with input of master card and address card followed by course card. The course masters and program are filed (File #5). |
| 2 D-3<br>2 E-2 | 7<br>9 | The computer punches a completed grade card for each course a student requested, and prints "student lists" on multi-part paper. Lists are sent to various offices and one to the student. |
| 2 C-4<br>2 D-3<br>2 C-3<br>2 C-5<br>2 D-5<br>2 D-4<br>2 E-4 | | A collating operation then puts master card and grade card together and pulls address and course request cards. Masters and grade cards are filed (File #7) and address/course requests are filed (File #6). |
| 3 A-3<br>3 B-3<br>3 B-4<br>3 B-2<br>3 B-1 | 24 | Since all formats are the same as main campus, the program compatability is 100%. Therefore, all programs mentioned earlier in this paper may be used for the Extension Campus. From File #7, Student Masters and grade cards are fed into the 1620 and an Hourly Equated Report generated. The summary cards generated are a duplicate of the Master card with hours carried by a student punched. These cards are filed (File #11) for later auditing of fees. The Hourly Equated Report is sent to the Registrar, President and Business Office. |
| 3 C-2<br>3 C-3<br>3 D-3<br>3 E-2<br>3 E-3<br>3 D-4 | 11 | The Course Master Cards from File #5 and Student Master/Grade Cards are sorted and pulled to make a deck for the 10- Day Report. This report is prepared by summarizing the total students by classification in each section. The 1620 program at this point punches the 10-Day Report Cards and as a by product, generates temporary class lists to be sent to 418 |

| Flow-Chart# Block# | Exhibit Number | Explanation |
|---|---|---|
| | | the Instructors and Registrar. One list is kept by the instructor and another returned at mid-term to the Computer Center with mid-term grades, additions and corrections posted to it. Drop and add procedures are not shown in the flow-chart but are processed by machine. "Adds" go through the 1620 program shown in flow-chart #2 block #D-2. Drops are processed with the 088 collator by pulling equals on student number and course code number. After mid-term a final Hourly Equated Report and 10-Day Report is run. The difference in the two reports becomes data on "Drop/Add" studies. |
| 3 E-4<br>3 E-5 | | Grade cards and course masters used as input to program 3 E-3 are filed (in class order) for future processing. (File #12) |
| 3 F-3<br>3 F-4<br>3 F-5<br>3 G-4<br>3 G-3 | 12 | The output cards are input for the 10-Day Report program. The report generated goes to the President's office and to Deans. The 10-Day Report cards are then filed (File #13). |

## SUMMARY

Although this system has been run through only one registration (Fall'65), it did work and the processing time from 1 D-4 thru 3 G-3 for 402 students was less than two hours. Many operations such as interpreting and forms handling have not been shown here. Total cost for supplies used was less than $5.00.

## Technical Services

During the course of a year the Computer Center at ISU performs services to the University too numerous to mention in detail in this paper. Probably the most popular is a gummed label service of name and addresses of students, colleges and universities, elementary schools, high schools, junior high schools, superintendents of school systems, and many others. The card files for these listings are kept current and are coded for sorting purposes. Labels are also made for several offices of transfer and new students for use in creating new file folders. Labels are also used by the President's office for selected mailings to faculty, staff and others.

Computer programs are written and on file to perform services such as: sorority rush, contract notifications to faculty and staff, room utilization, final exam scheduling, placement office reports, computer dance, degree audit for graduation, fee audit, meal ticket, admissions "no-show" studies, cost studies, test score analysis for instructors, salary distribution reports, scholarship reports, and many standard institutional research reports.

A library of research programs is maintained at the Computer Center for faculty and graduate research. During the past academic year, 78 research projects were completed, 297 students received "hands-on" training and a complete payroll system was developed. The metered time used on the 1620 main frame was 1887.67 hours. One caution that is observed at the ISU Computer Center is that the major objectives of teaching and research are not submerged in purely service activities.

All programs mentioned in this paper are documented with detailed card layouts, procedures, and program listings. Each one is available by writing the ISU Computer Center, Terre Haute, Indiana.

# EXHIBITS

EXHIBIT 1

# To Be Returned with Your Application for Admission

A- 03170
P- 3650317
Denied code-3

STUDENT MASTER CODE SHEET

EXHIBIT 1

# To Be Returned with Your Application for Admission

A- 03340
P- 3650334
Accepted

STUDENT MASTER CODE SHEET

The following code sheet must be completed in every detail and returned to the Registrar's Cffice, Indiana State University. Failure to complete the form or any part of it will jeopardize the applicant's admission to the University.

PRINT OR TYPE ALL INFORMATION

**Date of Enrollment**

1st Semester: FALL 1965

2nd Semester: 19

1st Summer: 19

2nd Summer: 19

NAME JANE P BIRD

HOME ADDRESS 222 MAPLE (STREET) TELL (CITY) VIM (COUNTY) IND (STATE)

PARENT NAME JOE M BIRD ADDRESS SAME

CLASSIFICATION [✓ FR.] SO. JR. SR. Post GR.

SEX MALE FEMALE X

DEGREE A.B. B.S. X Other

CHECK (√) ONE ONLY ELEMENTARY TCHG. X SECONDARY TCHG: NON TCHG:

IF SECONDARY TCHG. (MAJOR) COMPREHENSIVE AREAS: 1. 2.

(MINOR) RESTRICTED AREAS: 1. 2.

(MAJOR) SPECIAL SUBJECT AREAS:

NON TCHG. CURRICULA LIBERAL ARTS: MAJOR MINOR

NON TCHG. CURRICULA OTHER: MAJOR AREA:

G.I. BILL TERM AND YEAR ENTERING I.S.C. MARITAL STATUS SINGLE

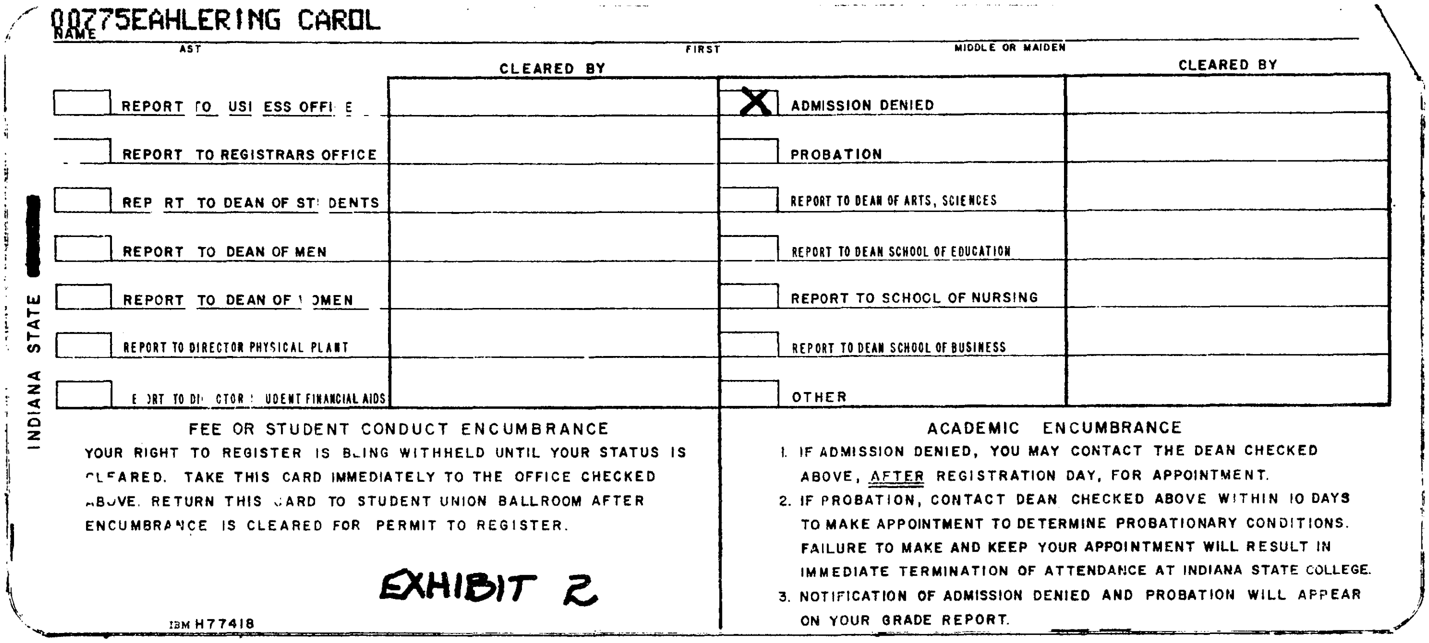
CHURCH PREFERENCE NONE

HIGH SCHOOL TELL (Name) VIM (County) IND (State)

DATE OF BIRTH: 11-15-44 NEW STUDENT: YES X NO RACE:

COLLEGES ATTENDED 1. 2. 3.

Indicate clearly the course of study you select. If foreign language is one of your areas, list specific language or languages. If Business is selected, indicate the specific Business field. Also list Science, Special Education, and other areas where a selection is designated.



NAME 00775EAHLERING CAROL
LAST | FIRST | MIDDLE OR MAIDEN

INDIANA STATE

| | | CLEARED BY | | | CLEARED BY |
|---|---|---|---|---|---|
| □ | REPORT TO USI ESS OFFI E | | ☒ | ADMISSION DENIED | |
| □ | REPORT TO REGISTRARS OFFICE | | □ | PROBATION | |
| □ | REP RT TO DEAN OF ST DENTS | | □ | REPORT TO DEAN OF ARTS, SCIENCES | |
| □ | REPORT TO DEAN OF MEN | | □ | REPORT TO DEAN SCHOOL OF EDUCATION | |
| □ | REPORT TO DEAN OF OMEN | | □ | REPORT TO SCHOOL OF NURSING | |
| □ | REPORT TO DIRECTOR PHYSICAL PLANT | | □ | REPORT TO DEAN SCHOOL OF BUSINESS | |
| □ | E ORT TO DI CTOR UDENT FINANCIAL AIDS | | □ | OTHER | |

FEE OR STUDENT CONDUCT ENCUMBRANCE

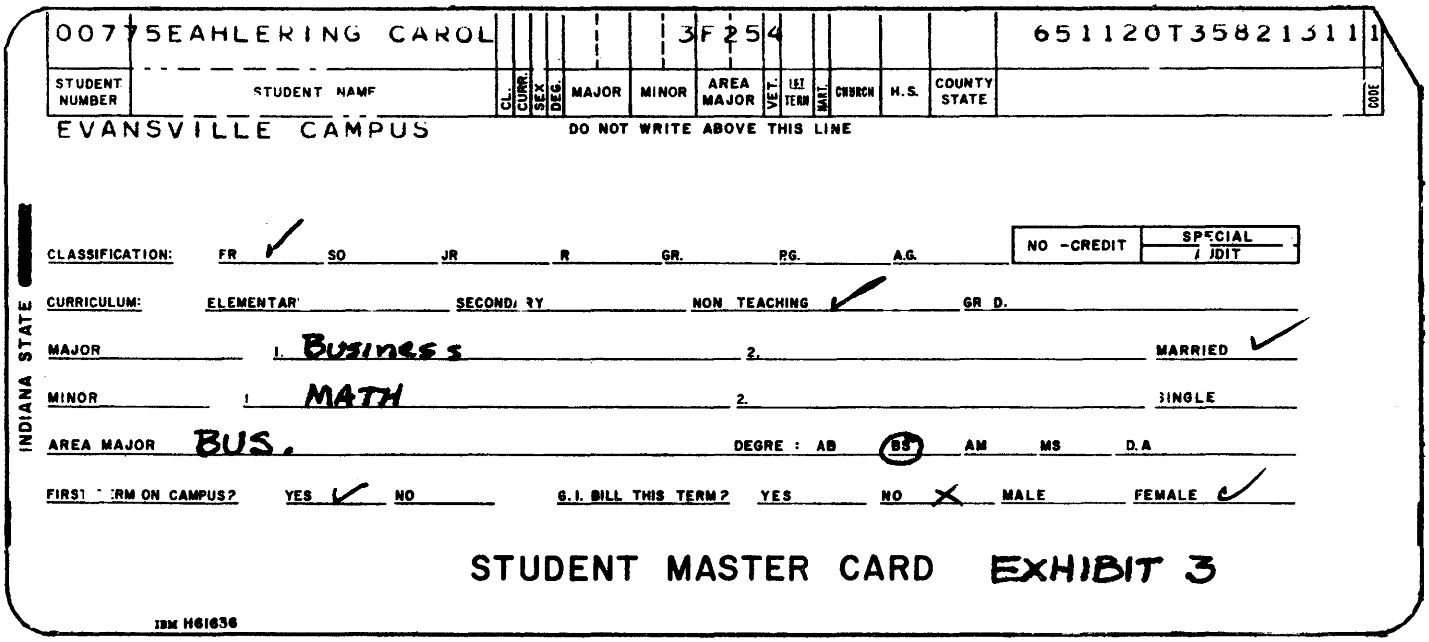
YOUR RIGHT TO REGISTER IS BEING WITHHELD UNTIL YOUR STATUS IS CLEARED. TAKE THIS CARD IMMEDIATELY TO THE OFFICE CHECKED ABOVE. RETURN THIS CARD TO STUDENT UNION BALLROOM AFTER ENCUMBRANCE IS CLEARED FOR PERMIT TO REGISTER.

ACADEMIC ENCUMBRANCE

1. IF ADMISSION DENIED, YOU MAY CONTACT THE DEAN CHECKED ABOVE, AFTER REGISTRATION DAY, FOR APPOINTMENT.
2. IF PROBATION, CONTACT DEAN CHECKED ABOVE WITHIN 10 DAYS TO MAKE APPOINTMENT TO DETERMINE PROBATIONARY CONDITIONS. FAILURE TO MAKE AND KEEP YOUR APPOINTMENT WILL RESULT IN IMMEDIATE TERMINATION OF ATTENDANCE AT INDIANA STATE COLLEGE.
3. NOTIFICATION OF ADMISSION DENIED AND PROBATION WILL APPEAR ON YOUR GRADE REPORT.

EXHIBIT 2

IBM H77418



| STUDENT NUMBER | STUDENT NAME | CL. | CURR. | SEX | DEG. | MAJOR | MINOR | AREA MAJOR | VET. | 1ST TERM | MART. | CHURCH | H.S. | COUNTY STATE | | CODE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00775 | EAHLERING CAROL | | | | | | | 3F254 | | | | | | | 651120T3582131 1 | 1 |

EVANSVILLE CAMPUS

DO NOT WRITE ABOVE THIS LINE

INDIANA STATE

CLASSIFICATION: FR ✓ SO JR R GR. P.G. A.G.

NO -CREDIT | SPECIAL AUDIT

CURRICULUM: ELEMENTARY SECONDARY NON TEACHING ✓ GR D.

MAJOR 1. Business 2. MARRIED ✓

MINOR 1 MATH 2. SINGLE

AREA MAJOR BUS. DEGREE: AB [✓ BS] AM MS D.A

FIRST TERM ON CAMPUS? YES ✓ NO G.I. BILL THIS TERM? YES NO ✗ MALE FEMALE ✓

STUDENT MASTER CARD EXHIBIT 3

IBM H61636

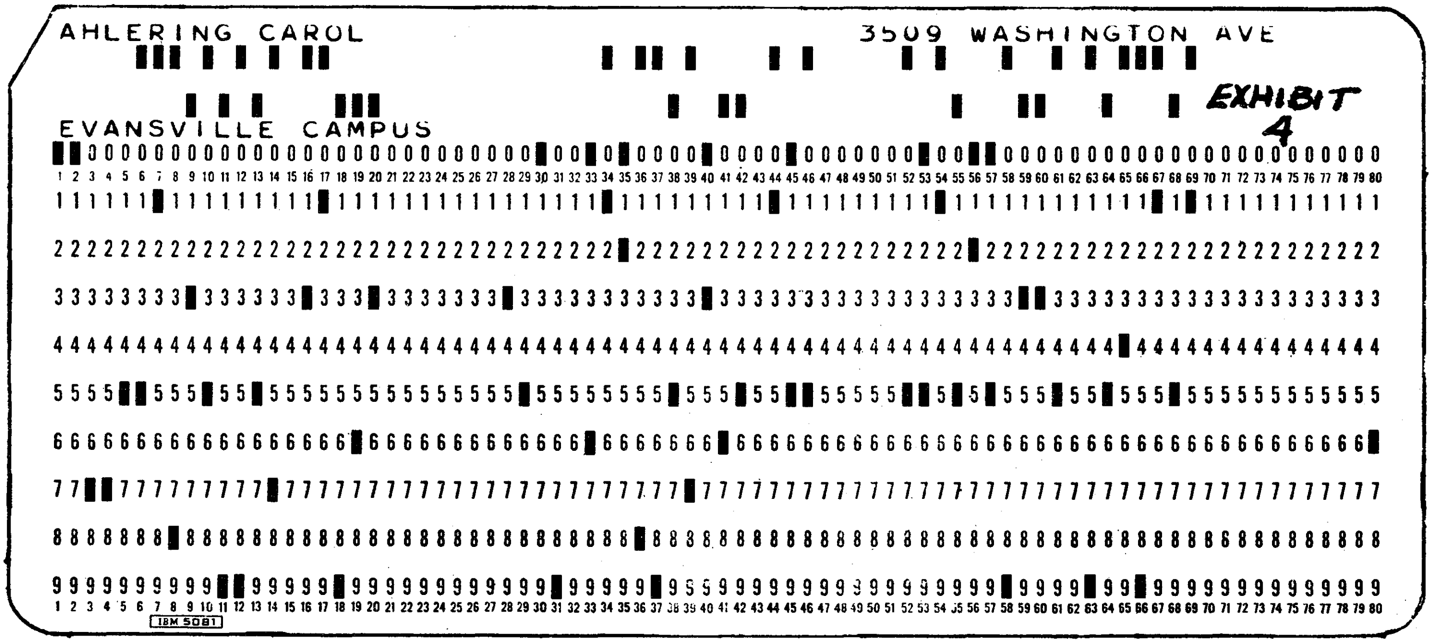
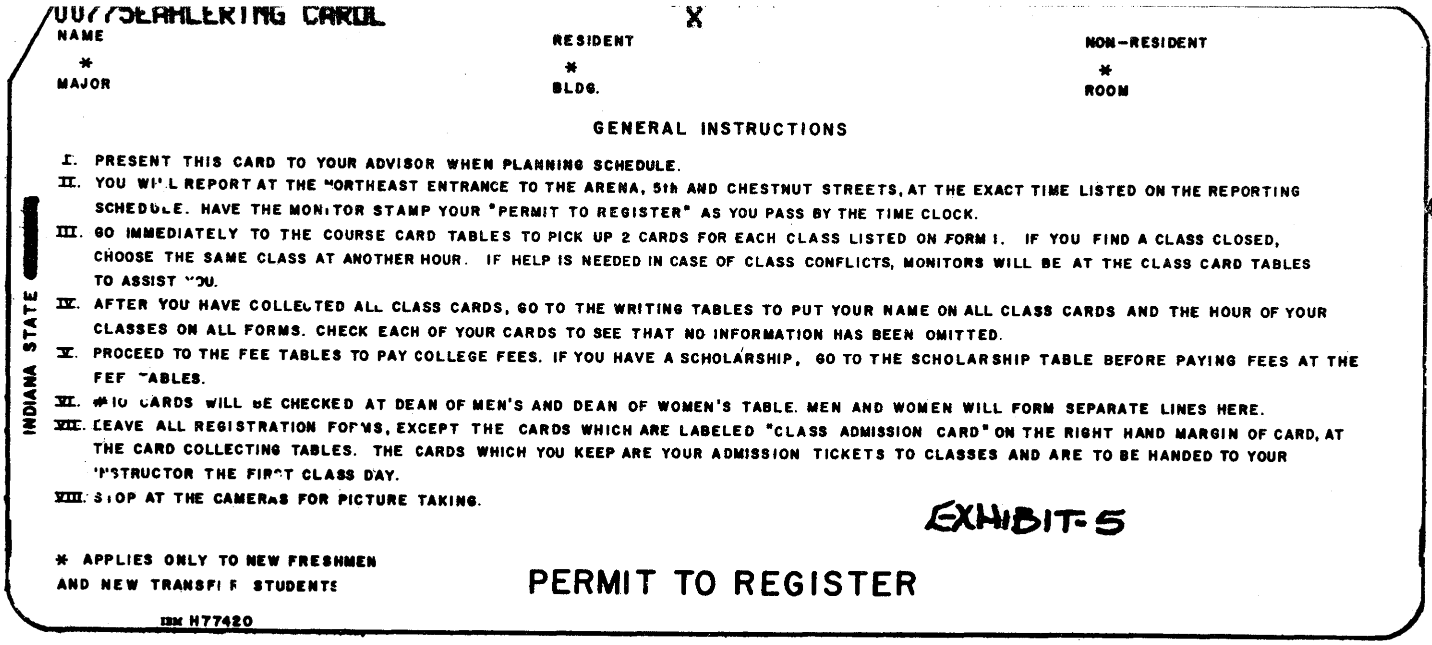


EXHIBIT 4



UU775EAHLERING CAROL X

NAME

RESIDENT

NON-RESIDENT

*
MAJOR

*
BLDG.

*
ROOM

INDIANA STATE

GENERAL INSTRUCTIONS

I. PRESENT THIS CARD TO YOUR ADVISOR WHEN PLANNING SCHEDULE.

II. YOU WI'.L REPORT AT THE "ORTHEAST ENTRANCE TO THE ARENA, 5th AND CHESTNUT STREETS, AT THE EXACT TIME LISTED ON THE REPORTING SCHEDULE. HAVE THE MONITOR STAMP YOUR "PERMIT TO REGISTER" AS YOU PASS BY THE TIME CLOCK.

III. GO IMMEDIATELY TO THE COURSE CARD TABLES TO PICK UP 2 CARDS FOR EACH CLASS LISTED ON FORM I. IF YOU FIND A CLASS CLOSED, CHOOSE THE SAME CLASS AT ANOTHER HOUR. IF HELP IS NEEDED IN CASE OF CLASS CONFLICTS, MONITORS WILL BE AT THE CLASS CARD TABLES TO ASSIST "OU.

IV. AFTER YOU HAVE COLLECTED ALL CLASS CARDS, GO TO THE WRITING TABLES TO PUT YOUR NAME ON ALL CLASS CARDS AND THE HOUR OF YOUR CLASSES ON ALL FORMS. CHECK EACH OF YOUR CARDS TO SEE THAT NO INFORMATION HAS BEEN OMITTED.

V. PROCEED TO THE FEE TABLES TO PAY COLLEGE FEES. IF YOU HAVE A SCHOLARSHIP, GO TO THE SCHOLARSHIP TABLE BEFORE PAYING FEES AT THE FEE "ABLES.

VI. #10 CARDS WILL BE CHECKED AT DEAN OF MEN'S AND DEAN OF WOMEN'S TABLE. MEN AND WOMEN WILL FORM SEPARATE LINES HERE.

VII. LEAVE ALL REGISTRATION FORMS, EXCEPT THE CARDS WHICH ARE LABELED "CLASS ADMISSION CARD" ON THE RIGHT HAND MARGIN OF CARD, AT THE CARD COLLECTING TABLES. THE CARDS WHICH YOU KEEP ARE YOUR ADMISSION TICKETS TO CLASSES AND ARE TO BE HANDED TO YOUR "NSTRUCTOR THE FIR"T CLASS DAY.

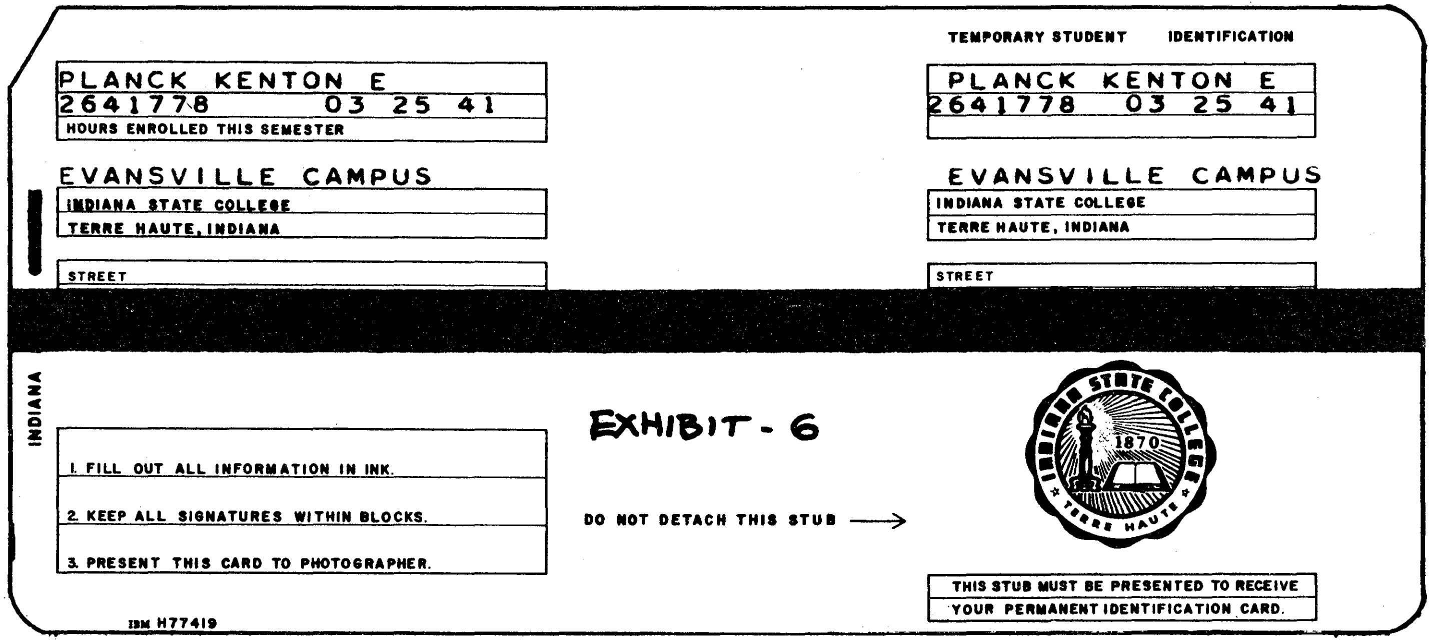
VIII. STOP AT THE CAMERAS FOR PICTURE TAKING.

EXHIBIT-5

* APPLIES ONLY TO NEW FRESHMEN
AND NEW TRANSFER STUDENTS

PERMIT TO REGISTER

IBM H77420



TEMPORARY STUDENT IDENTIFICATION

| | |
|---|---|
| PLANCK KENTON E | PLANCK KENTON E |
| 2641778 03 25 41 | 2641778 03 25 41 |
| HOURS ENROLLED THIS SEMESTER | |

EVANSVILLE CAMPUS

EVANSVILLE CAMPUS

| | |
|---|---|
| INDIANA STATE COLLEGE | INDIANA STATE COLLEGE |
| TERRE HAUTE, INDIANA | TERRE HAUTE, INDIANA |
| STREET | STREET |

INDIANA

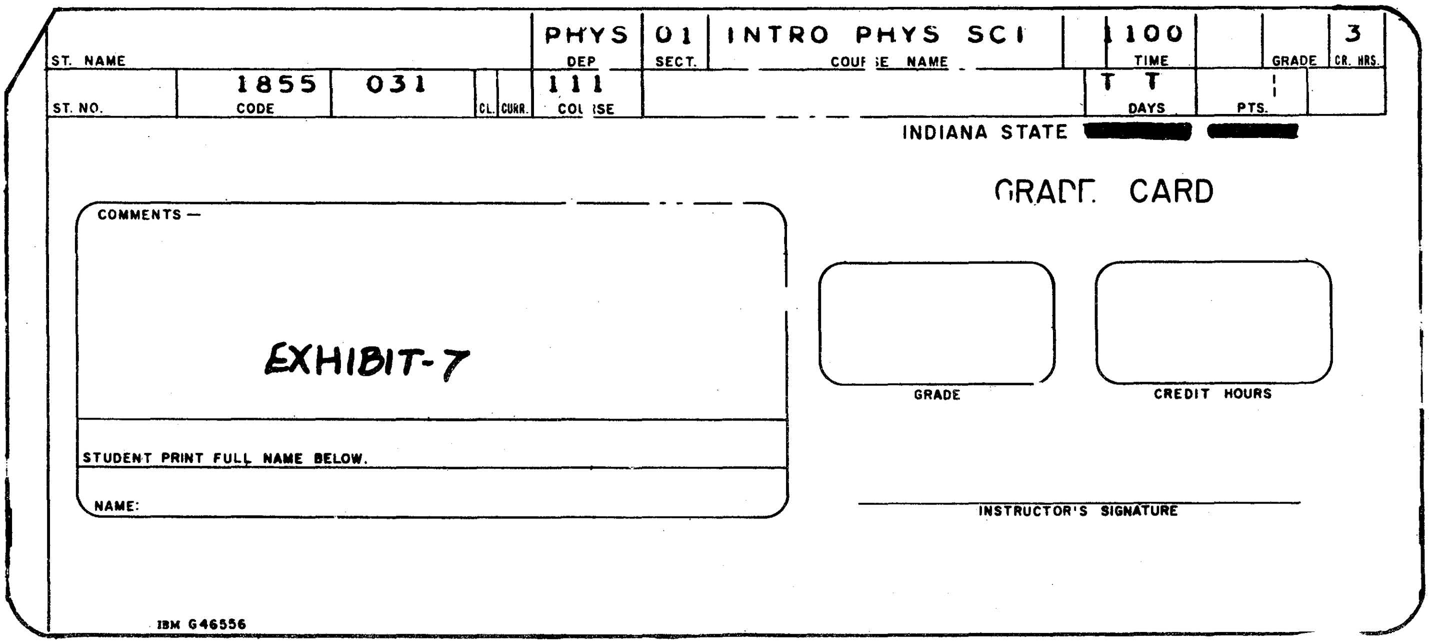
EXHIBIT - 6

1. FILL OUT ALL INFORMATION IN INK.
2. KEEP ALL SIGNATURES WITHIN BLOCKS.
3. PRESENT THIS CARD TO PHOTOGRAPHER.

DO NOT DETACH THIS STUB →

THIS STUB MUST BE PRESENTED TO RECEIVE
YOUR PERMANENT IDENTIFICATION CARD.

IBM H77419



| | | | | | PHYS | 01 | INTRO PHYS SCI | 1100 | | | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ST. NAME | | | | | DEP | SECT. | COURSE NAME | TIME | | GRADE | CR. HRS. |
| | 1855 | 031 | | | 111 | | | T T | | | |
| ST. NO. | CODE | | CL. | CURR. | COURSE | | | DAYS | PTS. | | |

INDIANA STATE

GRADE CARD

COMMENTS —

EXHIBIT-7

STUDENT PRINT FULL NAME BELOW.

NAME:

GRADE

CREDIT HOURS

INSTRUCTOR'S SIGNATURE

IBM G46556

# ISU EVANSVILLE CAMPUS

PLEASE PRINT IN INK

## REGISTRATION FORM

TODAY'S DATE

STUDENT NO.

NAME

HOME ADDRESS
STREET CITY COUNTY STATE PHONE

COLLEGE ADDRESS
STREET CITY STATE PHONE

U.S. CITIZEN

CLASSIFICATION FR. SO. JR. SR. SP. GR. ADV. POST GR.

SEX M F

DEGREE AB. B.S. M.S. M.A. M.B.A. OTHER

VETERAN YES NO

HIGH SCHOOL
NAME COUNTY STATE YR. OF GRADUATION

NEW STUDENT YES NO TRANSFER

TERM AND YR. ENTERING I.S.U.

DATE OF BIRTH

MARITAL STATUS

CHURCH PREFERENCE

CHECK ONE ONLY ELEMENTARY TCHG. SECONDARY TCHG. NON-TCHG.

MAJOR

MINOR

ADVISORS SIGNATURE

## COURSE SCHEDULE

| HOUR | DEPARTMENT | COURSE NO. | SEM. HRS. | MON. | TUE. | WED. | THUR. | FRI. | SAT | ROOM | DEPARTMENT AUTHORIZATION | CODE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | |

FOR BUSINESS OFFICE USE ONLY

SCHOLARSHIP

OTHER

REC. NO. BY

| | |
|---|---|
| TOTAL HOURS | |
| | AMOUNT |
| CONTINGENT FEE | |
| LATE FEE | |
| OTHER | |
| | |
| TOTAL FEES | |

FOR REGISTRAR'S OFFICE USE ONLY

EXHIBIT - 8

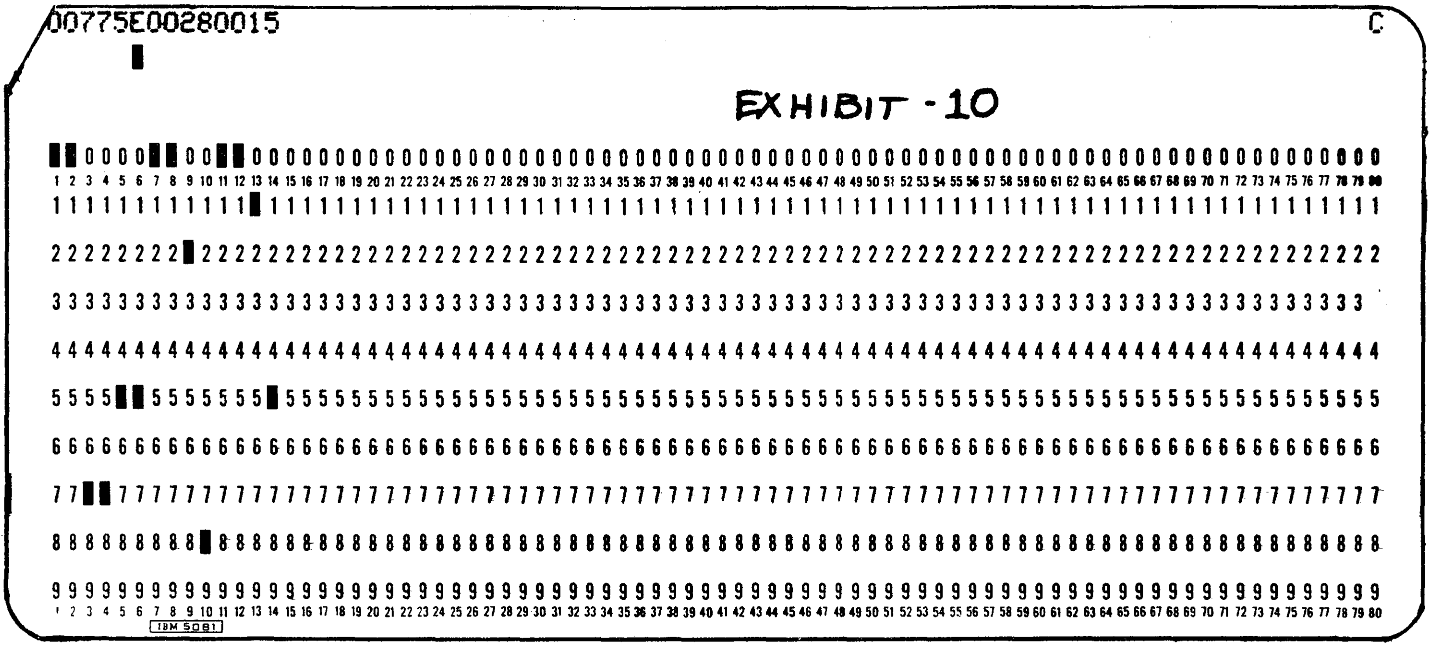
REGISTRAR

```
  46670                               FALL  1965-66

JOHN T. KLINE           201 OAKLAND DRIVE   TERRE HAUTE,INDIANA

HIGH SCHOOL CODE* 44-52  CLASSIFICATION * JUNIOR            SEX* MALE

MAJOR * MATHEMATICS    MARITAL STATUS * SINGLE  BIRTH DATE * 08/21/44

COUNTY * VIGO      STATE * INDIANA    PERMANENT NUMBER * 164-9987

NON-TEACHING CURRICULUM  SAT * VERBAL- 25 MATH- 99  CLASS RANK*  88/ 99

ACT * ENGLISH -95 MATH-99 SOCIAL STUDIES-84 SCIENCE-98 COMPOSIT-94

ACE * QUANTITATIVE-58 LANGUAGE-89 ENGLISH-77 TOTAL- 90

-------------------------------------------------------------------------

  DEPT COURSE  DAY+HOUR    CREDIT          DEPT COURSE  DAY+HOUR    CREDIT

   MATH 400    M W F  8.00   4 HRS         LANG   172  MTWTF  6.15   3 HRS

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   TOTAL HOURS THIS SEMESTER                                           35
```

```
  46670                               FALL  1965-66

JOHN T. KLINE           201 OAKLAND DRIVE   TERRE HAUTE,INDIANA

HIGH SCHOOL CODE* 44-52  CLASSIFICATION * JUNIOR            SEX* MALE

MAJOR * MATHEMATICS    MARITAL STATUS * SINGLE  BIRTH DATE * 08/21/44

COUNTY * VIGO      STATE * INDIANA    PERMANENT NUMBER * 164-9987

NON-TEACHING CURRICULUM  SAT * VERBAL- 25 MATH- 99  CLASS RANK*  88/ 99

CUMULATIVE DATA** HOURS* 96.0  POINTS*258.0  RATIO* 2.68


-------------------------------------------------------------------------

  DEPT COURSE  DAY+HOUR    CREDIT          DEPT COURSE  DAY+HOUR    CREDIT

   MATH 400    M W F  8.00   4 HRS         LANG   172  MTWTF  6.15   3 HRS

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   MATH 400    M W F  8.00   4             LANG   172  MTWTF  6.15   3

   TOTAL HOURS THIS SEMESTER                                           35
```

EXHIBIT-9

426



00775E00280015 C

EXHIBIT -10

```
██0000██00██00000000000000000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
111111111111█1111111111111111111111111111111111111111111111111111111111111111111
22222222█22222222222222222222222222222222222222222222222222222222222222222222222
33333333333333333333333333333333333333333333333333333333333333333333333333333333
44444444444444444444444444444444444444444444444444444444444444444444444444444444
5555██5555555█555555555555555555555555555555555555555555555555555555555555555555
66666666666666666666666666666666666666666666666666666666666666666666666666666666
77██7777777777777777777777777777777777777777777777777777777777777777777777777777
888888888█8888888888888888888888888888888888888888888888888888888888888888888888
99999999999999999999999999999999999999999999999999999999999999999999999999999999
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
IBM 5081
```

# ISU TEMPORARY CLASS LIST

To the Instructor: The students listed below registered for this course during the regular registration period. Changes in the roster should be made only upon receipt of official Notice of Registration or Add or Drop cards. Add the names of those who register late and indicate the date. Draw a line through the names of those who officially withdraw and indicate the date. Please maintain this record accurately.

| DEPT. | COURSE NO. | SEC. | DESCRIPTIVE COURSE TITLE | HRS. | SEM. | DAYS | CODE |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| INSTRUCTOR |
|---|
| |

| STUDENTS NO. & NAME | CL. | DATE ENTERED | DATE WITHDRAWN | TEMPORARY ATTENDANCE RECORD | | | | | | | | MID-TERM GRADE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| EXHIBIT 11 | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

| TOTAL STUDENTS | 20 |
|---|---|

INSTRUCTORS SIGNATURE

# INDIANA STATE

## COMPUTER CENTER – TEN DAY REPORT

09/16/65

SEM. 1ST 19 65

| INSTRUCTOR | DEPARTMENT | COURSE NO. | TIME AND DAYS | | ROOM & BUILDING | SECTION | SEM. HOURS | ENROLLMENT |
|---|---|---|---|---|---|---|---|---|
| MARTIN | ART | 151 | 230 | T T | 28E | 1 | 2 | 045 |
| DAVIS | BIO | 112 | 700 | M W | 23E | 1 | 3 | 027 |
| STRALEY | BIO | 112 | 530 | M W | 22E | 2 | 3 | 025 |
| HESS | BUS | 140 | 400 | M W | 25E | 1 | 3 | 052 |
| KELLEY | BUS | 201 | 530 | M W | 25E | 1 | 3 | 038 |
| LYNCH | ENG | 101 | 230 | M W | 7W | 1 | 3 | 025 |
| HARDESTY | ENG | 101 | 530 | M W | 6W | 5 | 3 | 027 |
| HOLLEY | ENG | 101 | 400 | T T | 7W | 6 | 3 | 027 |
| LYNCH | ENG | 220 | 400 | M W | 6W | 1 | 3 | 066 |
| MCDONALD | GEO | 111 | 700 | M W | 26E | 1 | 3 | 056 |
| KELL | HIST | 151 | 530 | M W | 26E | 2 | 3 | 051 |
| WAHNSIEDLER | HIST | 261 | 230 | T T | 26E | 1 | 3 | 053 |
| O LEARY | LSCI | 206 | 530 | T | 5W | | 3 | 021 |
| LABHART | MATH | 104 | 700 | M W | 27E | 1 | 4 | 046 |
| LABHART | MATH | 104 | 700 | T T | 27E | 2 | 4 | 039 |
| LABHART | MATH | 115 | 530 | T T | 27E | 1 | 3 | 015 |
| ROBERTS | PSCI | 130 | 400 | T T | 5W | 1 | 3 | 046 |
| VONFUHRMANN | ENG | 101 | 530 | T T | 8W | 4 | 3 | 029 |
| CUTLER | GEO | 111 | 530 | T | 7W | 2 | 3 | 063 |

429

EXHIBIT 12

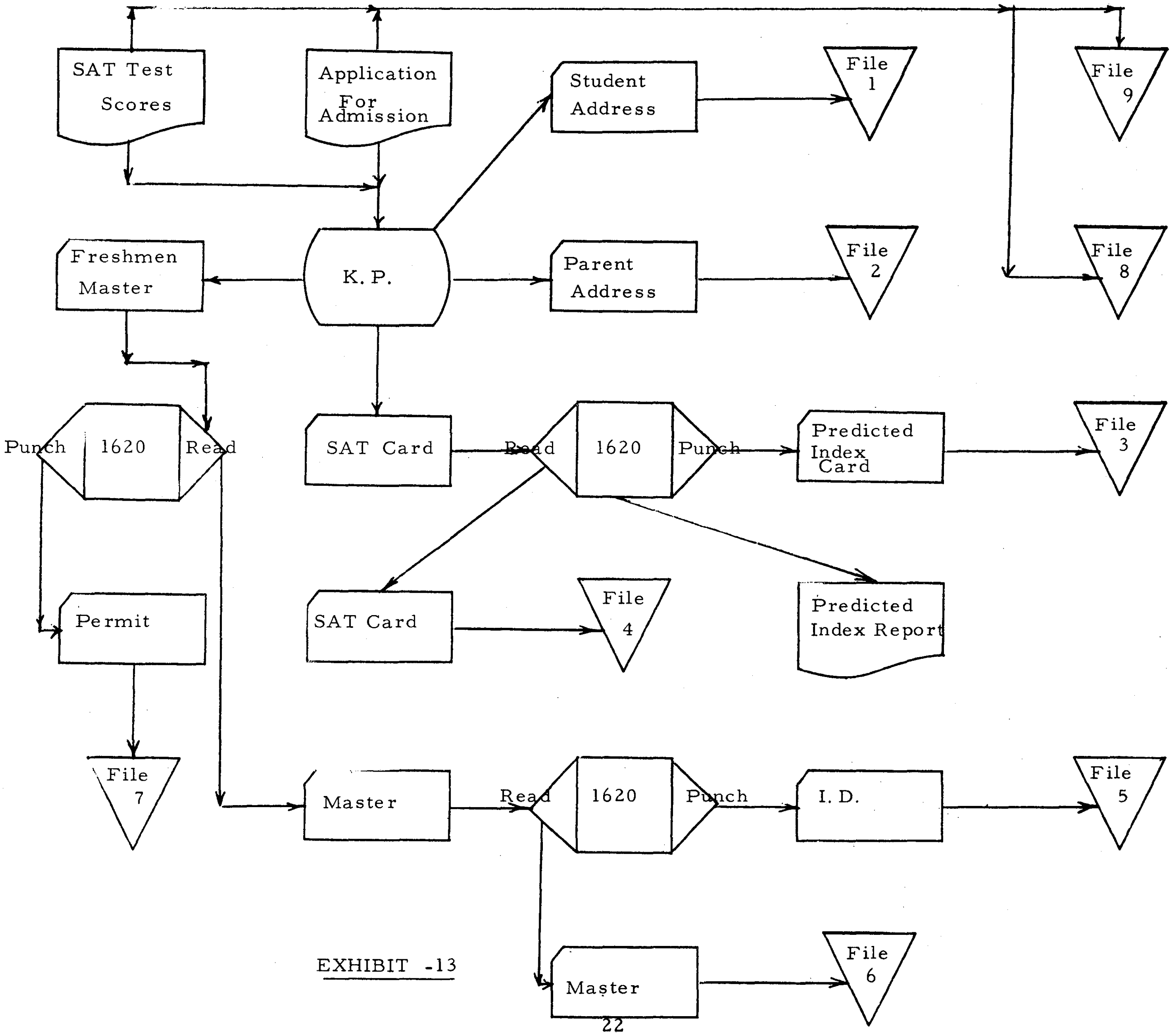

SAT Test Scores
Application For Admission
Student Address
File 1
File 9
Freshmen Master
K. P.
Parent Address
File 2
File 8
Punch
1620
Read
SAT Card
Read
1620
Punch
Predicted Index Card
File 3
Permit
SAT Card
File 4
Predicted Index Report
File 7
Master
Read
1620
Punch
I. D.
File 5
Master
File 6

EXHIBIT -13

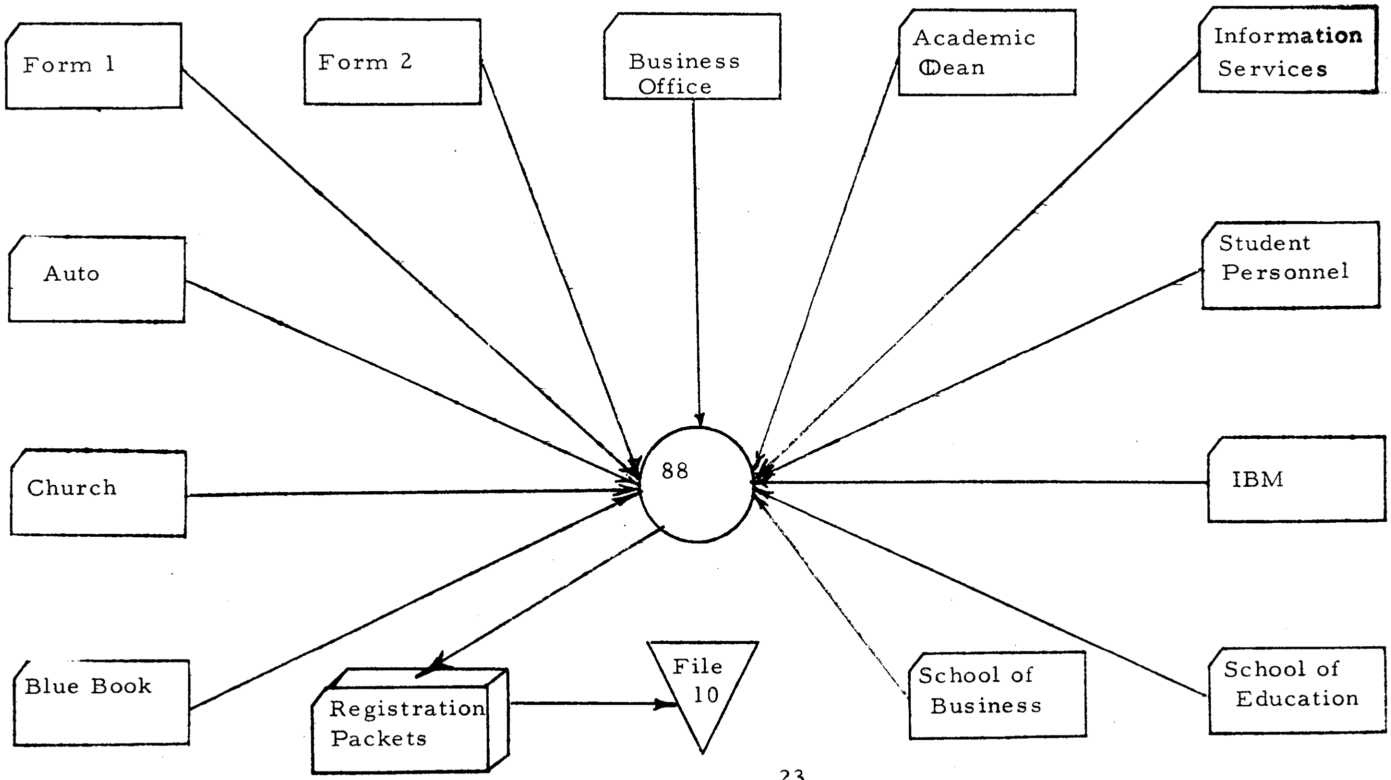

Form 1
Form 2
Business Office
Academic Dean
Information Services
Auto
Student Personnel
Church
88
IBM
Blue Book
Registration Packets
File 10
School of Business
School of Education

Exhibit-14

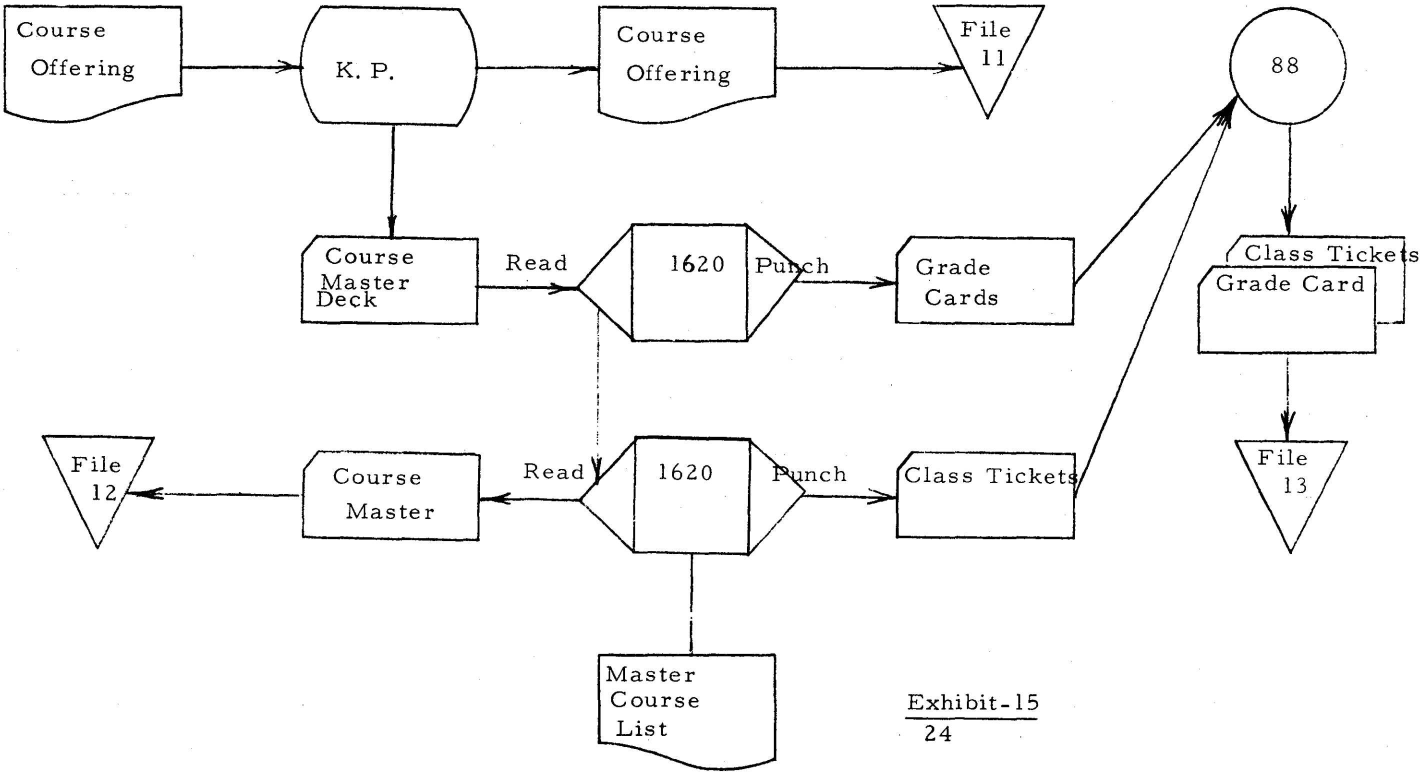


Exhibit-15
24

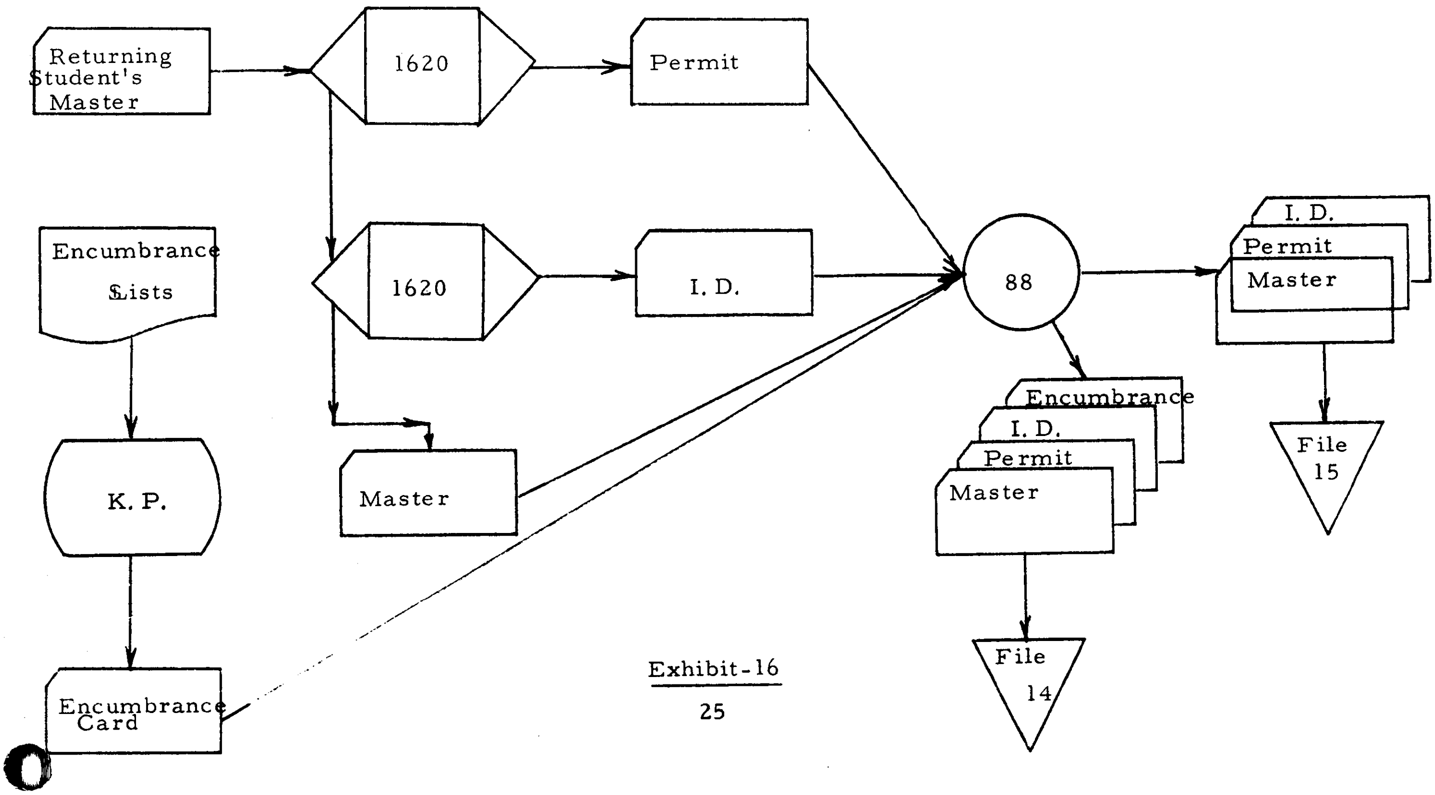


Exhibit-16

X 33

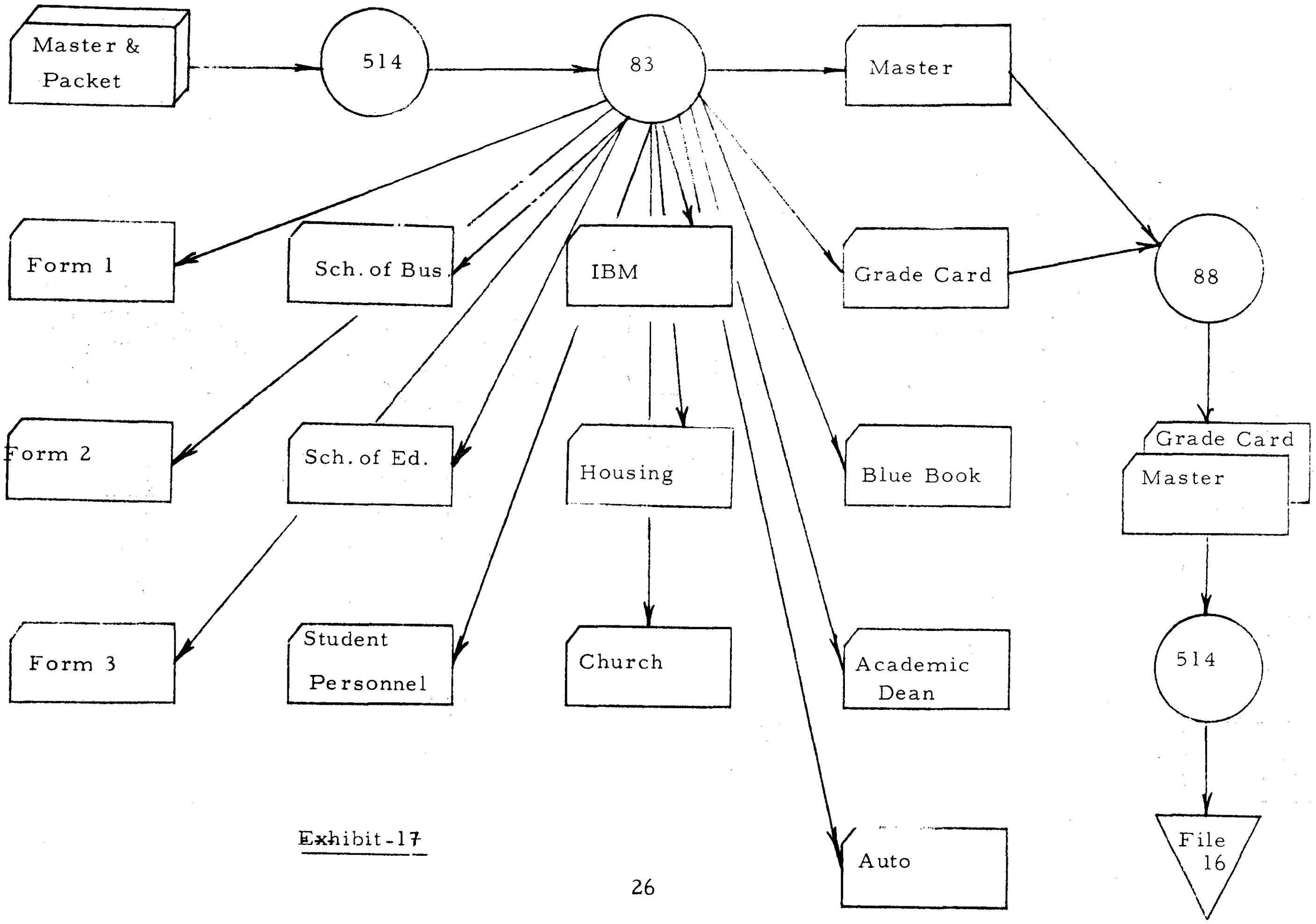


Exhibit-17

43K

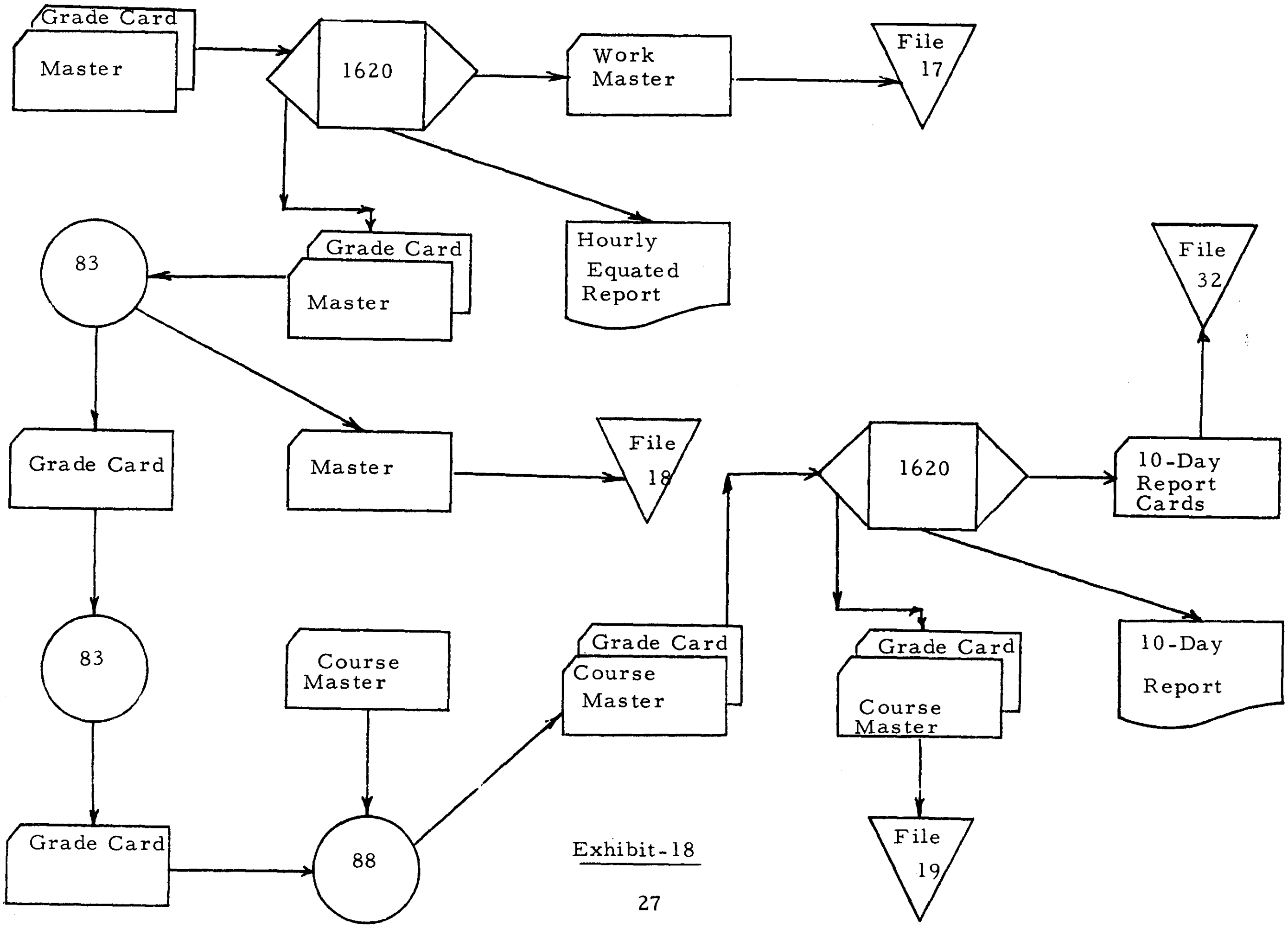

Grade Card
Master
1620
Work Master
File 17
Grade Card
Master
Hourly Equated Report
83
File 32
Grade Card
Master
File 18
1620
10-Day Report Cards
83
Course Master
Grade Card
Course Master
Grade Card
Course Master
10-Day Report
Grade Card
88
File 19

Exhibit-18

| CODE | DEPT | SEC. | TITLE | HR. | DAY | ROOM | SEM. HRS. | LIMIT | TALLY |
|---|---|---|---|---|---|---|---|---|---|
| 0942 | MATH445 | 02 | PRIN DIGIT COMP | 100 | F | LM 207 | 2 | 013 | |
| 0946 | MATH511 | | THEORY NUMBERS | 300 | T T | LM 217 | 2 | 015 | |
| 0949 | MATH516 | | THEORY MATRICES | 900 | T T | LM 208 | 2 | 015 | |
| 0950 | MATH525 | | NON EUCLID GEOM | 1000 | T T | LM 208 | 2 | 015 | |
| 0951 | MATH526 | | TOPOLOGY | 1000 | M W F | LM 208 | 3 | 015 | |
| 0952 | MATH530 | | INTERM ANALYSIS | 200 | M W F | LM 208 | 3 | 015 | |
| 0953 | MATH533 | 01 | DIFF EQUATIONS | 900 | M W F | LM 208 | 3 | 015 | |
| 0954 | MATH533 | 02 | DIFF EQUATIONS | 1100 | M W F | LM 217 | 3 | 015 | |
| 0955 | MATH534 | | ADV DIFF EQUAT | 200 | T T | LM 208 | 2 | 015 | |
| 0956 | MATH535 | | INT VECTOR ANAL | 1200 | M W F | LM 208 | 3 | 015 | |
| 0957 | MATH536 | | NUM ANALYSIS 1 | 800 | T T | HE 208 | 3 | 015 | |
| 0958 | MATH545 | 01 | PRIN DIGIT COMP | 800 | F | LM 217 | 2 | 012 | |
| 0970 | MATH200 | | CALCULUS 1 | ARR | | ARR | 4 | | |
| 0972 | MATH200 | | CALCULUS 1 | ARR | | ARR | 4 | | |

EXHIBIT 19

#36

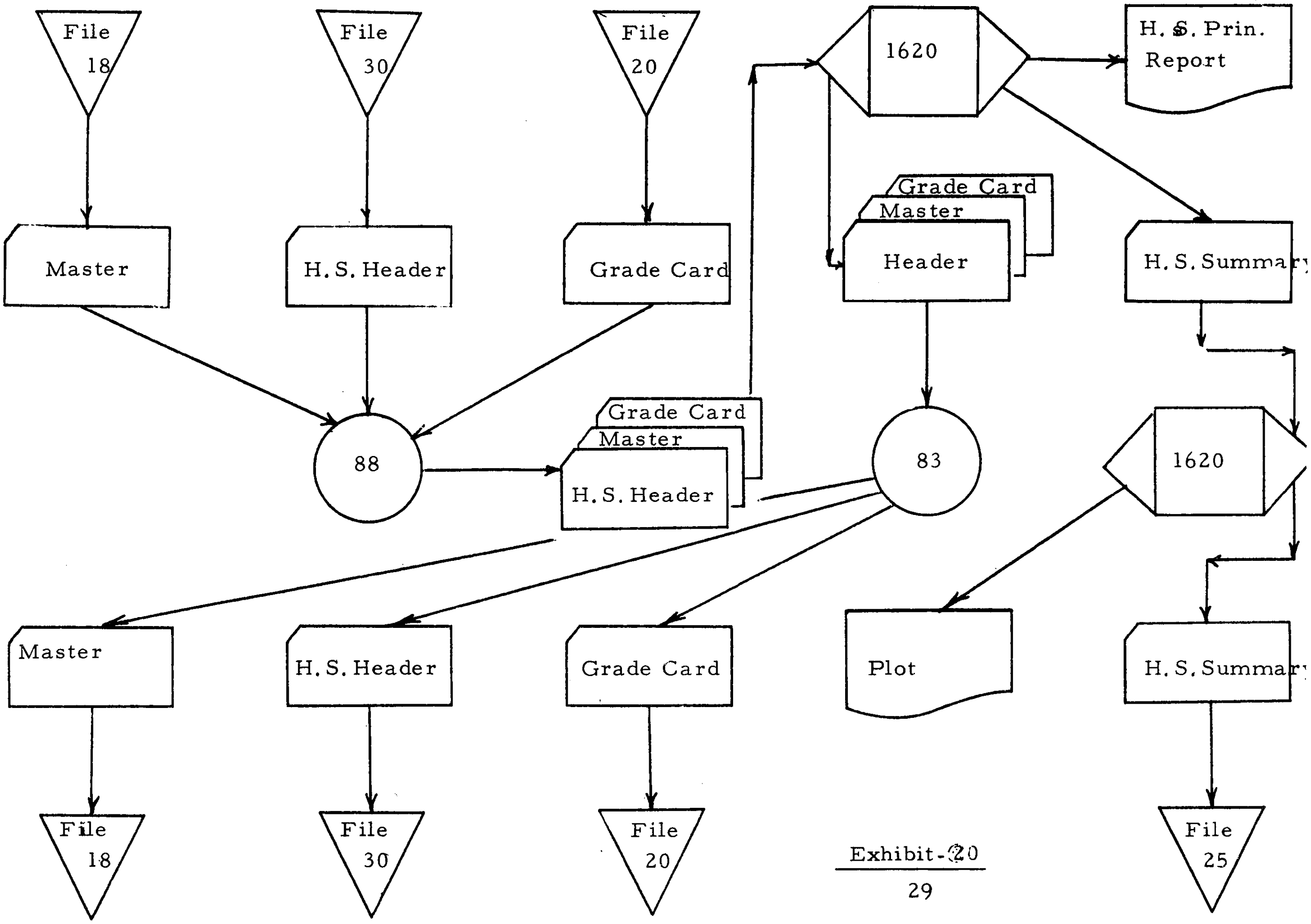

File 18
File 30
File 20
1620
H. S. Prin. Report
Master
H. S. Header
Grade Card
Grade Card
Master
Header
H. S. Summar
88
Grade Card
Master
H. S. Header
83
1620
Master
H. S. Header
Grade Card
Plot
H. S. Summar
File 18
File 30
File 20
File 25

Exhibit-20

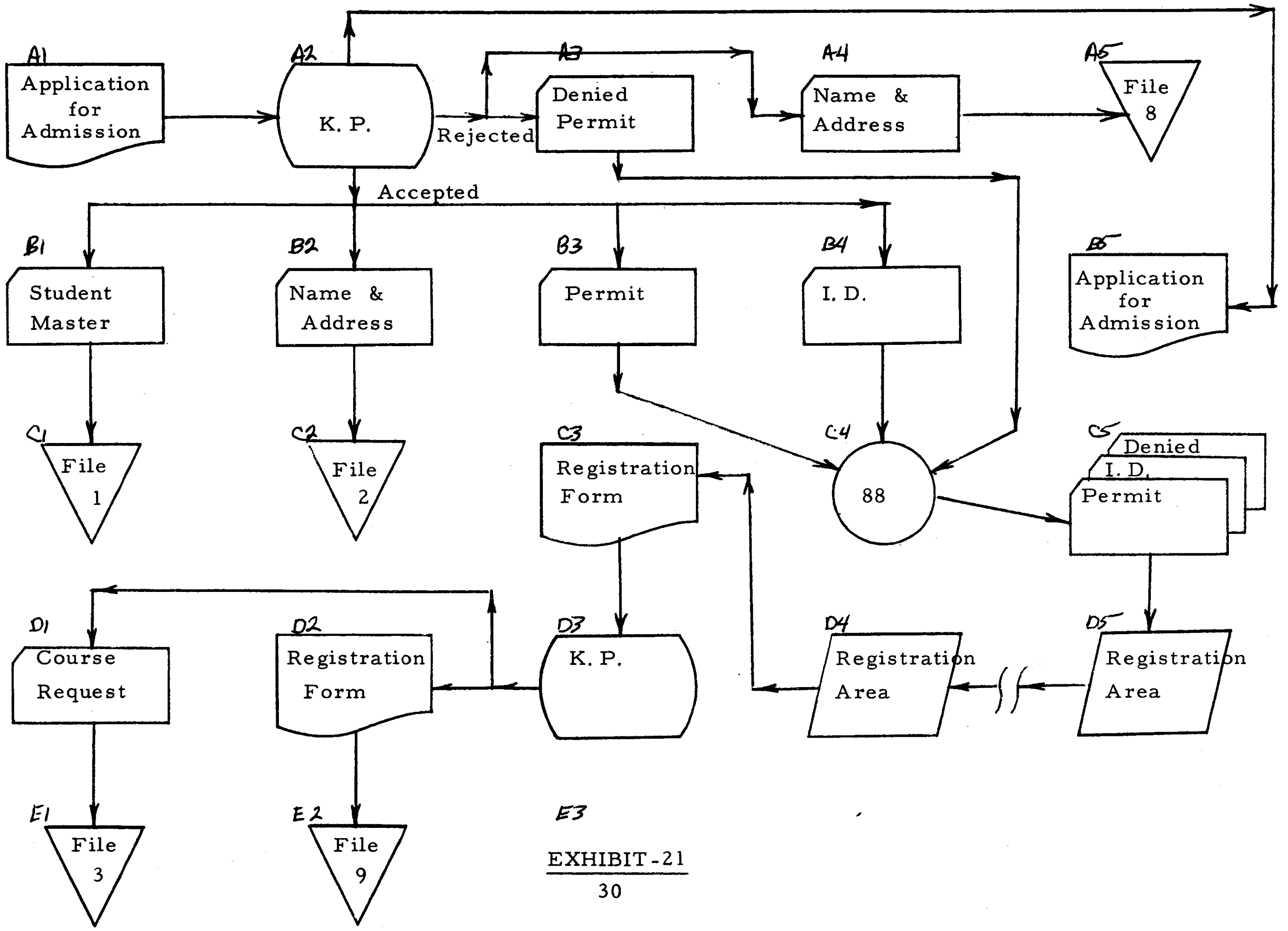

A1
Application for Admission
A2
K. P.
Rejected
Accepted
A3
Denied Permit
A4
Name & Address
A5
File 8
B1
Student Master
B2
Name & Address
B3
Permit
B4
I. D.
B5
Application for Admission
C1
File 1
C2
File 2
C3
Registration Form
C4
88
C5
Denied
I. D.
Permit
D1
Course Request
D2
Registration Form
D3
K. P.
D4
Registration Area
D5
Registration Area
E1
File 3
E2
File 9
E3

EXHIBIT-21

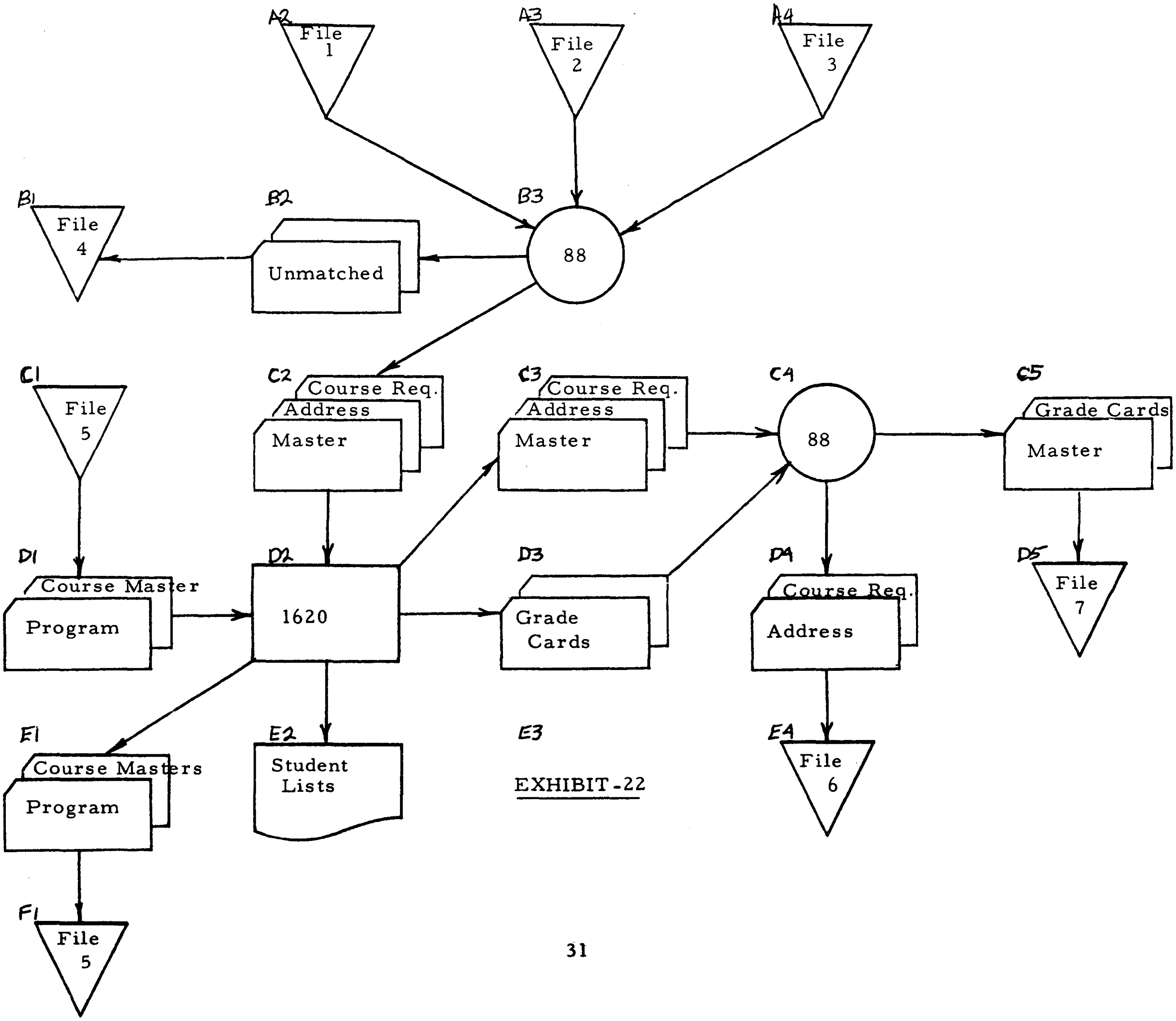

A2
File
1
A3
File
2
A4
File
3
B1
File
4
B2
Unmatched
B3
88
C1
File
5
C2
Course Req.
Address
Master
C3
Course Req.
Address
Master
C4
88
C5
Grade Cards
Master
D1
Course Master
Program
D2
1620
D3
Grade
Cards
D4
Course Req.
Address
D5
File
7
E1
Course Masters
Program
E2
Student
Lists
E3
E4
File
6
F1
File
5

EXHIBIT-22

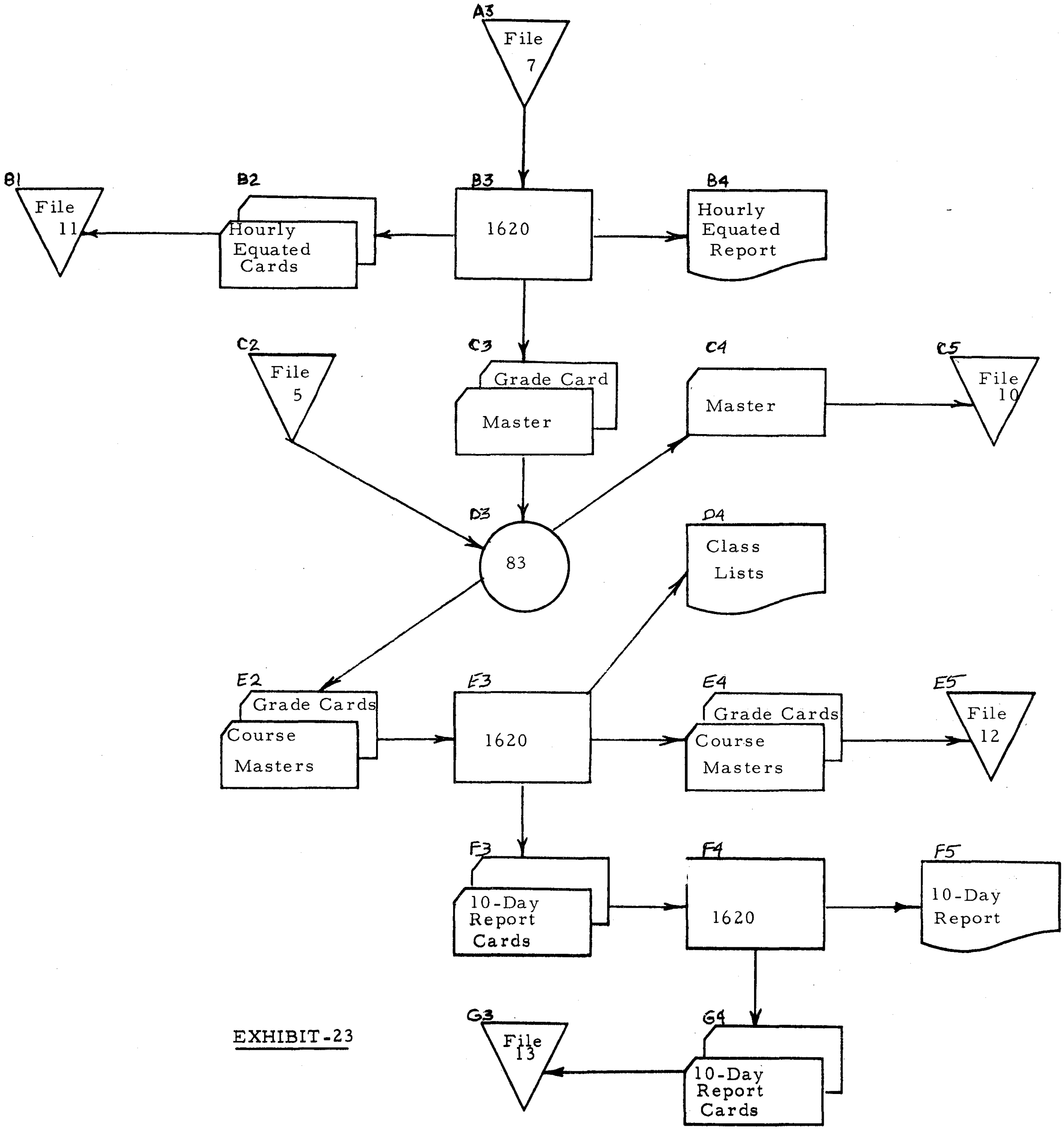

A3
File
7
B1
File
11
B2
Hourly
Equated
Cards
B3
1620
B4
Hourly
Equated
Report
C2
File
5
C3
Grade Card
Master
C4
Master
C5
File
10
D3
83
D4
Class
Lists
E2
Grade Cards
Course
Masters
E3
1620
E4
Grade Cards
Course
Masters
E5
File
12
F3
10-Day
Report
Cards
F4
1620
F5
10-Day
Report
G3
File
13
G4
10-Day
Report
Cards

EXHIBIT-23

## HOURLY EQUATED REPORT

| HOURS | FRESHMEN | | | SOPHOMORES | | | JUNIORS | | | SENIORS | | | GRADUATES | | | AUDITORS/SPECIAL | | | ALL STUDENTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TAKEN | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL, | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL |
| STUDENTS 22 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 5 |
| HOURS | 0 | 22 | 22 | 0 | 22 | 22 | 22 | 0 | 22 | 44 | 0 | 44 | 0 | 0 | 0 | 0 | 0 | 0 | 66 | 44 | 110 |
| STUDENTS 21 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 5 |
| HOURS | 21 | 0 | 21 | 21 | 0 | 21 | 42 | 21 | 63 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 84 | 21 | 105 |
| STUDENTS 20 | 0 | 0 | 0 | 1 | 0 | 1 | 5 | 2 | 7 | 6 | 3 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 5 | 17 |
| HOURS | 0 | 0 | 0 | 20 | 0 | 20 | 100 | 40 | 140 | 120 | 60 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 240 | 100 | 340 |
| STUDENTS 19 | 4 | 0 | 4 | 2 | 1 | 3 | 6 | 4 | 10 | 4 | 8 | 12 | 1 | 0 | 1 | 0 | 0 | 0 | 17 | 13 | 30 |
| HOURS | 76 | 0 | 76 | 38 | 19 | 57 | 114 | 76 | 190 | 76 | 152 | 228 | 19 | 0 | 19 | 0 | 0 | 0 | 323 | 247 | 570 |
| STUDENTS 18 | 13 | 20 | 33 | 35 | 31 | 66 | 60 | 45 | 105 | 31 | 19 | 50 | 1 | 0 | 1 | 0 | 0 | 0 | 140 | 115 | 255 |
| HOURS | 234 | 360 | 594 | 630 | 558 | 1188 | 1080 | 810 | 1890 | 558 | 342 | 900 | 18 | 0 | 18 | 0 | 0 | 0 | 2520 | 2070 | 4590 |
| STUDENTS 17 | 104 | 109 | 213 | 83 | 102 | 185 | 97 | 66 | 163 | 48 | 27 | 75 | 4 | 0 | 4 | 0 | 0 | 0 | 336 | 304 | 640 |
| HOURS | 1768 | 1853 | 3621 | 1411 | 1734 | 3145 | 1649 | 1122 | 2771 | 816 | 459 | 1275 | 68 | 0 | 68 | 0 | 0 | 0 | 5712 | 5168 | 10880 |
| STUDENTS 16 | 271 | 275 | 546 | 167 | 132 | 299 | 133 | 104 | 237 | 49 | 41 | 90 | 9 | 1 | 10 | 0 | 0 | 0 | 629 | 553 | 1182 |
| HOURS | 4336 | 4400 | 8736 | 2672 | 2112 | 4784 | 2128 | 1664 | 3792 | 784 | 656 | 1440 | 144 | 16 | 160 | 0 | 0 | 0 | 10064 | 8848 | 18912 |
| STUDENTS 15 | 362 | 227 | 589 | 196 | 137 | 333 | 107 | 85 | 192 | 155 | 186 | 341 | 23 | 9 | 32 | 0 | 0 | 0 | 843 | 644 | 1487 |
| HOURS | 5430 | 3405 | 8835 | 2940 | 2055 | 4995 | 1605 | 1275 | 2880 | 2325 | 2790 | 5115 | 345 | 135 | 480 | 0 | 0 | 0 | 12645 | 9660 | 22305 |
| STUDENTS 14 | 233 | 118 | 351 | 131 | 68 | 199 | 84 | 55 | 139 | 84 | 72 | 156 | 8 | 3 | 11 | 0 | 0 | 0 | 540 | 316 | 856 |
| HOURS | 3262 | 1652 | 4914 | 1834 | 952 | 2786 | 1176 | 770 | 1946 | 1176 | 1008 | 2184 | 112 | 42 | 154 | 0 | 0 | 0 | 7560 | 4424 | 11984 |
| STUDENTS 13 | 132 | 90 | 222 | 71 | 49 | 120 | 67 | 32 | 99 | 97 | 52 | 149 | 22 | 7 | 29 | 0 | 0 | 0 | 389 | 230 | 619 |
| HOURS | 1716 | 1170 | 2886 | 923 | 637 | 1560 | 871 | 416 | 1287 | 1261 | 676 | 1937 | 286 | 91 | 377 | 0 | 0 | 0 | 5057 | 2990 | 8047 |
| STUDENTS 12 | 161 | 63 | 224 | 118 | 52 | 170 | 74 | 32 | 106 | 36 | 19 | 55 | 38 | 12 | 50 | 1 | 0 | 1 | 428 | 178 | 606 |
| HOURS | 1932 | 756 | 2688 | 1416 | 624 | 2040 | 888 | 384 | 1272 | 432 | 228 | 660 | 456 | 144 | 600 | 12 | 0 | 12 | 5136 | 2136 | 7272 |
| SUBTOTAL | | | | | | | | | | | | | | | | | | | | | |
| STUDENTS | 1281 | 903 | 2184 | 805 | 573 | 1378 | 636 | 426 | 1062 | 512 | 427 | 939 | 106 | 32 | 138 | 1 | 0 | 1 | 3341 | 2361 | 5702 |
| HOURS | 18775 | 13618 | 32393 | 11905 | 8713 | 20618 | 9675 | 6578 | 16253 | 7592 | 6371 | 13963 | 1448 | 428 | 1876 | 12 | 0 | 12 | 49407 | 35708 | 85115 |

## HOURLY EQUATED REPORT

| HOURS | FRESHMEN | | | SOPHOMORES | | | JUNIORS | | | SENIORS | | | GRADUATES | | | AUDITORS/SPECIAL | | | ALL STUDENTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TAKEN | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL, | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL | MEN | WOMEN | TOTAL |
| STUDENTS 11 | 22 | 9 | 31 | 18 | 10 | 28 | 12 | 7 | 19 | 19 | 9 | 28 | 19 | 6 | 25 | 0 | 0 | 0 | 90 | 41 | 131 |
| HOURS | 242 | 99 | 341 | 198 | 110 | 308 | 132 | 77 | 209 | 209 | 99 | 308 | 209 | 66 | 275 | 0 | 0 | 0 | 990 | 451 | 1441 |
| STUDENTS 10 | 15 | 5 | 20 | 11 | 10 | 21 | 9 | 9 | 18 | 9 | 6 | 15 | 15 | 10 | 25 | 0 | 0 | 0 | 59 | 40 | 99 |
| HOURS | 150 | 50 | 200 | 110 | 100 | 210 | 90 | 90 | 180 | 90 | 60 | 150 | 150 | 100 | 250 | 0 | 0 | 0 | 590 | 400 | 990 |
| STUDENTS 9 | 18 | 28 | 46 | 10 | 8 | 18 | 7 | 11 | 18 | 8 | 12 | 20 | 25 | 12 | 37 | 0 | 0 | 0 | 68 | 71 | 139 |
| HOURS | 162 | 252 | 414 | 90 | 72 | 162 | 63 | 99 | 162 | 72 | 108 | 180 | 225 | 108 | 333 | 0 | 0 | 0 | 612 | 639 | 1251 |
| STUDENTS 8 | 11 | 6 | 17 | 4 | 2 | 6 | 3 | 6 | 9 | 9 | 4 | 13 | 14 | 10 | 24 | 0 | 0 | 0 | 41 | 28 | 69 |
| HOURS | 88 | 48 | 136 | 32 | 16 | 48 | 24 | 48 | 72 | 72 | 32 | 104 | 112 | 80 | 192 | 0 | 0 | 0 | 328 | 224 | 552 |
| STUDENTS 7 | 10 | 6 | 16 | 3 | 1 | 4 | 1 | 1 | 2 | 1 | 5 | 6 | 5 | 1 | 6 | 1 | 0 | 1 | 21 | 14 | 35 |
| HOURS | 70 | 42 | 112 | 21 | 7 | 28 | 7 | 7 | 14 | 7 | 35 | 42 | 35 | 7 | 42 | 7 | 0 | 7 | 147 | 98 | 245 |
| STUDENTS 6 | 18 | 12 | 30 | 16 | 5 | 21 | 6 | 5 | 11 | 12 | 6 | 18 | 89 | 44 | 133 | 0 | 0 | 0 | 141 | 72 | 213 |
| HOURS | 108 | 72 | 180 | 96 | 30 | 126 | 36 | 30 | 66 | 72 | 36 | 108 | 534 | 264 | 798 | 0 | 0 | 0 | 846 | 432 | 1278 |
| STUDENTS 5 | 15 | 5 | 20 | 4 | 10 | 14 | 6 | 2 | 8 | 5 | 2 | 7 | 23 | 17 | 40 | 1 | 2 | 3 | 54 | 38 | 92 |
| HOURS | 75 | 25 | 100 | 20 | 50 | 70 | 30 | 10 | 40 | 25 | 10 | 35 | 115 | 85 | 200 | 5 | 10 | 15 | 270 | 190 | 460 |
| STUDENTS 4 | 5 | 3 | 8 | 2 | 2 | 4 | 4 | 3 | 7 | 2 | 0 | 2 | 10 | 6 | 16 | 0 | 2 | 2 | 23 | 16 | 39 |
| HOURS | 20 | 12 | 32 | 8 | 8 | 16 | 16 | 12 | 28 | 8 | 0 | 8 | 40 | 24 | 64 | 0 | 8 | 8 | 92 | 64 | 156 |
| STUDENTS 3 | 52 | 51 | 103 | 18 | 16 | 34 | 10 | 12 | 22 | 14 | 19 | 33 | 258 | 235 | 493 | 13 | 9 | 22 | 365 | 342 | 707 |
| HOURS | 156 | 153 | 309 | 54 | 48 | 102 | 30 | 36 | 66 | 42 | 57 | 99 | 774 | 705 | 1479 | 39 | 27 | 66 | 1095 | 1026 | 2121 |
| STUDENTS 2 | 11 | 13 | 24 | 5 | 4 | 9 | 6 | 5 | 11 | 3 | 5 | 8 | 47 | 47 | 94 | 11 | 3 | 14 | 83 | 77 | 160 |
| HOURS | 22 | 26 | 48 | 10 | 8 | 18 | 12 | 10 | 22 | 6 | 10 | 16 | 94 | 94 | 188 | 22 | 6 | 28 | 166 | 154 | 320 |
| STUDENTS 1 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 3 |
| HOURS | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 3 |
| SUBTOTAL | | | | | | | | | | | | | | | | | | | | | |
| STUDENTS | 177 | 138 | 315 | 93 | 68 | 161 | 64 | 61 | 125 | 82 | 68 | 150 | 506 | 388 | 894 | 26 | 16 | 42 | 948 | 739 | 1687 |
| HOURS | 1093 | 779 | 1872 | 641 | 449 | 1090 | 440 | 419 | 859 | 603 | 447 | 1050 | 2289 | 1533 | 3822 | 73 | 51 | 124 | 5139 | 3678 | 8817 |
| GRAND TOTAL | | | | | | | | | | | | | | | | | | | | | |
| STUDENTS | 1458 | 1041 | 2499 | 898 | 641 | 1539 | 700 | 487 | 1187 | 594 | 495 | 1089 | 612 | 420 | 1032 | 27 | 16 | 43 | 4289 | 3100 | 7389 |
| HOURS | 19868 | 14397 | 34265 | 12546 | 9162 | 21708 | 10115 | 6997 | 17112 | 8195 | 6818 | 15013 | 3737 | 1961 | 5698 | 85 | 51 | 136 | 54546 | 39386 | 93932 |

EXHIBIT-24

|  | SAT |  |  |  | ACT |  |  | H S RANK |
|---|---|---|---|---|---|---|---|---|
| NAME | V | M | ENG | MA | SS | SCI | COMP |  |
| WOOD SUSAN KAY | 66 | 55 |  |  |  |  |  | 086/0230 |
| WOOD VIRGINIA CAROL |  |  | 36 | 78 | 46 | 46 | 57 | 003/0017 |
| WOODBURY RICHARD | 46 | 12 |  |  |  |  |  | 272/0403 |
| WOODS PAMELA SUE | 89 | 88 |  |  |  |  |  | 002/0026 |
| WOODS PENNY L | 81 | 76 |  |  |  |  |  | 017/0075 |
| WOODWORTH MICHAEL B | 50 | 46 |  |  |  |  |  | 109/0122 |
| WOOLS GERALD LEE | 59 | 86 |  |  |  |  |  | 109/0122 |
| WOOSLEY ERNIE G |  |  | 29 | 74 | 9 | 23 | 28 | 049/0117 |
| WORTHINGTON JOHN WM | 88 | 45 |  |  |  |  |  | 026/0115 |
| WOZNIAK THOMAS S | 74 | 60 |  |  |  |  |  | 069/0162 |
| WRAY PATRICIA A | 46 | 08 |  |  |  |  |  | 082/0163 |
| WRIGHT BILLY JOE | 81 | 46 |  |  |  |  |  | 052/0218 |
| WRIGHT CAROL ANN |  |  |  |  |  |  |  | 015/0130 |
| WRIGHT DEBORAH ANN | 71 | 69 |  |  |  |  |  | 064/0161 |
| WRIGHT DONALD LEE | 01 | 02 |  |  |  |  |  | 054/0067 |
| WRIGHT DONNA JEAN | 43 | 07 |  |  |  |  |  | 240/0341 |
| WRIGHT GARY ALLEN | 90 | 57 |  |  |  |  |  | 054/0109 |
| WRIGHT MARY JEAN |  |  |  |  |  |  |  | 009/0127 |
| WRIGHT SHARON KAY |  |  | 72 | 48 | 65 | 28 | 49 | 026/0053 |
| WUCHNER ANN LOUISE | 67 | 87 |  |  |  |  |  | 009/0080 |
| WUTHRICH SUSAN LYNN | 91 | 94 |  |  |  |  |  | 096/0496 |
| WYCOFF CAROLYN J | 58 | 18 |  |  |  |  |  | 084/0122 |
| WYLIE ROWENA SUE | 54 | 49 |  |  |  |  |  | 092/0140 |
| WYMAN LARRY EUGENE |  |  |  |  |  |  |  | 088/0129 |
| WYNDHAM RITA LYNN | 67 | 75 |  |  |  |  |  | 010/0140 |
| WYTHE WM FREDERICK | 50 | 64 |  |  |  |  |  | 048/0108 |
| YACKISH ELIZABETH J | 48 | 49 |  |  |  |  |  | 131/0174 |
| YARNALL CAROL SUE | 92 | 91 |  |  |  |  |  | 007/0064 |
| YEAGER DIANA LYNN | 65 | 52 |  |  |  |  |  | 042/0130 |
| YOMTOUBIAN CYRUS |  |  |  |  |  |  |  | / |
| YORK JULIA ANN |  |  |  |  |  |  |  | 044/0164 |
| YOUNG BARBARA JEAN |  |  | 65 | 64 | 20 | 33 | 42 | 098/0222 |
| YOUNG DENNIS R |  |  | 44 | 82 | 59 | 81 | 70 | 028/0040 |
| YURO SANDRA LEE | 61 | 34 |  |  |  |  |  | 118/0182 |
| ZAITCHICK HOWARD W |  |  | 44 | 78 | 71 | 46 | 64 | / |
| ZEHNER RICHARD M |  |  |  |  |  |  |  | 209/0317 |
| ZENTKO EVELYN MARIE |  |  |  |  |  |  |  | 023/0108 |
| ZERR ELAINE SUE | 39 | 68 |  |  |  |  |  | 055/0218 |
| ZIMMERMAN WIHELMENA |  |  |  |  |  |  |  | 017/0131 |
| ZSOLDOS EVELYN MARIE |  |  |  |  |  |  |  | 052/0178 |

EXHIBIT-25

| | SAT | | ACT | | | | | H S RANK |
|---|---|---|---|---|---|---|---|---|
| NAME | V | M | ENG | MA | SS | SCI | COMP | |
| WOOD SUSAN KAY | 66 | 55 | | | | | | 086/0230 |
| WOOD VIRGINIA CAROL | | | 36 | 78 | 46 | 46 | 57 | 003/0017 |
| WOODBURY RICHARD | 46 | 12 | | | | | | 272/0403 |
| WOODS PAMELA SUE | 89 | 88 | | | | | | 002/0026 |
| WOODS PENNY L | 81 | 76 | | | | | | 017/0075 |
| WOODWORTH MICHAEL B | 50 | 46 | | | | | | 109/0122 |
| WOOLS GERALD LEE | 59 | 86 | | | | | | 109/0122 |
| WOOSLEY ERNIE G | | | 29 | 74 | 9 | 23 | 28 | 049/0117 |
| WORTHINGTON JOHN WM | 88 | 45 | | | | | | 026/0115 |
| WOZNIAK THOMAS S | 74 | 60 | | | | | | 069/0162 |
| WRAY PATRICIA A | 46 | 08 | | | | | | 082/0163 |
| WRIGHT BILLY JOE | 81 | 46 | | | | | | 052/0218 |
| WRIGHT CAROL ANN | | | | | | | | 015/0130 |
| WRIGHT DEBORAH ANN | 71 | 69 | | | | | | 064/0161 |
| WRIGHT DONALD LEE | 01 | 02 | | | | | | 054/0067 |
| WRIGHT DONNA JEAN | 43 | 07 | | | | | | 240/0341 |
| WRIGHT GARY ALLEN | 90 | 57 | | | | | | 054/0109 |
| WRIGHT MARY JEAN | | | | | | | | 009/0127 |
| WRIGHT SHARON KAY | | | 72 | 48 | 65 | 28 | 49 | 026/0053 |
| WUCHNER ANN LOUISE | 67 | 87 | | | | | | 009/0080 |
| WUTHRICH SUSAN LYNN | 91 | 94 | | | | | | 096/0496 |
| WYCOFF CAROLYN J | 58 | 18 | | | | | | 084/0122 |
| WYLIE ROWENA SUE | 54 | 49 | | | | | | 092/0140 |
| WYMAN LARRY EUGENE | | | | | | | | 088/0129 |
| WYNDHAM RITA LYNN | 67 | 75 | | | | | | 010/0140 |
| WYTHE WM FREDERICK | 50 | 64 | | | | | | 048/0108 |
| YACKISH ELIZABETH J | 48 | 49 | | | | | | 131/0174 |
| YARNALL CAROL SUE | 92 | 91 | | | | | | 007/0064 |
| YEAGER DIANA LYNN | 65 | 52 | | | | | | 042/0130 |
| YOMTOUBIAN CYRUS | | | | | | | | / |
| YORK JULIA ANN | | | | | | | | 044/0164 |
| YOUNG BARBARA JEAN | | | 65 | 64 | 20 | 33 | 42 | 098/0222 |
| YOUNG DENNIS R | | | 44 | 82 | 59 | 81 | 70 | 028/0040 |
| YURO SANDRA LEE | 61 | 34 | | | | | | 118/0182 |
| ZAITCHICK HOWARD W | | | 44 | 78 | 71 | 46 | 64 | / |
| ZEHNER RICHARD M | | | | | | | | 209/0317 |
| ZENTKO EVELYN MARIE | | | | | | | | 023/0108 |
| ZERR ELAINE SUE | 39 | 68 | | | | | | 055/0218 |
| ZIMMERMAN WIHELMENA | | | | | | | | 017/0131 |
| ZSOLDOS EVELYN MARIE | | | | | | | | 052/0178 |

EXHIBIT-25

UNION DUGGER

| | SAT | | ACT | | | | | ACE | | | | RANK | GPR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STUDENT NAME | VERB | MATH | ENG | MATH | SOC | SCI | COMP | Q | L | T | ENG | IN CLASS | ACT. | PRED. |
| ALLEN DIANA KAY | | | | | | | | 11 | 17 | 12 | 29 | 0012/0040 | 1.70 | 1.45 |

| COURSE | GRADE | HOURS | POINTS |
|---|---|---|---|
| ENG 101 | D+ | 3 | 4.5 |
| ART 151 | D | 2 | 2.0 |
| P ED 151 | C+ | 3 | 7.5 |
| P ED 015 | C+ | 1 | 2.5 |
| P ED 001 | B | 1 | 3.0 |
| SP 101 | D+ | 2 | 3.0 |
| HIST 151 | D | 3 | 3.0 |
| | | --- | ----- |
| TOTAL | | 15 | 25.5 |

| STUDENT NAME | SAT VERB | SAT MATH | ACT ENG | ACT MATH | ACT SOC | ACT SCI | ACT COMP | ACE Q | ACE L | ACE T | ACE ENG | RANK IN CLASS | GPR ACT. | GPR PRED. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BEDWELL RONALD LEE | 335 | 379 | | | | | | | | | | 0015/0040 | 0.80 | 1.91 |

| COURSE | GRADE | HOURS | POINTS |
|---|---|---|---|
| I ED 161 | F | 2 | .0 |
| I ED 132 | C | 2 | 4.0 |
| SOC 170 | F | 3 | .0 |
| BIOL 112 | F | 3 | .0 |
| ENG 101 | C | 3 | 6.0 |
| I ED 050 | D | 1 | 1.0 |
| P ED 051B | D | 1 | 1.0 |
| | | --- | ----- |
| TOTAL | | 15 | 12.0 |

| STUDENT NAME | SAT VERB | SAT MATH | ACT ENG | ACT MATH | ACT SOC | ACT SCI | ACT COMP | ACE Q | ACE L | ACE T | ACE ENG | RANK IN CLASS | GPR ACT. | GPR PRED. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLY GEORGE RUSSELL | | | 25 | 40 | 53 | 50 | 39 | | | | | 0007/0039 | 2.77 | 1.73 |

| COURSE | GRADE | HOURS | POINTS |
|---|---|---|---|
| ART 151 | C | 2 | 4.0 |
| BIOL 115 | A | 3 | 12.0 |
| P ED 051B | B+ | 1 | 3.5 |
| ENG 100A | C | 1 | 2.0 |
| CHEM 105 | C | 4 | 8.0 |
| MATH 104 | B | 4 | 12.0 |
| | | --- | ----- |
| TOTAL | | 15 | 41.5 |

EXHIBIT-26

35

464

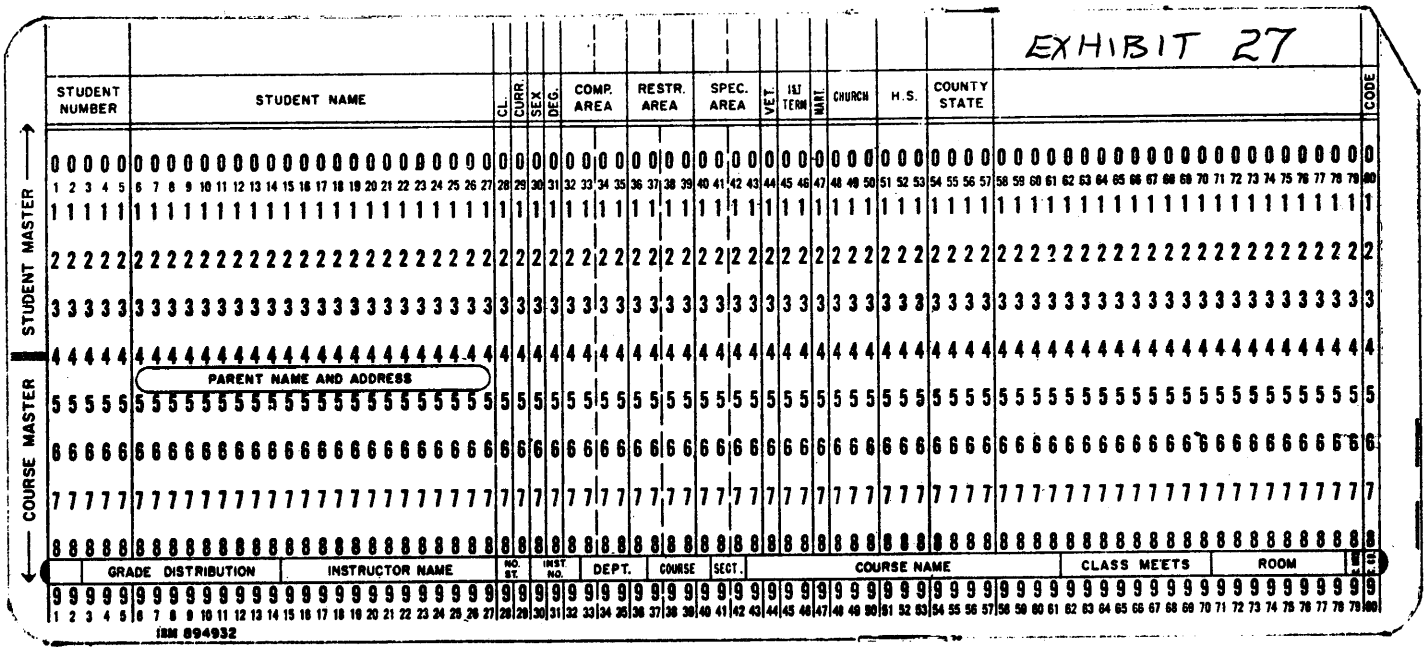


EXHIBIT 27

| STUDENT NUMBER | STUDENT NAME | CL. | CURR | SEX | DEG. | COMP. AREA | RESTR. AREA | SPEC. AREA | VET. | 1st TERM | MART. | CHURCH | H.S. | COUNTY STATE | | CODE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

STUDENT MASTER ⟶

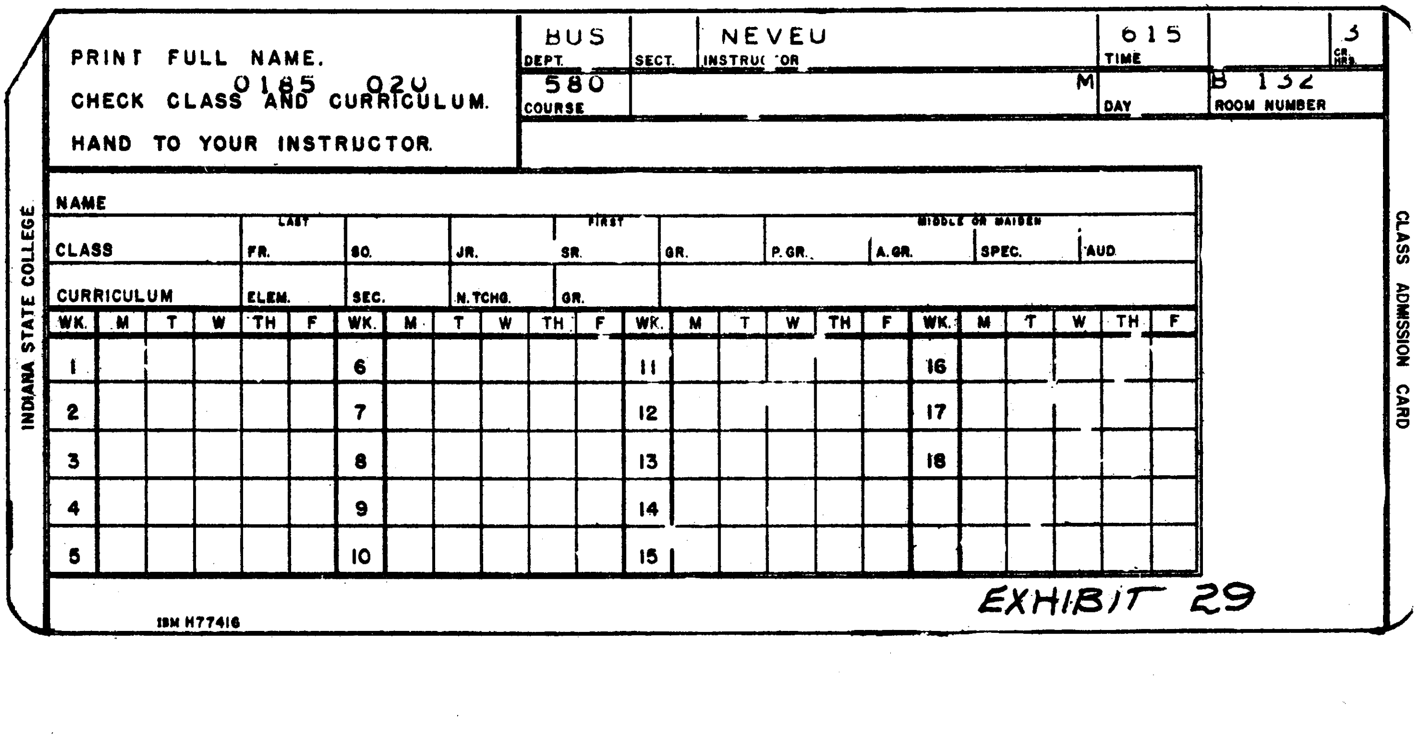
⟵ COURSE MASTER

```
00000 0000000000000000000000 0 0 0 0 0000 0000 0000 0 00 0 000 000 0000 0000000000000000000000 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
11111 1111111111111111111111 1 1 1 1 1111 1111 1111 1 11 1 111 111 1111 1111111111111111111111 1
22222 2222222222222222222222 2 2 2 2 2222 2222 2222 2 22 2 222 222 2222 2222222222222222222222 2
33333 3333333333333333333333 3 3 3 3 3333 3333 3333 3 33 3 333 333 3333 3333333333333333333333 3
44444 4444444444444444444444 4 4 4 4 4444 4444 4444 4 44 4 444 444 4444 4444444444444444444444 4
             PARENT NAME AND ADDRESS
55555 5555555555555555555555 5 5 5 5 5555 5555 5555 5 55 5 555 555 5555 5555555555555555555555 5
66666 6666666666666666666666 6 6 6 6 6666 6666 6666 6 66 6 666 666 6666 6666666666666666666666 6
77777 7777777777777777777777 7 7 7 7 7777 7777 7777 7 77 7 777 777 7777 7777777777777777777777 7
88888 8888888888888888888888 8 8 8 8 8888 8888 8888 8 88 8 888 888 8888 8888888888888888888888 8
```

| | GRADE DISTRIBUTION | INSTRUCTOR NAME | NO. ST. | INST. NO. | DEPT. | COURSE | SECT. | COURSE NAME | CLASS MEETS | ROOM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
99999 9999999999999999999999 9 9 9 9 9999 9999 9999 9 99 9 999 999 9999 9999999999999999999999 9
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

IBM 894932



INDIANA STATE COLLEGE

CLASS ADMISSION CARD

PRINT FULL NAME.
0185 020
CHECK CLASS AND CURRICULUM.
HAND TO YOUR INSTRUCTOR.

| | | | | | |
|---|---|---|---|---|---|
| BUS<br>DEPT. | SECT. | NEVEU<br>INSTRUC OR | 615<br>TIME | | 3<br>CR. HRS. |
| 580<br>COURSE | | M | DAY | B 132<br>ROOM NUMBER | |

NAME — LAST — FIRST — MIDDLE OR MAIDEN

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| CLASS | FR. | SO. | JR. | SR. | GR. | P.GR. | A.GR. | SPEC. | AUD. |
| CURRICULUM | ELEM. | SEC. | N.TCHG. | GR. | | | | | |

| WK. | M | T | W | TH | F | WK. | M | T | W | TH | F | WK. | M | T | W | TH | F | WK. | M | T | W | TH | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | 6 | | | | | | 11 | | | | | | 16 | | | | | |
| 2 | | | | | | 7 | | | | | | 12 | | | | | | 17 | | | | | |
| 3 | | | | | | 8 | | | | | | 13 | | | | | | 18 | | | | | |
| 4 | | | | | | 9 | | | | | | 14 | | | | | | | | | | | |
| 5 | | | | | | 10 | | | | | | 15 | | | | | | | | | | | |

EXHIBIT 29

IBM H77416

| CODE NO | DEPT | NO. | DESCRIPTION | SEC | TIME | DAYS | INSTRUCTOR | ROOM | CR |
|---|---|---|---|---|---|---|---|---|---|
| 0031 | BUS | 535 | ADV SECRET PRAC | | 1200 | MTWTF | ORNER | B 234 | 2 |
| 0064 | ED | 467 | MEAS + EVAL ED | 02 | 1200 | MTWTF | BEYMER | B 2 | 3 |
| 0065 | ED | 467 | MEAS + EVAL ED | 03 | 930 | MTWTF | OSMON | LC 1 | 3 |
| ~~0184~~ | ~~FREN~~ | ~~316~~ | ~~READ LA FANTAIN~~ | | ~~200~~ | ~~MTWTF~~ | ~~ALBRO~~ | ~~B 132~~ | ~~3~~ |
| 0204 | SPAN | 422 | THEATR GOLD AGE | | 730 | MTWTF | GOYER | LM 205 | 3 |
| 0230 | I ED | 121 | GENERAL METALS | | 1200 | MTWTF | LAWSON B | IE 111 | 2 |
| ~~0232~~ | ~~I ED~~ | ~~201~~ | ~~ADV TECH DRAW~~ | | ~~400~~ | ~~MTWTF~~ | ~~BURNS~~ | ~~IE 201~~ | ~~2~~ |
| 0236 | I ED | 325 | WELDING | | 1200 | MTWTF | LAWSON B | IE 111 | 3 |
| 0247 | I ED | 487 | CONF PANEL CASE | | 900 | MTWTF | JAMES | IE 315 | 2 |
| 0249 | I ED | 493 | WORKSH DRAFTING | 02 | 900 | MTWTF | CONAWAY | IE 201 | 1 |
| 0252 | I ED | 494 | MACH TOOL TECH | 02 | 900 | MTWTF | HALE | IE 315 | 1 |
| 0253 | I ED | 494 | VOC DAY TR WKSH | 03 | 900 | MTWTF | JAMES | I U | 1 |
| ~~0256~~ | ~~I ED~~ | ~~519~~ | ~~SPEC PROB WOOD~~ | | ~~730~~ | ~~MTWTF~~ | ~~SVENDSEN~~ | ~~IE .18~~ | ~~3~~ |
| 0264 | I ED | 587 | CONF PANEL CASE | | 900 | MTWTF | JAMES | IE 315 | 2 |
| 0274 | I ED | 529 | SPEC PROB METAL | | ARR | | TURNER | ARR | 2 |
| 0276 | I ED | 529 | SPEC PROB METAL | | ARR | | BARRICK | ARR | 3 |
| 0277 | I ED | 101 | TECHNICAL DRAW | | ARR | | MORTON | ARR | 2 |
| 0325 | MUS | 149 | PERCUSSION | | ARR | | CANEDY | FA 407 | 1 |
| 0333 | MUS | 250 | VOICE | | ARR | | HOUNCHELL | FA 316 | 1 |
| 0340 | MUS | 345 | | | ARR | | WATKINS | FA 028 | 1 |
| 0341 | MUS | 349 | PERCUSSION | | ARR | | CANEDY | FA 407 | 1 |
| ~~0342~~ | ~~MUS~~ | ~~350~~ | ~~VOICE~~ | | ~~ARR~~ | | ~~HOUNCHELL~~ | ~~FA 316~~ | ~~1~~ |
| 0344 | MUS | 351 | PIANO | 02 | ARR | | HARLAN | FA 232 | 1 |
| 0356 | MUS | 523 | PERCUSSION TECH | | 1200 | T T | CANEDY | FA 409 | 1 |
| 0364 | MUS | 571 | ADV WOODWINDS | | ARR | | GEE | FA 410 | 1 |
| 0365 | MUS | 572 | ADVANCE BRASSES | | ARR | | WATKINS | FA 028 | 1 |
| 0366 | MUS | 573 | ADV PERCUSSION | | ARR | | CANEDY | FA 407 | 1 |
| 0457 | SCED | 461 | SUPERV SCI K 12 | | 930 | MTWTF | UHLHORN | S 125 | 2 |
| 0494 | CHEM | 462 | PHYS CHEM II | | 730 | MTWTF | SMITH | S 25 | 4 |
| 0496 | CHEM | 463 | RADIOCHEMISTRY | | 930 | MTWTF | ORMOND | S 25 | 3 |
| ~~0536~~ | ~~PHYS~~ | ~~450~~ | ~~ELECTROMAG THEO~~ | | ~~1200~~ | ~~MTWTF~~ | ~~BESS~~ | ~~S 120~~ | ~~3~~ |
| 0539 | PHYS | 575 | PROJECT PHYSICS | | ARR | | HUGHES | ARR | |
| 0540 | PHYS | 585 | ADV TOPICS PHYS | | ARR | | HUGHES | ARR | |
| 0541 | PHYS | 598 | RESEARCH PHYSIC | | ARR | | HUGHES | ARR | |

EXHIBIT 28

FALL '64

Total Number Students (on campus) 7389

Total Semester Hours 93932

$\frac{93932}{15}$ = 6262.2 equated (full time) campus enrollment

| | | OFF CAMPUS<br>No. Students | Semester Hour |
|---|---|---|---|
| Extension Classes | 43 | 817 | 2382 |
| Correspondence Courses | | 738 | 1924 |
| TOTAL - Off Campus | | 1555 | 4306 |

$\frac{4306}{15}$ = 287.1 equated full time off campus

SUMMARY:

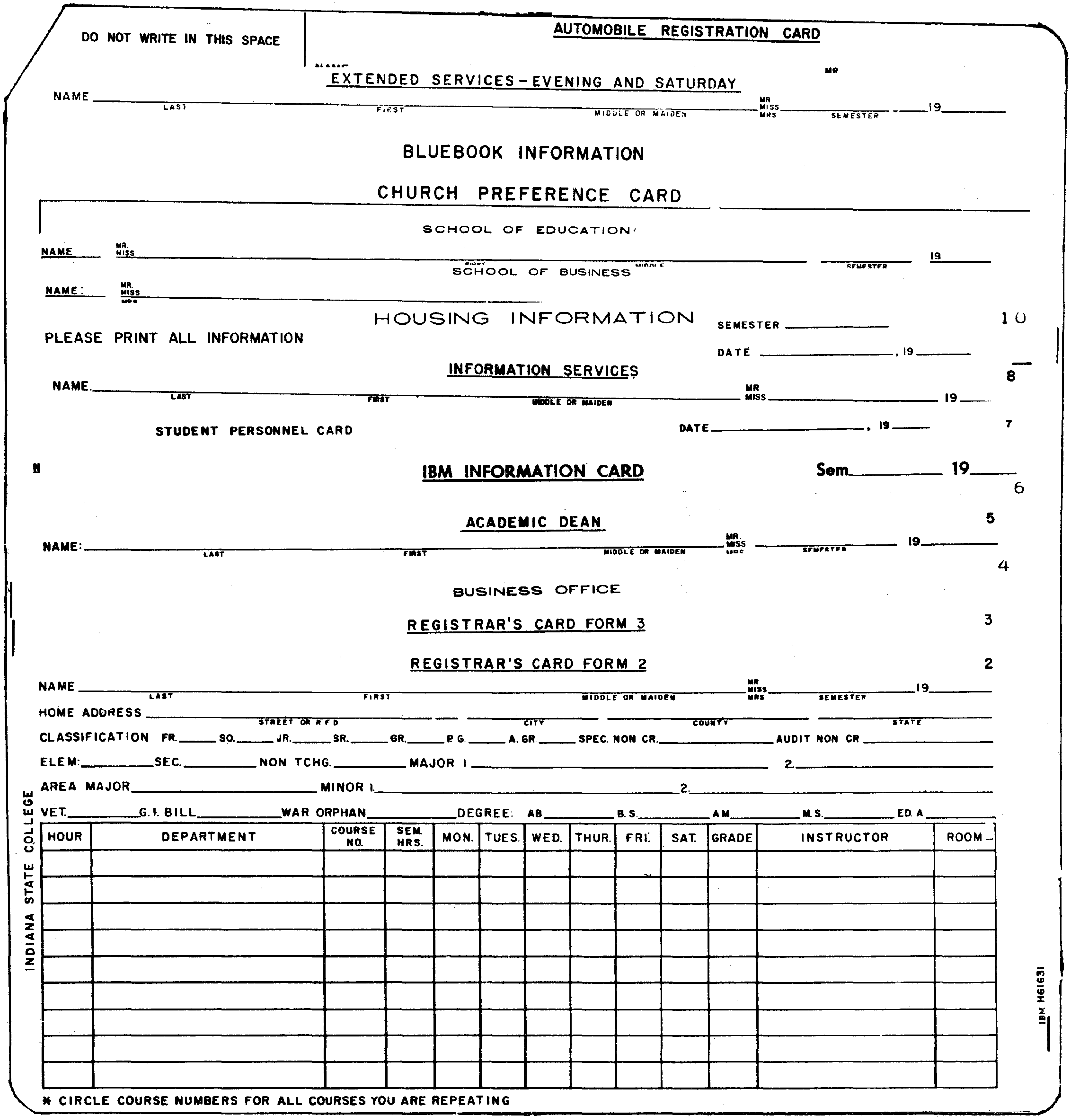
Total Students (on campus
(off campus
(special and auditors - 9099

| | |
|---|---|
| Total Semester Hours | 98238 |
| Equated full time enrollment | 6549.3 |

## SEMESTER HOUR ENROLLMENT BY CLASSIFICATION

| CLASS | No. Students | Semester Hou |
|---|---|---|
| Freshmen | 2499 | 34265 |
| Sophomores | 1539 | 21708 |
| Juniors | 1187 | 17112 |
| Seniors | 1089 | 15013 |
| Graduate | 1032 | 5698 |
| Auditors and Special (non-credit) | 43 | 136 |
| TOTAL | 7389 | 93932 |

EXHIBIT 30



DO NOT WRITE IN THIS SPACE

AUTOMOBILE REGISTRATION CARD

NAME MR

EXTENDED SERVICES - EVENING AND SATURDAY

NAME ______ LAST ______ FIRST ______ MIDDLE OR MAIDEN ______ MR MISS MRS ______ SEMESTER ______ 19______

BLUEBOOK INFORMATION

CHURCH PREFERENCE CARD

SCHOOL OF EDUCATION

NAME MR. MISS ______ FIRST ______ MIDDLE ______ SEMESTER ______ 19______

SCHOOL OF BUSINESS

NAME: MR. MISS MRS ______

HOUSING INFORMATION SEMESTER ______ 10

PLEASE PRINT ALL INFORMATION DATE ______, 19______

INFORMATION SERVICES 8

NAME ______ LAST ______ FIRST ______ MIDDLE OR MAIDEN ______ MR MISS ______ 19______

STUDENT PERSONNEL CARD DATE ______, 19______ 7

N IBM INFORMATION CARD Sem ______ 19______ 6

ACADEMIC DEAN 5

NAME: ______ LAST ______ FIRST ______ MIDDLE OR MAIDEN ______ MR. MISS MRS ______ SEMESTER ______ 19______ 4

BUSINESS OFFICE

REGISTRAR'S CARD FORM 3 3

REGISTRAR'S CARD FORM 2 2

NAME ______ LAST ______ FIRST ______ MIDDLE OR MAIDEN ______ MR MISS MRS ______ SEMESTER ______ 19______

HOME ADDRESS ______ STREET OR R F D ______ CITY ______ COUNTY ______ STATE ______

CLASSIFICATION FR.______ SO.______ JR.______ SR.______ GR.______ P G.______ A. GR______ SPEC. NON CR.______ AUDIT NON CR ______

ELEM:______ SEC.______ NON TCHG.______ MAJOR 1 ______ 2.______

AREA MAJOR ______ MINOR 1.______ 2.______

VET.______ G.I. BILL______ WAR ORPHAN______ DEGREE: AB______ B.S.______ A.M______ M.S.______ ED. A.______

| HOUR | DEPARTMENT | COURSE NO. | SEM. HRS. | MON. | TUES. | WED. | THUR. | FRI. | SAT. | GRADE | INSTRUCTOR | ROOM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

INDIANA STATE COLLEGE

IBM H61631

* CIRCLE COURSE NUMBERS FOR ALL COURSES YOU ARE REPEATING

EXHIBIT-31

E-Z-OUT ® **PLEASE USE A HARD SURFACE TO WRITE ON**

**PLEASE BEAR DOWN**
**YOU ARE MAKING**
**6 COPIES**

02302
**PRINT LEGIBLY**
UARCO BUSINESS FORMS
CHICAGO

# CHANGE IN COURSE CARD
## INDIANA STATE UNIVERSITY

**BUSINESS OFFICE COPY**

LAST NAME ______________________ (PLEASE PRINT) FIRST NAME ______________________ (PLEASE PRINT) DATE ________

| HOUR | COURSE ADDED | COURSE NUMBER | SEMESTER HOURS | INSTRUCTOR'S SIGNATURE | DATE ENTERED CLASS |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| HOUR | COURSE DROPPED | COURSE NUMBER | SEMESTER HOURS | INSTRUCTOR'S SIGNATURE | DATE OF LAST ATTENDANCE |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

APPROVED BY COUNSELOR

★ REGISTRARS OFFICE

| STUDENTS DO NOT WRITE IN THIS AREA | FEES |
|---|---|
| CHANGE OF COURSE FEE | $ |
| ADDITIONAL FEES | $ |
| TOTAL FEES | $ |

BUSINESS OFFICE

DATE FEES PAID OR REFUNDED ________

AMOUNT PAID $________

RECEIPT NO. ________

AMT. REFUNDED $________

PROCESSED BY ________
BUSINESS OFFICE

CHANGE OF COURSE FEE WAIVED–REASON ______________________

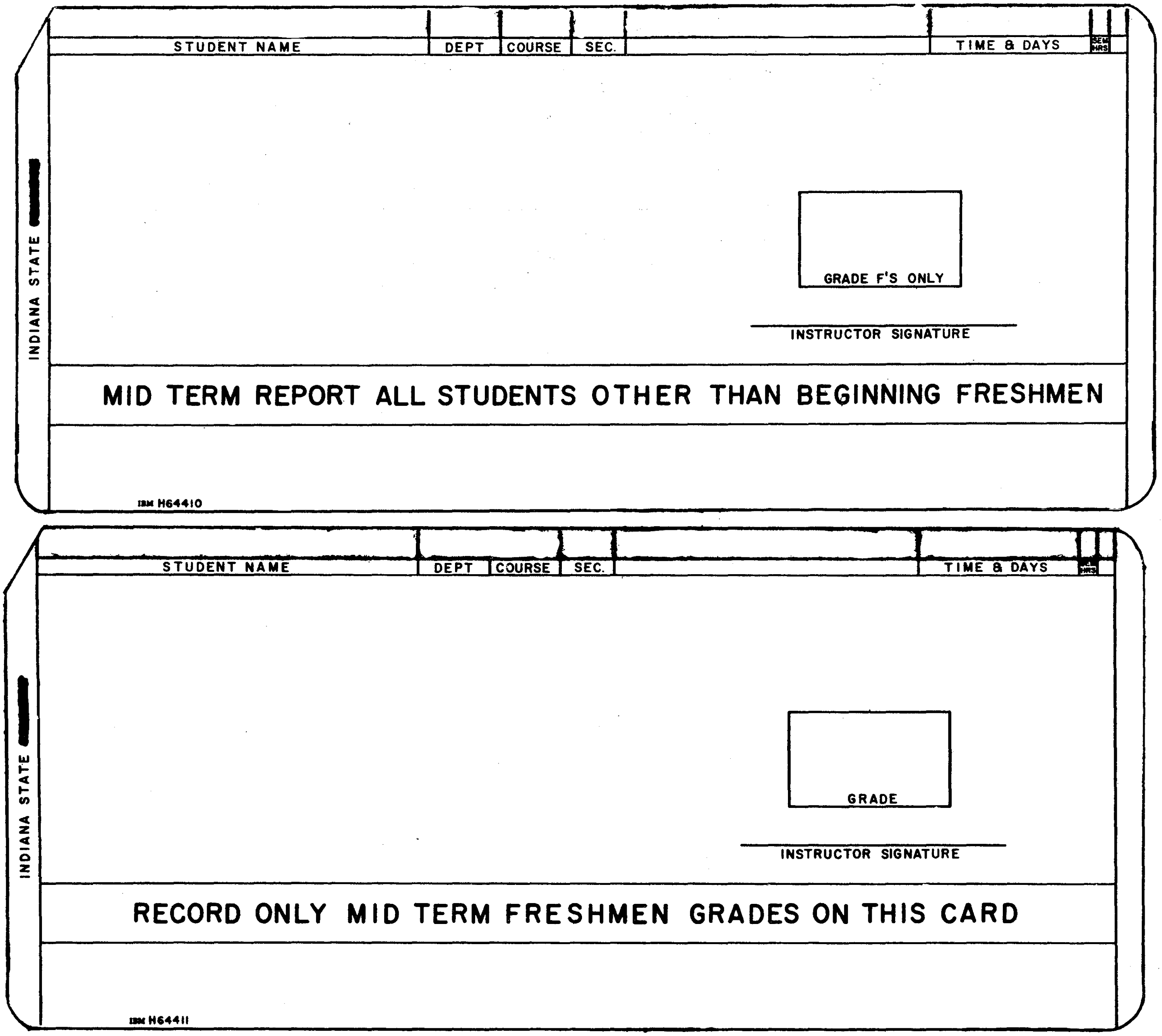
______________________ APPROVED BY ________
REGISTRAR

**DETERMINATION OF FEE OR REFUND WILL BE BASED UPON DATE COMPLETED FORM IS PRESENTED TO BUSINESS OFFICE**

EXHIBIT-32

DATE COMPLETED ________ APPROVED BY ________
REGISTRAR

444



INDIANA STATE [redacted]

| STUDENT NAME | DEPT | COURSE | SEC. | | TIME & DAYS | SEM HRS |
|---|---|---|---|---|---|---|

GRADE F'S ONLY

INSTRUCTOR SIGNATURE

MID TERM REPORT ALL STUDENTS OTHER THAN BEGINNING FRESHMEN

IBM H64410

INDIANA STATE [redacted]

| STUDENT NAME | DEPT | COURSE | SEC. | | TIME & DAYS | SEM HRS |
|---|---|---|---|---|---|---|

GRADE

INSTRUCTOR SIGNATURE

RECORD ONLY MID TERM FRESHMEN GRADES ON THIS CARD

IBM H64411

EXHIBIT-33

VERO BEACH H S

AVERAGE SCHOLARSHIP INDEX OF FRESHMEN FROM YOUR HIGH SCHOOL IN COMPARISON WITH ALL COLLEGE FRESHMEN AT INDIANA STATE UNIVERSITY, FIRST SEMESTER

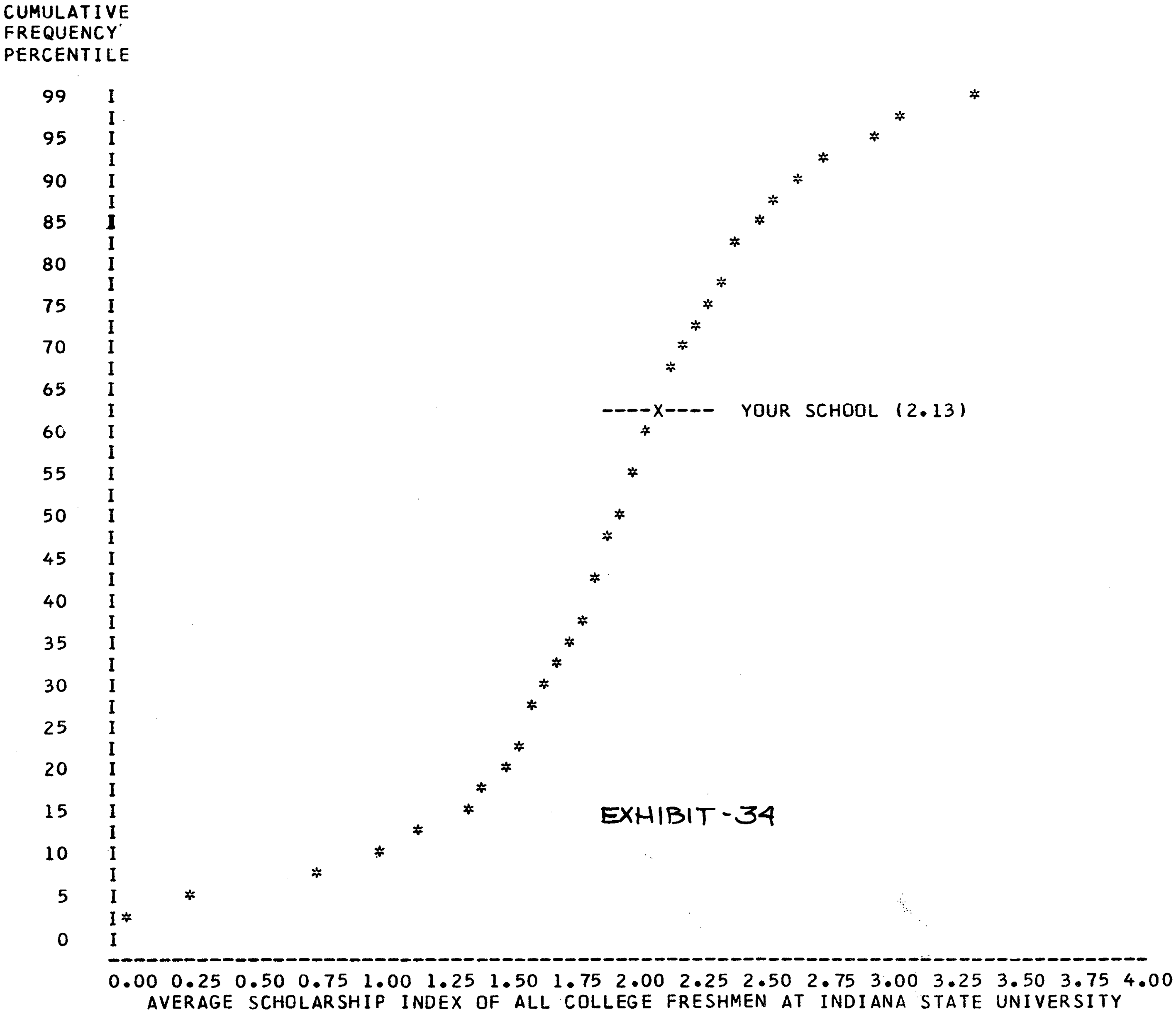


EXHIBIT-34

458

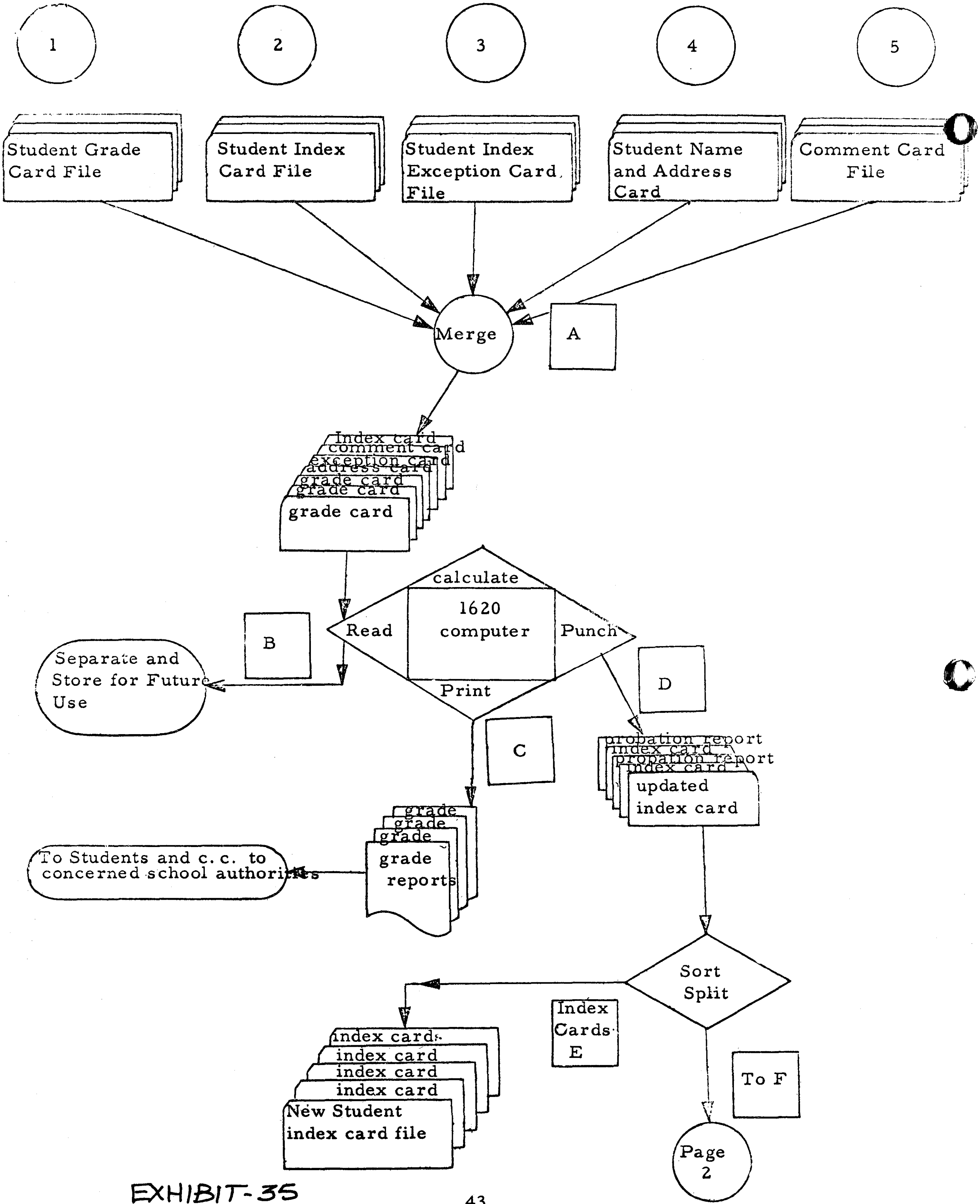

1
2
3
4
5
Student Grade Card File
Student Index Card File
Student Index Exception Card File
Student Name and Address Card
Comment Card File
Merge
A
Index card
comment card
exception card
address card
grade card
grade card
grade card
calculate
Read
1620 computer
Punch
Print
B
Separate and Store for Future Use
D
C
probation report
index card
probation report
index card
updated index card
grade
grade
grade
grade reports
To Students and c. c. to concerned school authorities
Sort Split
Index Cards E
index cards
index card
index card
index card
New Student index card file
To F
Page 2

EXHIBIT-35

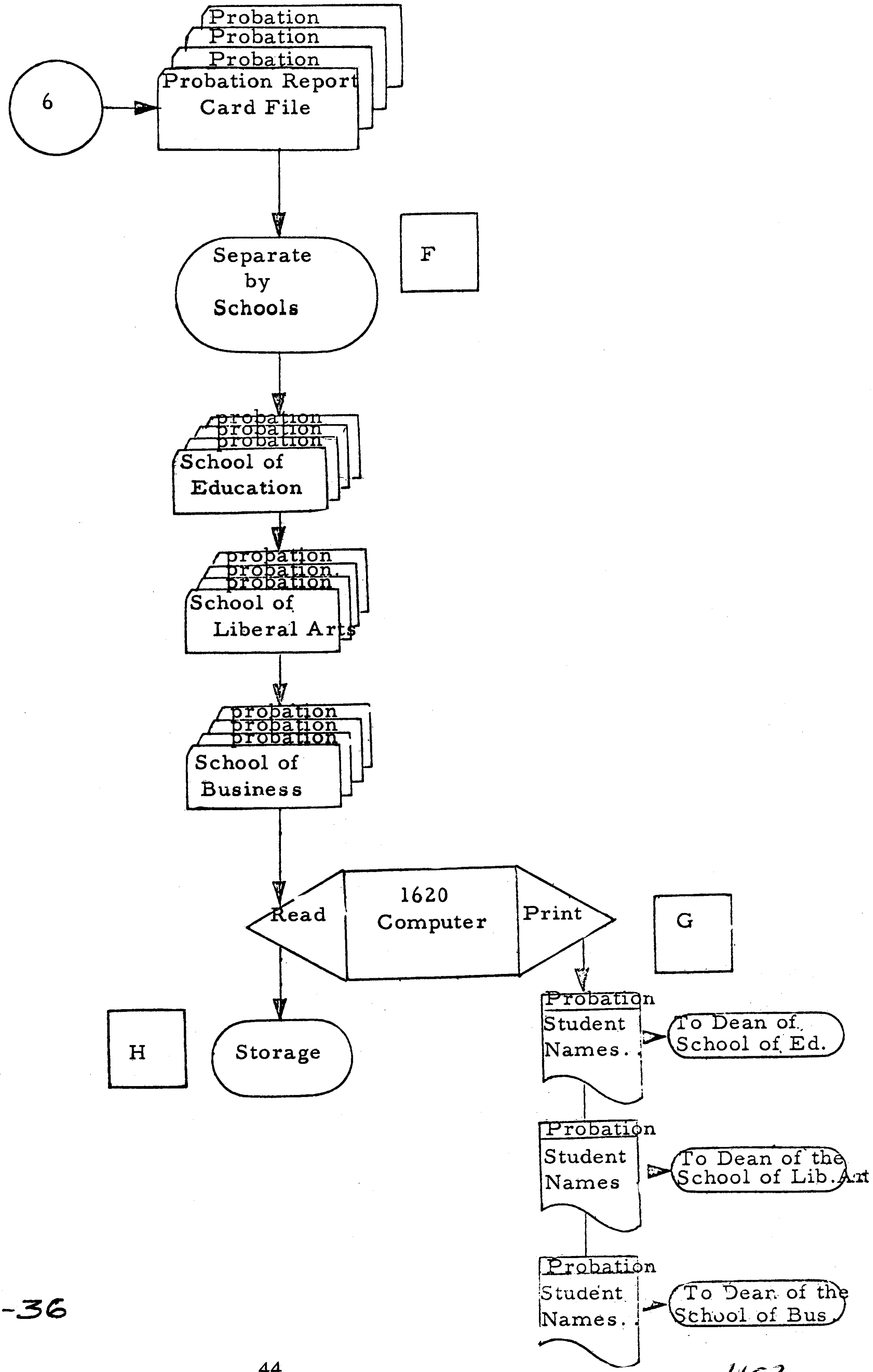

6
Probation
Probation
Probation
Probation Report Card File
Separate by Schools
F
probation
probation
probation
School of Education
probation
probation.
probation
School of Liberal Arts
probation
probation
probation
School of Business
Read
1620 Computer
Print
G
H
Storage
Probation Student Names.
To Dean of School of Ed.
Probation Student Names
To Dean of the School of Lib. Art
Probation Student Names.
To Dean of the School of Bus.

EXHIBIT-36

## BY CLASSIFICATION

### MEN

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| FRESHMEN | 754 | 7,959 | 12,856.0 | 1.615 |
| SOPHOMORES | 1,191 | 16,063 | 33,315.0 | 2.074 |
| JUNIORS | 699 | 9,603 | 23,857.5 | 2.484 |
| SENIORS | 766 | 10,096 | 26,256.5 | 2.601 |
| OTHERS | 223 | 2,472 | 6,930.5 | 2.804 |
| TOTAL | 3,633 | 46,193 | 103,215.5 | 2.234 |

### WOMEN

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| FRESHMEN | 383 | 3,445 | 6,167.0 | 1.790 |
| SOPHOMORES | 878 | 12,593 | 29,146.0 | 2.314 |
| JUNIORS | 581 | 8,181 | 21,498.5 | 2.628 |
| SENIORS | 603 | 7,750 | 22,368.0 | 2.886 |
| OTHERS | 215 | 2,122 | 6,585.5 | 3.103 |
| TOTAL | 2,659 | 34,091 | 85,765.0 | 2.516 |

### TOTAL

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| FRESHMEN | 1,137 | 11,404 | 19,023.0 | 1.668 |
| SOPHOMORES | 2,069 | 28,656 | 62,461.0 | 2.180 |
| JUNIORS | 1,280 | 17,784 | 45,356.0 | 2.550 |
| SENIORS | 1,369 | 17,846 | 48,624.5 | 2.725 |
| OTHERS | 437 | 4,574 | 13,516.0 | 2.955 |
| TOTAL | 6,292 | 80,269 | 188,930.5 | 2.354 |

EXHIBIT- 37-1

E-Z-OUT ® PLEASE USE A HARD SURFACE TO WRITE ON

PLEASE BEAR DOWN YOU ARE MAKING 6 COPIES

02302

PRINT LEGIBLY

UARCO BUSINESS FORMS CHICAGO

# CHANGE IN COURSE CARD
INDIANA STATE UNIVERSITY

BUSINESS OFFICE COPY

LAST NAME ________________ (PLEASE PRINT) FIRST NAME ________________ (PLEASE PRINT) DATE ________

| HOUR | COURSE ADDED | COURSE NUMBER | SEMESTER HOURS | INSTRUCTOR'S SIGNATURE | DATE ENTERED CLASS |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| **HOUR** | **COURSE DROPPED** | **COURSE NUMBER** | **SEMESTER HOURS** | **INSTRUCTOR'S SIGNATURE** | **DATE OF LAST ATTENDANCE** |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

APPROVED BY COUNSELOR

★ REGISTRARS OFFICE

| STUDENTS DO NOT WRITE IN THIS AREA | FEES |
|---|---|
| CHANGE OF COURSE FEE | $ |
| ADDITIONAL FEES | $ |
| TOTAL FEES | $ |

BUSINESS OFFICE

DATE FEES PAID OR REFUNDED ________

AMOUNT PAID $________

RECEIPT NO. ________

AMT. REFUNDED $________

PROCESSED BY ________ BUSINESS OFFICE

CHANGE OF COURSE FEE WAIVED–REASON ________________

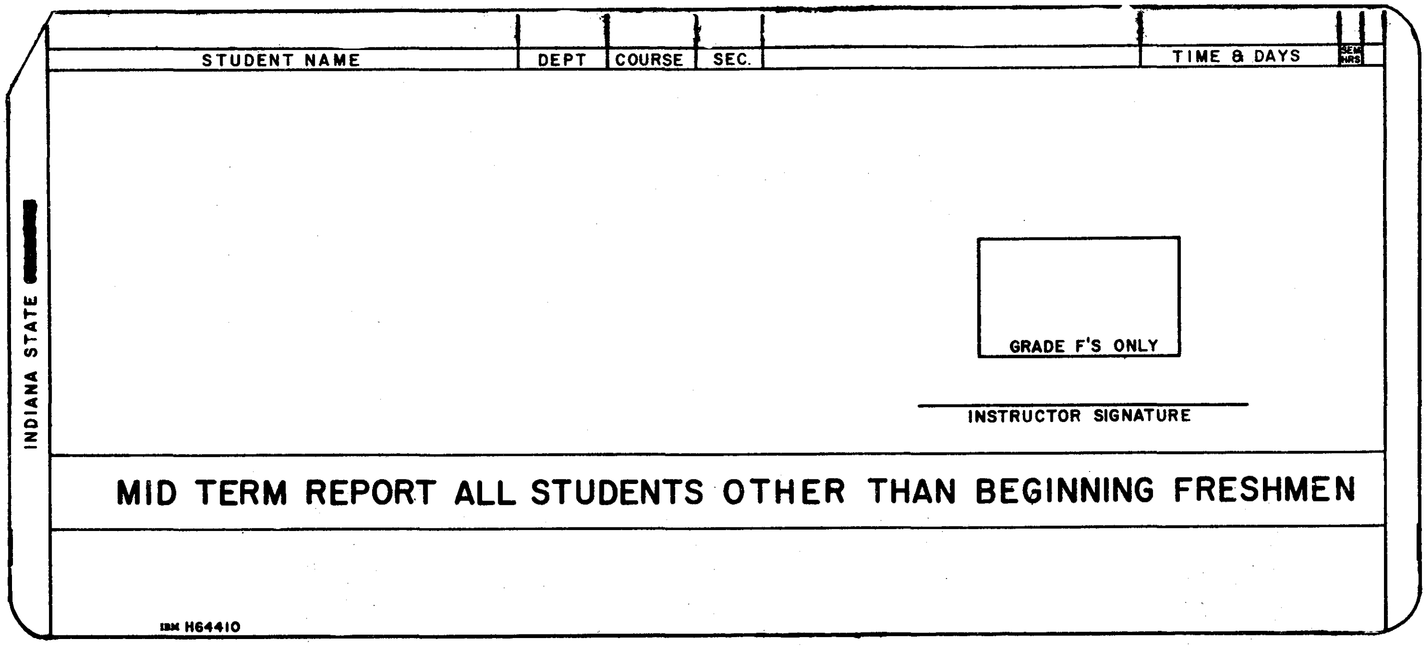
________________ APPROVED BY ________ REGISTRAR

DETERMINATION OF FEE OR REFUND WILL BE BASED UPON DATE COMPLETED FORM IS PRESENTED TO BUSINESS OFFICE

EXHIBIT-32

DATE COMPLETED ________ APPROVED BY ________ REGISTRAR

474



INDIANA STATE

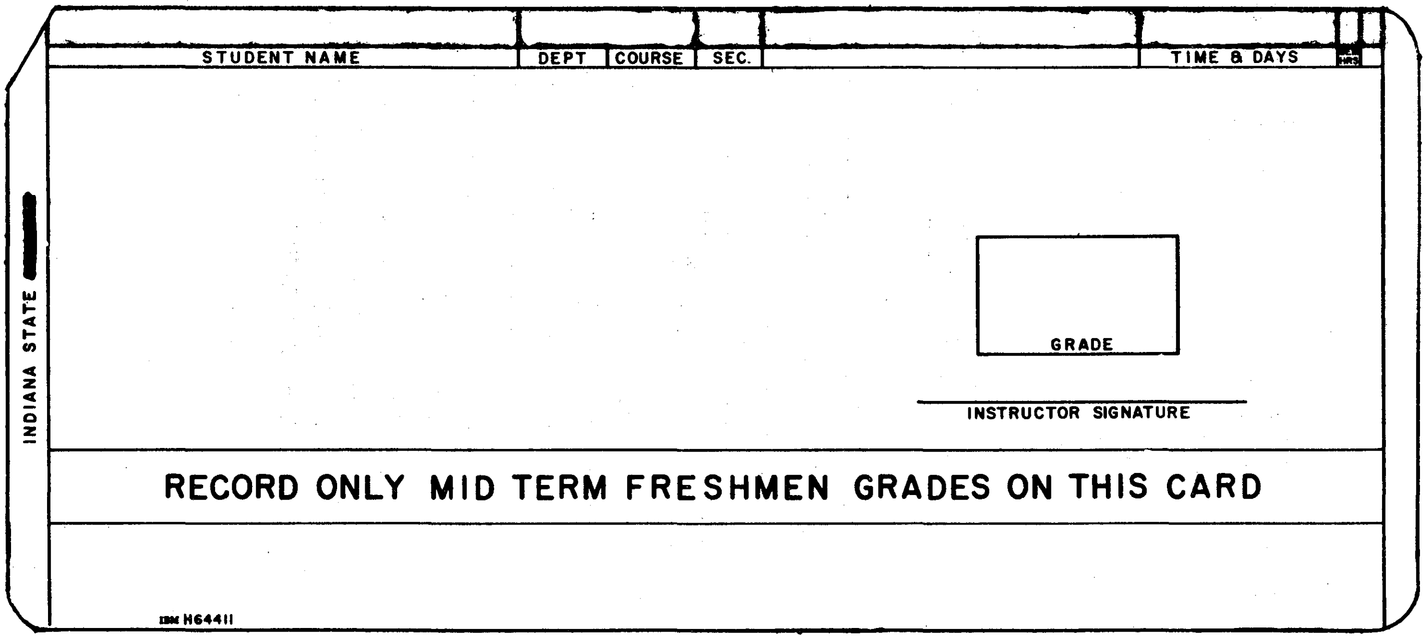
| | | | | |
|---|---|---|---|---|
| STUDENT NAME | DEPT | COURSE | SEC. | TIME & DAYS | SEM HRS |

GRADE F'S ONLY

INSTRUCTOR SIGNATURE

MID TERM REPORT ALL STUDENTS OTHER THAN BEGINNING FRESHMEN

IBM H64410



INDIANA STATE

| | | | | | |
|---|---|---|---|---|---|
| STUDENT NAME | DEPT | COURSE | SEC. | TIME & DAYS | SEM HRS |

GRADE

INSTRUCTOR SIGNATURE

RECORD ONLY MID TERM FRESHMEN GRADES ON THIS CARD

IBM H64411

EXHIBIT-33

## FRATERNITIES

### PLEDGES

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| PI LAMBDA PHI | 0 | 0 | .0 | .000 |
| THETA CHI | 0 | 0 | .0 | .000 |
| ALPHA TAU OMEGA | 0 | 0 | .0 | .000 |
| TAU KAPPA EPSILON | 0 | 0 | .0 | .000 |
| SIGMA PHI EPSILON | 0 | 0 | .0 | .000 |
| LAMBDA CHI ALPHA | 0 | 0 | .0 | .000 |
| ALL FRATERNITY MEN | 0 | 0 | .0 | .000 |

EXHIBIT-37-4

# FRATERNITIES

## COMBINED

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| PI LAMBDA PHI | 34 | 447 | 1,157.5 | 2.589 |
| THETA CHI | 52 | 676 | 1,661.5 | 2.458 |
| ALPHA TAU OMEGA | 107 | 1,431 | 3,474.5 | 2.428 |
| TAU KAPPA EPSILON | 96 | 1,303 | 3,159.0 | 2.424 |
| SIGMA PHI EPSILON | 96 | 1,283 | 3,105.0 | 2.420 |
| LAMBDA CHI ALPHA | 111 | 1,526 | 3,487.5 | 2.285 |
| ALL FRATERNITY MEN | 496 | 6,666 | 16,042.0 | 2.407 |
| NON FRATERNITY | 3,137 | 39,528 | 87,179.5 | 2.206 |

EXHIBIT-37-5

## SORORITIES

### ACTIVES

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| ZETA TAU ALPHA | 62 | 887 | 2,503.0 | 2.822 |
| DELTA GAMMA | 62 | 845 | 2,384.0 | 2.821 |
| GAMMA PHI BETA | 62 | 902 | 2,528.0 | 2.803 |
| SIGMA KAPPA | 61 | 837 | 2,307.0 | 2.756 |
| ALPHA PHI | 59 | 829 | 2,275.5 | 2.745 |
| CHI OMEGA | 68 | 923 | 2,516.5 | 2.726 |
| ALPHA OMICRON PI | 63 | 860 | 2,340.0 | 2.721 |
| ALPHA SIGMA ALPHA | 51 | 738 | 1,987.5 | 2.693 |
| ALL SORORITY WOMEN | 488 | 6,821 | 18,841.5 | 2.762 |

EXHIBIT-37-6

## SORORITIES

### PLEDGES

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| ZETA TAU ALPHA | 0 | 0 | .0 | .000 |
| GAMMA PHI BETA | 0 | 0 | .0 | .000 |
| DELTA GAMMA | 0 | 0 | .0 | .000 |
| CHI OMEGA | 0 | 0 | .0 | .000 |
| ALPHA PHI | 0 | 0 | .0 | .000 |
| SIGMA KAPPA | 0 | 0 | .0 | .000 |
| ALPHA SIGMA ALPHA | 0 | 0 | .0 | .000 |
| ALPHA OMICRON PI | 0 | 0 | .0 | .000 |
| ALL SORORITY WOMEN | 0 | 0 | .0 | .000 |

EXHIBIT - 37-7

## SORORITIES

### COMBINED

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| ZETA TAU ALPHA | 62 | 887 | 2,503.0 | 2.822 |
| DELTA GAMMA | 62 | 845 | 2,384.0 | 2.821 |
| GAMMA PHI BETA | 62 | 902 | 2,528.0 | 2.803 |
| SIGMA KAPPA | 61 | 837 | 2,307.0 | 2.756 |
| ALPHA PHI | 59 | 829 | 2,275.5 | 2.745 |
| CHI OMEGA | 68 | 923 | 2,516.5 | 2.726 |
| ALPHA OMICRON PI | 63 | 860 | 2,340.0 | 2.721 |
| ALPHA SIGMA ALPHA | 51 | 738 | 1,987.5 | 2.693 |
| ALL SORORITY WOMEN | 488 | 6,821 | 18,841.5 | 2.762 |
| NON SORORITY | 2,171 | 27,270 | 66,925.5 | 2.454 |

EXHIBIT-37-8

## RESIDENCE HALL SUMMARY

| | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| ERICKSON | 264 | 3,897 | 9,766.5 | 2.506 |
| BURFORD | 252 | 3,629 | 9,081.5 | 2.502 |
| PICKERL | 271 | 3,852 | 9,561.0 | 2.482 |
| REEVE | 282 | 4,120 | 9,499.0 | 2.306 |
| BLUMBERG | 362 | 5,351 | 12,188.5 | 2.278 |
| SANDISON | 286 | 4,055 | 9,101.5 | 2.245 |
| GILLUM | 287 | 4,035 | 8,990.5 | 2.228 |
| HULMAN | 257 | 3,572 | 7,819.0 | 2.189 |
| CROMWELL | 383 | 5,313 | 11,103.0 | 2.090 |
| PARSONS | 247 | 3,496 | 7,282.5 | 2.083 |
| TOTAL | 2,891 | 41,320 | 9,439.3 | .228 |
| OTHER | 3,401 | 38,949 | 94,537.5 | 2.427 |
| OTHER (EXCLUDING FRATERNITIES AND SORORITIES) | 2,417 | 25,462 | 59,654.0 | 2.343 |

EXHIBIT-37-9

## BY RESIDENCE HALL

### ERICKSON HALL

| HOUSE | NUMBER OF STUDENTS | CREDIT HOURS ATTEMPTED | GRADE POINTS EARNED | GRADE-POINT RATIO |
|---|---|---|---|---|
| 2 | 54 | 780 | 1,986.5 | 2.547 |
| 3 | 47 | 692 | 1,715.0 | 2.478 |
| 4 | 55 | 812 | 1,942.5 | 2.392 |
| 5 | 54 | 806 | 2,030.0 | 2.519 |
| 6 | 54 | 807 | 2,092.5 | 2.593 |
| TOTAL | 264 | 3,897 | 9,766.5 | 2.506 |

EXHIBIT-37-10

| COURSE | | CREDIT | | |
|---|---|---|---|---|
| DEPT. | NUMBER | HOURS | POINTS | GRADE |
| | | | | |

# GRADE REPORT

INDIANA STATE UNIVERSITY
TERRE HAUTE, INDIANA

| STATUS | HOURS | POINTS | RATIO |
|---|---|---|---|
| THIS SEMESTER | | | |
| CUMULATIVE | | | |

| COMMENT | |
|---|---|
| Type | By |
| | |

| MA-JOR |
|---|
| |

| TRANSFER HOURS | |
|---|---|
| Attempted | Accepted |
| | |

COMMENTS:

EXHIBIT-38

James H. Hinger
REGISTRAR

Date

INDIANA STATE UNIVERSITY
STUDENT BREAK-DOWN BY COUNTY
8/24/65

EXHIBIT-39

| COUNTY | ADV. GRADUATE | | FRESHMAN | | SOPHOMORE | | JUNIOR | | SENIOR | | GRADUATE | | POST GRAD | | SP. STUDENTS | | AUDITOR | | TOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | |
| NOT IN IND | 0 | 0 | 97 | 57 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 154 |
| ADAMS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALLEN | 0 | 0 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| BARTHOLOMEW | 0 | 0 | 12 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 |
| BENTON | 0 | 0 | 3 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 |
| BLACKFORD | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| BOONE | 0 | 0 | 12 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 |
| BROWN | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| CARROLL | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 |
| CASS | 0 | 0 | 20 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 |
| CLARK | 0 | 0 | 7 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 |
| CLAY | 0 | 0 | 44 | 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| CLINTON | 0 | 0 | 1 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| CRAWFORD | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| DAVIES | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 |
| [illegible] | 0 | 0 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| DECATUR | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| DEKALB | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| DELAWARE | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| DUBOIS | 0 | 0 | 9 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 |
| ELKHART | 0 | 0 | 12 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 |
| FAYETTE | 0 | 0 | 11 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 |
| FLOYD | 0 | 0 | 7 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| FOUNTAIN | 0 | 0 | 13 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 |
| FRANKLIN | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

465

INDIANA STATE UNIVERSITY
STUDENT BREAK-DOWN BY CLASSIFICATION
8/24/65

EXHIBIT - 39

| STATE | ADV GRADUATE |  | FRESHMAN |  | SOPHOMORE |  | JUNIOR |  | SENIOR |  | GRADUATE |  | POST GRAD |  | SP. STUDENTS |  | AUDITOR |  | TOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN | MEN | WOMEN |  |
| ALABAMA | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ARIZONA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ARKANSAS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CAL. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| COLORADO | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CONN. | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| DELAWARE | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| WASH. D.C. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| FLORIDA | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| GEORGIA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IDAHO | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ILLINOIS | 0 | 0 | 37 | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 67 |
| INDIANA | 0 | 0 | 1233 | 933 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2166 |
| IOWA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| KANSAS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| KENTUCKY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| LOUISIANA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MAINE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MARYLAND | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| MASS. | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| MICHIGAN | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| MINNESOTA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MISS. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MISSOURI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MONTANA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

466

INDIA STATE UNIVERSITY
8/30/65

COMPUTER CENTER -- STUDENT INDEX SPRING 1965

| | STUDENT NAME | SEMESTER INDEX | | | CUMULATIVE INDEX | | | TOTAL | |
|---|---|---|---|---|---|---|---|---|---|
| | | HRS | PTS | CPR | HRS | PTS | CPR | HRS | |
| 01110 | ALLEN JOHN PATRICK | 17 | 62.5 | 3.68 | 124.0 | 363.3 | 2.93 | 124.0 | SEM HNOR |
| 01675 | ANDERSON WM DAVID | 15 | 38.5 | 2.57 | 029.0 | 071.0 | 2.45 | 029.0 | |
| 01760 | ANDREWS RICHARD F | 17 | 44.5 | 2.62 | 062.0 | 153.5 | 2.48 | 092.0 | |
| 02705 | AVE MARK ANTHONY | 15 | 33.0 | 2.20 | 030.0 | 073.0 | 2.43 | 030.0 | |
| 04310 | BAUERMEISTER JOHN F | 18 | 42.0 | 2.33 | 117.0 | 277.5 | 2.37 | 107.0 | |
| 04315 | BAUERMEISTER STEPHEN | 12 | 28.0 | 2.33 | 030.0 | 071.5 | 2.38 | 030.0 | |
| 04360 | BAUSMAN GORDON PARKS | 17 | 52.0 | 3.06 | 096.0 | 265.0 | 2.76 | 093.0 | |
| 06155 | BEST ROBERT WAYNE | 16 | 53.0 | 3.31 | 032.0 | 112.5 | 3.52 | 032.0 | CUM HNOR |
| 06240 | BEVINGTON JOHN M | 17 | 58.0 | 3.41 | 097.0 | 278.0 | 2.87 | 094.0 | |
| 06250 | BIANCHETTA JAMES E | 07 | 22.5 | 3.21 | 119.0 | 298.0 | 2.50 | 125.0 | |
| 07800 | BONNESS RICHARD R | 16 | 39.5 | 2.47 | 061.0 | 150.0 | 2.46 | 061.0 | |
| 08615 | BOZELL THOMAS RICKY | 15 | 29.0 | 1.93 | 029.0 | 071.0 | 2.45 | 029.0 | |
| 09990 | BROWDER RICHARD ALLEN | 16 | 44.0 | 2.75 | 125.0 | 291.0 | 2.33 | 125.0 | |
| 10130 | BROWN DOUGLAS KAY | 15 | 40.0 | 2.67 | 094.0 | 223.5 | 2.38 | 120.0 | |
| 10250 | BROWN JERRY EARL | 09 | 27.5 | 3.06 | 115.0 | 326.5 | 2.84 | 121.0 | |
| 10730 | BRUGH JOSEPH R | 16 | 47.0 | 2.94 | 065.0 | 175.0 | 2.69 | 065.0 | |
| 11630 | BURNS LARRY STEVEN | 14 | 45.0 | 3.21 | 099.0 | 254.0 | 2.57 | 099.0 | |
| 11950 | BUSH FRANK ANTHONY JR | 10 | 30.5 | 3.05 | 117.0 | 273.0 | 2.33 | 124.0 | |
| 13430 | CARROLL RAYMOND LEE | 16 | 39.5 | 2.47 | 090.0 | 208.5 | 2.32 | 087.0 | |

EXHIBIT-4C

58

467

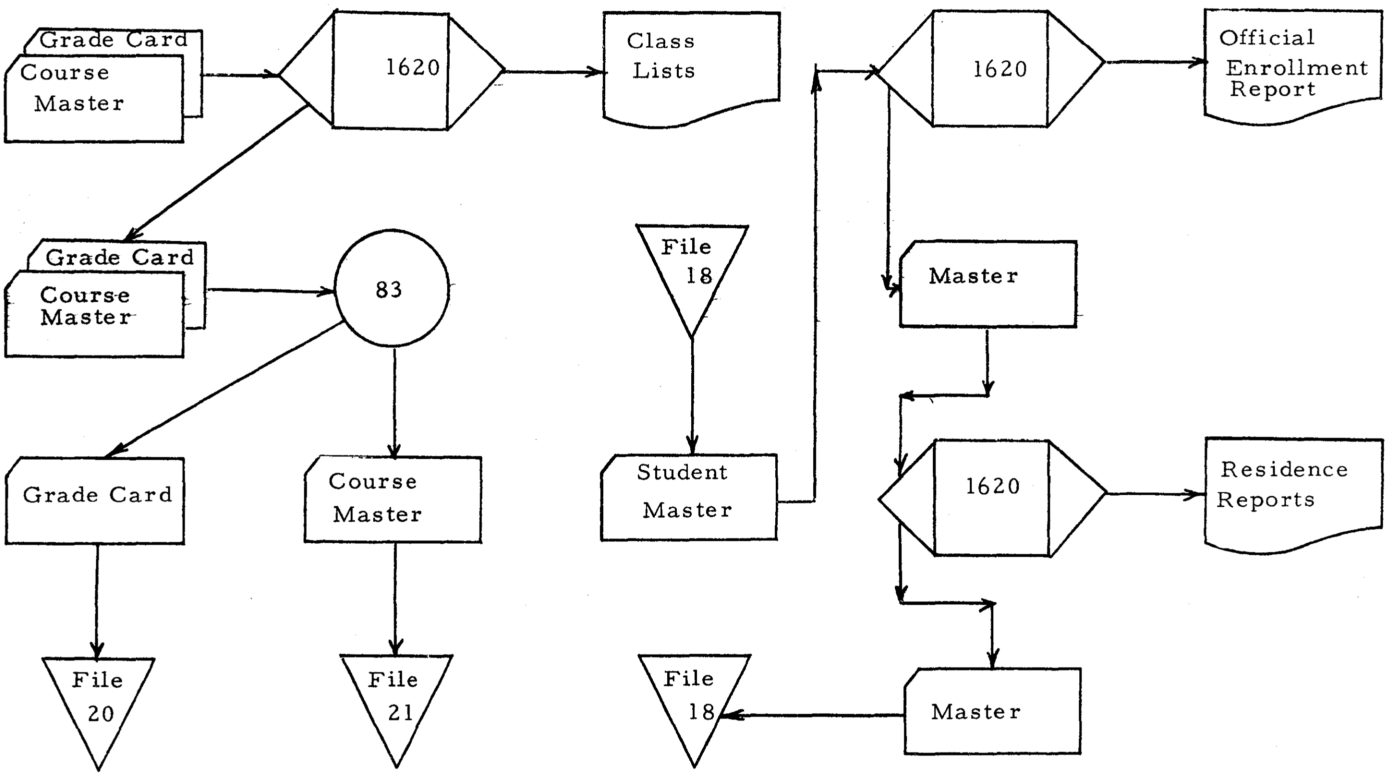

Grade Card
Course
Master
1620
Class
Lists
1620
Official
Enrollment
Report
Grade Card
Course
Master
83
File
18
Master
Grade Card
Course
Master
Student
Master
1620
Residence
Reports
File
20
File
21
File
18
Master

Exhibit-41

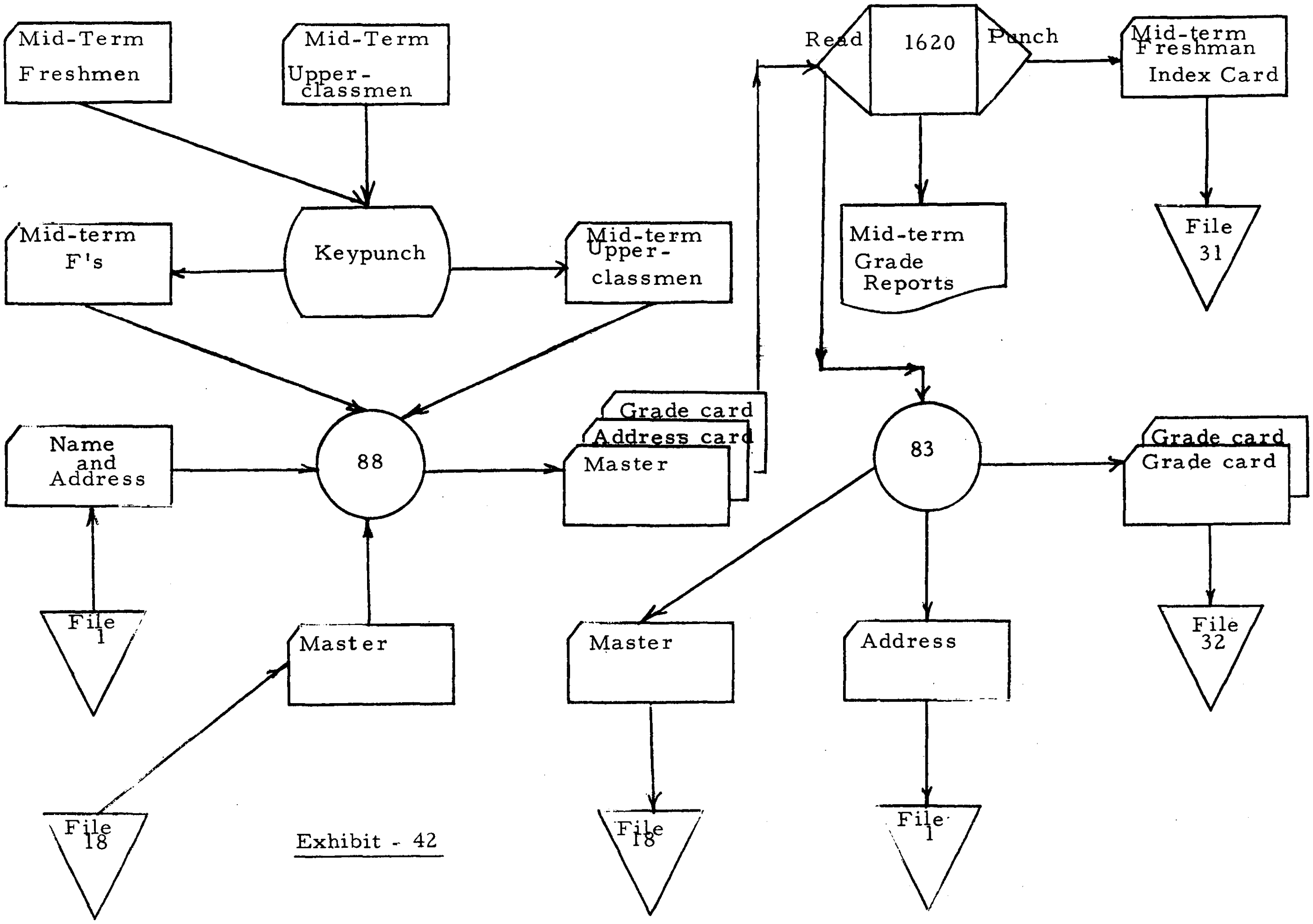

Mid-Term Freshmen
Mid-Term Upper-classmen
Keypunch
Mid-term F's
Mid-term Upper-classmen
Name and Address
88
File 1
Master
File 18
Grade card
Address card
Master
Read
1620
Punch
Mid-term Freshman Index Card
File 31
Mid-term Grade Reports
83
Grade card
Grade card
File 32
Master
File 18
Address
File 1

Exhibit - 42

# ISU COMPUTER CENTER EQUIPMENT

| | |
|---|---|
| 1-Mod-1 | 1620, 20 K, IA, Auto-Divide, Edit Instructions |
| 1-Mod-3 | 1311 Disk Drive |
| 1-Mod-2 | 1443 Printer-144 Print Positions |
| 1-Mod-2 | 1622 Card Read/Punch 500/250 w/File Feed |
| 6-026 | Keypunches |
| 1 | Verifier |
| 1-082 | Sorter with Counters |
| 1-083 | Sorter |
| 1-077 | Collator with Internal Counters |
| 1-088 | Collator with Internal Counters |
| 1-557 | Interpreter |
| 1-403 | Accounting Machine |
| 1-514 | Reproducer |
| 1 | Decollator |
| 1 | Burster with Slitters and Imprinter |
| 10 | Disk Packs |

(Exhibit 43)

# NOTES

Binders courtesy of: H. M. Oshry Co. Inc.
Crawfordsville, Indiana

WILKES-BARRE CITY SCHOOLS

COMPUTER CENTER

# THE NORTH CAROLINA SUMMER TEACHERS

# DATA PROCESSING INSTITUTE

Presented at
1620 USERS GROUP MEETING
NEW YORK, N. Y.
October 8, 1965

A. David Mayer
Coordinator, Data Processing
Wilkes-Barre City Schools

472

# THE NORTH CAROLINA SUMMER TEACHERS

# DATA PROCESSING INSTITUTE

The Wall Street Journal for September 21, 1965 carried an article describing the shortage of computer programmers in this country. According to this article there are about 100,000 programmers employed at the present time and an additional 25,000 are currently needed to efficiently handle the nation's 23,000 computers. In the past many programmers have been trained on the job, but the history of this country shows that the apprenticeship system has disappeared in most work areas and schools take over with more efficient training methods. This challenge to the schools of the nation has been handicapped by a lack of teachers capable of Data Processing instruction. To eleviate this shortage of trained teachers the Department of Health, Education, and Welfare of the United States Office of Education, together with various state Education Departments, in 1963 organized a series of Summer Teacher Institutes in Data Processing. These Institutes have been held in California, Wisconsin, Colorada, Florida, and North Carolina.

The 1963 Institute in North Carolina was held at Charlotte. Ashville, for the first year class, and Burlington, for the second year class, were the locations of the 1964 Institutes. During the summer of 1965 the first and second year classes were both held at Raleigh. Successful completion of the institute courses may entitle the participant to degree or non-degree credit depending upon the wishes and status of the participants. The course content is designed specifically to meet the needs of participants in the subject areas of Introduction to Business Data Processing, Electric Accounting Machines, Data Processing Applications, Computer Programming I (first year courses), and Computer Programming II, Programming Systems, Business Systems Design and Development, Advanced Programming Systems, Data Processing Field Project (second year course). A seminar in

Philosophy and Principles of Technical Education was also included. The reader who is interested can get an outline of these courses in the booklet ELECTRONIC DATA PROCESSING-I, U. S. Government Printing Office. The courses are of college level.

The major purpose of the institute is to assist in the development of knowledge and skill essential for teaching specialized courses in preparatory curriculums in business electronic data processing under Federally-supported Vocational and Technical Education Acts. One of the purposes of this program is to determine if successful teachers in the field of Office or Business Education can be retrained through a series of specially designed summer courses to teach in a field having some relationship to their previous experience. There were 33 participants in the Raleigh classes, from seven states.

The criteria for selection of applicants included the following:

1. Bachelor degree in Business Education, Mathematics, or equivalent, preferably with one year of study in accounting.
2. At least three years of teaching experience or combination of teaching and work experience in the field of business or data processing, preferably in accounting, administration, mathematics, or business law.
3. Currently employed as a teacher or department chairman with teaching responsibilities in the field of business or data processing.
4. Available for a teaching assignment and capable of qualifying, at the completion of the institute, under the State plan as a teacher of business data processing in a curriculum designed to prepare computer programmers and application analysts.
5. Recommended for enrollment by State Director of Vocational Education in sending State.

Computer programming comprised about one half of the time in the first-year course at Raleigh. There were 2 20k IBM 1620 model I's in the school available to the first-year class. They were available during and outside of class time from 8 a.m. to 10 p.m. Monday thru Friday and by request on Saturday. The 1620 is an excellent computer for instructional purposes. The first day of the institute the students were given hands-on instruction. By the third day members of the first-year class were writing and running simple programs themselves. The teacher-students made excellent progress in moving up the scale toward more complex problems and by the end of the fifth week each had written an involved payroll program with many decisions, deductions, and computations in machine language. This good background in machine language made possible very rapid progress during the last three weeks of the institute in SPS. Again the students started with simple problems, but each day or two saw as much progress as had been made in a week in machine language. In the three weeks of SPS programming the students wrote many of the same programs which they had done in machine language for a comparison of time and method, and in addition they wrote other programs including a complicated program for checking student test scores and selecting from all the scores certain ones fitting the requirements of the problem.

The members of this class are teachers and machine language is taught first so that a thorough understanding is achieved of the way the computer works. Many people hold to the philosophy that a problem-oriented language is quicker and easier for beginners to learn and we have no disagreement with this philosophy. But this writer holds that teachers must know machine language for a thoro understanding of the work that they will be called upon to teach. They also need to thoroughly understand machine language for debugging purposes in assembler and compiler type languages. A teacher who understands machine language has a big advantage over one who does not.

The 1620 is one of the most common computers in the nation's schools and the teachers from this class were returning to teach on the same computer in their home schools in most cases. Several teachers in the class were returning to different makes or models and so some time was spent in the class discussing and explaining variations in techniques needed for different computers. Field trips also helped in this area.

The second year class was planned primarily for those who had completed the first year program. Fortran and SPS were the languages used in the second year class, with greater emphasis on Fortran. This class also used the two 1620's in the building where classes were held and in addition had the use of a 1620 disk pack off campus. This made possible a major part of the second year work in Fortran II D.

William A. Gannon, writing in COMPUTER DESIGN for April, 1964, states "Perhaps another potential danger is the tendency to rush immediatly to the more glamorous lessons involving programming and machine operation and thereby allotting insufficient time to fundamentals. Reducing problems to the problem-solving framework of a specific computer and actually executing student-prepared programs is an obvious goal of any computer course. However, this goal should be approached from the right direction in a well-balanced curriculum. The successful development of modern computing and information processing systems is largely attributed to the extensive application of digital logic, Boolean algebra, and binary arithmetic. Consequently, these concepts and the techniques used in applying them are recognized as vital areas of knowledge. While the design of computer equipment and programming techniques are in a constant state of evolution, these basic concepts remain essentially unchanged." This philosophy also seems pointed up by the previously mentioned publication of the United States Office of Education SUGGESTED 2-YEAR POST HIGH SCHOOL CURRICULUM FOR COMPUTER PROGRAMMERS AND BUSINESS APPLICATION ANALYSTS. We cannot train our students in a narrow field applicable to only one machine or one situation. We must train people in fundamentals which will apply in many situations. No one can predict accurately the course computer techniques will take in the future and so the courses must be flexibile. Instruction has been given on a 1620 because it is available and because it is presently the most common computer in our schools. But this computer can be and is being used for simulation of other machines and as a device to teach basic programming techniques. An excellent simulation example is to be found in the book by Swallow and Price and in the 141 programming instruction. Our schools should develop other similar techniques to make the most of the available equipment and to help overcome the deficiency in trained manpower which now exists in so many data processing applications.

Most of the teacher-students at Raleigh will return to teach in post-high school situations, many of them in technical schools or community colleges. Several members of the class expected to be teaching in secondary schools. This is another problem which needs further exploration. How much data processing instruction can or should be given in high school? Certainly for those students who expect to enter engineering or similar schools we can easily give them a good start in Fortran programming in high school. And the 1620 is an excellent tool for this type of instruction. But those students who do not expect to go on to college already have enough to learn without omitting some studies to insert courses in programming. Perhaps the future will see a pattern of unit record instruction in high school, leaving the major programming instruction to more mature post-high school minds. Business does not seem anxious to hire our younger graduates. Very few businessmen want to trust the programming of their business to a 17 or 18 year old.

And too many colleges and universities seem to be adding computer programming only as a part of some other course in the curriculum. Certainly it is best for our students to take a 4-year degree, but many cannot or do not want to. There are very few colleges or universities where he can specialize in data processing. In too many cases he has to learn to be an engineer in order to learn to program. The writer feels that the best place to train programmers is in a 2-year post-high school course, preferably one offering an associate degree. The U. S. Office of Education has given us a good guide for a 2-year course. Let's follow it.

The following is a list of text material used at Raleigh this summer.

First year course:

IBM Manuals

85 and 87 Collator

513-514 Reproducing punch

82-83-84 Sorter

24-26 Card Punch

402-403-419 Accounting Machine

Flow Charting Template x20-8020

Flow Charting Techniques C20-8152

IBM 1620 CPU, Model I

BASIC PROGRAMMING CONCEPTS

Leeson and Dimitry

ELECTRONIC BUSINESS DATA PROCESSING

Schmidt and Meyers

Second year course:

IBM 1620 Monitor I System Reference Manual

INTRODUCTION TO BOOLEAN ALGEBRA AND LOGIC DESIGN

Hoernes & Heilweil

LEANEAR PROGRAMMING

Loomba

Modern Business Statistics

Freund and Williams

In conclusion, this paper has tried to show the valuable place the 1620 holds in instructing business programmers in this country. Perhaps it seems a little strange that a machine originally built as a scientific type of computer is now being used so much in training business programmers. But our technical school and college graduates are finding, and will probably continue to find, the greatest number of positions in business rather than in other fields. The schools are wise in looking at where their graduates go, and preparing them to go there. Business teachers of the country are adapting to the age of automation and the new computers just as they did when Mr. Sholes invented that modern machine called the typewriter or when Mr. Burroughs started selling that fancy adding machine. They learned to replace the straight pen with a fountain pen, the fountain pen with a bookkeeping machine, and now they are learning to replace the ledger page with a punched card or a reel of tape. The bread we eat today is made of the same kind of wheat as the 5 loaves which were divided to feed the 5000 in that desert.place. Only now one farmer with his machines can raise more wheat than was ever thought possible when the ground was tilled with human or animal power. The principles of accounting are about the same today as they were in 1492 when Friar Luciola wrote the first known account of double entry bookkeeping. Only the tools have changed. Man always seems to better himself when he betters his tools. And he has found that the only thing more expensive than education is ignorance.

# A COMPUTER-AIDED MECHANICAL LINKAGE DESIGN ANALYSIS SYSTEM

by
D. N. Frayne and H. H. Hansen

An Abstract only is presented here as the material has already appeared in two publications listed below.

A significant portion of mechanical engineering effort is spent in the kinematic analysis of mechanisms such as gears, cams, and linkages. Although linkages present a more complex problem of analysis than most other basic mechanisms, they are widely used because of their reliability, speed, and force-transmission properties. Engineers continually seek improvements in existing linkages and devise linkages for new mechanical systems. Linkage analyses have traditionally been performed on the drafting board, but this is difficult and time consuming, and complete analyses are not feasible for the more involved linkage systems encountered in practice.

This paper describes an experimental tool for the analysis of proposed two or three-dimensional linkages. Called KAM (Kinematic Analysis Method), the tool consists of a programmed system for the IBM 1620. Based on vector mathematics, the system can provide position, motion, and force analyses for a wide class of linkages. The user describes a proposed linkage to KAM in a problem oriented language. This language functions solely as a means of describing the connectivity of parts in a linkage; the action statements that request calculations are specified by other means. From a linkage description in KAM language, the KAM program builds a tree-organized model of the linkage within computer memory.

The data required by KAM consist primarily of the coordinates of points in the linkage at design position and the magnitudes of input positions, motions, and forces. Position, motion, and force results are displayed in standardized formats. To calculate special parameters of interest, or to exhibit the results in a special way, the user can provide supplementary programs that further process the normal output.

Further information may be obtained from the Society of Automotive Engineers publication SP-272 which is entitled "Kinematic Analysis Method". The IBM Systems Journal (October, 1965, Vol. 4, No. 3) also contains a discussion of the System.

# GRAPHIC DATA PROCESSING

by

S. S. Husson
International Business Machines Corporation
Systems Development Division, Poughkeepsie, New York

Those of you who attended the '65 IFIP Congress this Spring must have been impressed with the emphasis given to the time sharing systems. The exhibit area had more remote terminals than an airline ticket office. Time sharing has paved the way for a new dimension in computer applications.

Basically, there are three dimensions to human communications; alphanumeric, audio, and graphic. If we apply these three dimensions to man - machine communication we will find that in the past decade we have concentrated on alphameric communications. Some achievements have been realized in the field of audio communication, but that must still be considered in the research stage. Today I will discuss graphic data processing, which now makes it possible to take graphic information in graphic form, convert it to digital form so that it can be operated upon by a computer, and bring it back out into graphic form.

This new dimension in man-machine communication is designed to help engineers, designers, management, and businessmen to work directly with curves, graphs, sketches, and engineering drawings at electronic speed. This instantaneous communication of information in graphic form reduces the time between conceiving and testing or executing an idea.

The circuit designer, with the aid of this graphic input/output can perform a dynamic simulation of circuit responses to the variations in component parameter values.

The mathematician can display complex functions and study various functional behavior and surface response.

Business management can graphically informed of up-to-date developments in manpower, inventory, scheduling, profit growth, and other important decision-making parameters.

The Graphic Data Processing System is designed to be attached to any model ot the IBM System/360. This makes the following three functions of graphic data processing available:

1. the ability to read graphics into the computer at high speed and high volume,
2. the ability to get graphics out of the computer in graphic form by using a high speed recorder,
3. the ability to manipulate graphic information using a display console and a light pen,

The main components of the IBM Graphic Data Processing system are:

1. the System/360,
2. the IBM 2250 Display Unit/Light Pen,
3. the IBM 2280 Film Recorder,
4. the IBM 2281 Film Scanner,
5. the IBM 2840 Display Control.

Through the use of a 2860 selector channel, one can operate eight 2840 Graphic Display Control Units, and each 2840 display control unit can service up to five 2250 display units and one 2281 film scanner. Theoretically, 64 different Graphic I/O units can share the same System/360 processor.

The 2250 Display Unit is organized around a 21-inch Cathode Ray Tube (CRT) having a 12" x 12" display area. All point, alphameric, and graphic information are displayed at very high speed to provide visual communication between the computer and the user. The alphameric keyboard, the light pen and the program function keyboard allow the user to communicate with the computer. The CRT display area consists of a grid format of 1024 x 1024 addressable points -- more than one million addressable points. The distance between any two points is called a raster unit. Thus, by proper combination of these raster units, the programmer can display lines, curves, surfaces, or any complex geometric figure.

Just as in TV tubes, the visable display on the face of the CRT is produced by the deflected electron beam hitting a phosphor coating, causing it to glow briefly. Therefore, information displayed on the CRT fades within a fraction of a second. In order to maintain the image displayed, it is necessary to regenerate or repaint that image approximately 30 times per second.

A vector of any length can be displayed between any two points on the display area by giving the addresses of the two end points. Each vector requires four eight-bit characters of storage. Any alphameric character can be synthronized by any number of vectors or strokes.

A character generator feature is available on the system. This permits the display of any one of 64 alphameric characters in one of two character sizes, B or L. In Size B each character is generated over an area of 28 x 28 raster units, thus providing 52 lines of 74 characters each. In Size L the area allocated for any character is 56 x 56 units, thus reducing the display to a maximum of 35 lines of 49 characters each. Consequently, the display can be performed in either a graphic mode or a character mode. In the graphic mode, a line is either blanked or unblanked to allow tracing with the light pen without painting unnecessary lines. The width of a stroke can be either 1.8 or 0.75 mils, with the beam density being specified as 0.6 or a density visibly less than this. Hence with the use of the line width and the beam density one can generate four combinations of lines with four different shades of gray.

The 2250 display unit is available in two models. Model 1 has the control circuitry and buffer memory, while model 2 has to be attached to a 2840 multiple display control unit. The 2840 control unit contains the multiplexing circuitry, the character generator, and an 8192- or 16384-byte core storage. In the case of multiple display, the 2280 provides each display unit with a local buffer for storing the image for generation. Thus it is possible to generate different images simultaneously on each display.

Due to time delay considerations, the display console has to be located no further than 2000 feet from the control unit. Data is transferred between the 2280 and the CPU under program control.

There are three optional manual inputs. These allow the user to retrieve alphameric or graphic information from memory for display, to create new images, to add and delete, to rearrange, and to store in memory or record on film.

The first and perhaps the most fascinating manual input is the light pen. It is a pen-like device, containing a photo cell that enables the user to delete or trace information on the face of the CRT. The light pen is a light detecting device. It must be held very close to and also perpendicular to the face of the CRT. Maximum tracking speed is about 15 inches per second.

The second manual input option is the alphameric keyboard, which is a standard IBM 1052 alphameric keyboard. It can be used to compose messages or revise alphameric information on the face of the CRT. When the entire message is composed, it is displayed on the screen for verifications and then transferred to the main storage. A special, arrow-like symbol called a curser is used to identify the position where the next character is to be displayed.

The third input device is a Program Function Keyboard. It consists of 32 pushbutton keys with indicator lights, and eight overlay selector switches. This overlay arrangement allows up to 256 distinct functional subroutines stored in memory to be called into action by the appropriate switch. For example, a subroutine might be written to shrink, enlarge, rotate, delete, record, or scan an image.

The next major component of Graphic Data Processing equipment is the IBM 2282 Recorder/Scanner -- actually two totally independent units. The primary mode of graphic output from the system is the recorder. Graphic information stored digitally in the computer is brought out onto 35-mm microfilm by using a high-precision cathode ray tube. In this way sketches, drawings, and graphs can be the output from a computer. The 35-mm film can then be developed and displayed on the enclosed display screen; it can be put into the familiar aperture cards; or it can be used to prepare hard-copy drawings.

This recorder has the capability of printing alphameric information of up to 40,000 characters per second or 20,000 pages per hour. The recorder is built around a high-precision, high-resolution, 5-inch cathode ray tube with 4096 x 4096 addressable points. Digital and analog control circuitry is used to project a light beam onto the unexposed, silver-emulsion film used to record images. The exposed film is transported through developing, fixing, rinsing, and air drying stations, so that it is immediately available for projection on a large screen before it is stored. The following options are available: one of two different line densities, one of two different line widths, and one of four options of distances between recorded frames.

The third component is the IBM 2281 Scanner. It is basically a high-speed, high-volume graphic-input device. It operates in the reverse sequence of the Recorder. Existing drawings and sketches can be photographed onto 35-mm film. Each photograph is scanned, again using the same high-precision CRT in the Recorder. Thus, graphic information is converted to digital form and stored in the computer.

The light beam from a CRT is directed along two paths: one path leads directly to a photomultiplier tube; the second path is directed through the film to be scanned before going to another photomultiplier tube. The intensity of the light passing through the second photomultiplier tube is compared to the light intensity passing through the first, and if the ratio exceeds a preselected threshold, the machine registers a hit. This hit, no-hit pattern is stored in a matrix form in memory -- one for a hit and zero for a no-hit. A variety of program scanning techniques can be used to register the digitized image in a minimum storage space.

A variety of scanning techniques have been devised and programmed; among them are edge following and vector scanning. This area in graphic data processing is still very much in the research stage. More sophisticated scanning algorithms are needed to handle the wide range of scanning applications one can encounter.

A second topic of interest in graphic data processing is the application area in research, engineering, manufacturing, finance, and management. The list of potential application areas for graphic data processing is extensive and impressive. It might be used in electrical, mechanical, structural, and civil engineering analysis design and engineering drawings, in ship and aircraft missiles and satellite course plotting, in kinematic analysis, in meteorological studies, and in petroleum and chemical processes. In management, one can display and modify the PERT netword drawings. Business graphs, including sales analysis, cost analysis, production control, manpower forecasts, and sales forecasts may also be displayred. In the field of mathematics, Fourier analysis and functional analysis are possibilities for application. One can go on and on listing application areas that can efficiently utilize this new concept.

I have selected two examples to examine: personnel records retrieval and analysis, and the electronic circuit analysis and design field. In the first application, the problem is to asses a file of personnel data efficiently in order to:

1. retrieve and display personnel file data,
2. update a personnel file,
3. have graphical statistics immediately available concerning personnel manpower, education, salaries, and benefits.

We all recognize the limitations of the present computing facilities in this field. We also recognize the time lag before these files are updated and before the proper statistical information is obtained and hand plotted on graph paper.

Updating is a batch operation and is done periodically. This problem can be handled on the IBM 2250 by using the functional keyboard; the personnel specialist would select any area of an employee's record file. The personnel files can be arrayed in frames pertaining to an over-all personnel summary of school training, special skills, past assignments, and any other pertinent data. The light pen can be used to update, add to, or delete from the records. This area is relatively simple; however, the concepts, problems, and problem solutions derived can be applicable to any similar alphameric information-retrieval and updating area.

The next example is in the area of electronic circuit analysis and design. In the area of nonlinear switching circuitry, which is the basic building block of any computer system, the present simulation techniques are inadequate and too expensive. In designing a switching circuit, the engineer has to satisfy constraints, such as the logic capability, maximum fan-in, fan-out, minimum turn-on and turn-off delays, minimum power dissipation, and up and down d-c levels. From an analysis of these constraints, one will realize the conflicting specifications given to an engineer. A coincidental problem is that the system of equations of the d-c or transient behavior of the system is an underdetermined system. The present method of solving this problem is to apply "engineering intuition" in assigning values to some unknown parameters in order to have a determined system of equations, and then proceed to analyze the problem. While many circuit families have been designed this way, they were not necessarily the best circuits.

To obtain an optimum circuit design one must have a good understanding of the trade-offs between these circuit specifications and the effect of the change-in for every component parameter on each specification and on the total circuit response. Also, the designer must minimize his assumptions, and thus allow himself to deal with the whole region of feasible solutions before selecting any specific design. IBM Graphic Data Processing can be an excellent tool here. The engineer can retrieve the circuit configuration he is working with, study its transient response, display a three-dimensional plot of the time delay, power dissipation, and up or down d-c levels, and then proceed to optimize his circuit. The designer, by merely pushing the correct switch on the functional keyboard, could perform any one of the many different computations on this configuration. When the circuit is designed, the circuit configuration component listing and specifications can be recorded on the Recorder, thus automatically producing a circuit flyer that can be transmitted to hundreds of personnel and locations throughout the corporation.

The IBM ECAP program has been converted to a demonstration package for internal use. This application of Graphic Data Processing will result in:

1. reduction of the problem-solution time from months to days or from days to hours;
2. giving more position feedback and more insight to the circuit operation and parameter trade-offs;
3. reducing the need for the design to work through scientific centers and programmers (the designer will have complete control over his problem).

# GAMMAMERIC CODING

Presented at the 1620 Users Group
Joint Eastern-Midwestern Meeting
October 8, 1965

by
Howard Givner

Brooklyn College of
The City University of New York

# Gammameric Coding

The purpose of this paper is to announce and describe the invention, by the writer, of a new code, called the gammameric code. This code makes it possible for the user of an IBM 1620 system to store, in a given storage area within his system, 60% more alphameric characters than he could have stored had he used only the alpha-meric code incorporated into the hardware of his system. First the need and value of new codes for storing data are established. Then two possible new coding methods are examined briefly. Finally, gammameric coding, which was an outgrowth of the other two, is presented. Details on the programming needed for its implementation are included in an appendix.

A storage problem exists if you must use your IBM 1620 to store, for reference, a file of 20000 names and addresses, on a disk pack with a storage capacity of 20000 100-digit sectors when the source of data is 20000 fully punched 80-column cards, that is, 80 columns are required to contain the entire name and address. Restated, this is the problem of putting 3.2 million digits into 2 million loca tions. By coding the raw data as it enters the system, this problem of storing in a limited on-line storage area a relatively large amount of raw data can be solved.

Since the coding system offered automatically by the hardware of the IBM 1620, alphameric coding, although powerful for sorting or comparing alphameric fields, is inefficient when only the storage of information for later retrieval is the task, a look for alternative codes which make better utilization of the available space began. If there is any fear that attempting better utilization of the available storage area by means of a new code might possibly lead to additional costs due to the encoding time used during the initial loading of the information into storage, or to the decoding time used for each subsequent reference to the stored information, that fear may be erased by knowing that those transformation times can be absorbed by overlapping the coding times with the buffered input and output operation times. Provided no additional transformation time will be needed to process the stored information internally, as is the case with a reference file, a new code offers compactness at merely the cost of developing the software to handle the new code.

The selection of an alternative to the alphameric code depends to some extent on the nature of the data to be stored, on one's preference to fixed or variable length records, and on the ultimate use of the stored information. One thing sacrificed by each of the data coding methods discussed in this paper is the ability to collate the stored data without decoding, unless an uncoded key is associated with each item of data to be stored.

The augmented character set codes to be discussed first save space in a manner similar to the method used by people when they use abbreviations and acronyms. Since only 48 of 100 possible 2-digit numbers are used in the alphameric code, some or all of the remaining 52 possible 2-digit numbers may be assigned a meaning by the user. For instance, if the data to be stored is a Fortran source deck, then 15 could be used to replace each occurrence of 1414 (exponentiation), and 07 could be used to replace each occurrence of 62585963 (the square root function). The sa ving from this method is appreciable only if a variable length record file is being used and the raw data contains multiple occurrences of lengthy character sequences.

Several augmented character sets, containing as many as 400 2-digit symbols each, may be adopted, each set adapted for use with a specific body of raw data to be stored. A 400-character set may be created by utilizing the flag, which normally carries no information in the alphameric code. For instance, 01 might stand for B, $0\overline{1}$ for P, $\overline{0}1$ for V, and $\overline{0}\overline{1}$ for F. In one application, that of student record keeping, we use an augmented character set in which a 2-digit number represents a phrase in some cases; a glossary is used for translation. To create an augmented character set, replace the notion of being limited to 48 valid alphameric code numbers where the flag bit contains no information, with the notion that there are 100 valid code numbers where the flag does contain information.

The alpha-shift code, which will be discussed next, does not require a glossary, or special conversion table, in order to produce space saving. Let us assume that the normal 48-cha racter set is sufficient for representing the data to be stored, but that the repetition of groups of characters in the raw data is not one that would suggest using an augmented character set. For example, one might have to store a list of pa rt numbers, license plate numbers, or house addresses, all of which contain many strings of consecutive numerals, but intermittantly, letters or other alphameric characters. If only numerals were used in the raw data, we would use TNS (Transmit Numeric Strip) hardware, or software, to yield a 50% storage saving over pure alphameric coding. But the zone digits discarded will not always be sevens if we used TNS on license plate numbers. Space saving can still be achieved by a compromise of storing the zone digit the first time it occurs and flagging it, followed by all the numerical digits having that same zone digit until the zone digit changes. When the zone digit shifts, that is, changes, store the new zone digit and flag it. Do not flag the numerical digits, and do not store successive occurrences of identical zone digits. For example, the raw data "175-12ST" would be stored as $\bar{7}175\bar{2}0\bar{7}12\bar{6}23$.

The idea of a shift character is not new; it appears in the 5-channel teletype code where 30 of the 32 possible code characters may have one of two meanings depending on which of the remaining two (‘shift) characters occurred most recently. This space saving technique requires variable length data records, and has no guaranteed storage saving unless the data is "suitable".

We deduce from the discussion so far that when developing a spacing saving code, the motto is DO NOT WASTE BITS. This means use the flag-bit, and use all digit combinations.

The gammameric code, a code to handle a reduced character set was developed after we examined the names and addresses we wanted to store, and found that special characters such as asterisks, parentheses, at signs, and the like, did not occur. In fact, except for numerals, letters of the alphabet, and the blank, the only special characters occurring were the period, the comma, and the hyphen. Thus we were really using a 40-character set. Knowing that in a single storage position in an IBM 1620 you can store any of 20 bit combinations of a possibility of 32combinations, if you use all 5 data bits (F, 8, 4, 2, and 1), but restrict yourself to combinations on which arithmetic can be performed. With or without binary capabilities, the only bits that may legitimately be set or cleared independently of each other in a single memory position are the F, 4, 2, and 1; the 8-bit cannot be set and cleared independently.

We assigned to each of the twenty selected bit combinations a pair of characters from our 40-character set. In order to resolve any ambiguity, one of the four bits is set on or off in another memory position. This other memory position is used to resolve four ambiguities. Thus, we are able to pack 4 characters from out 40-character set into 5 memory positions, a saving of 3 memory positions for every 8 that would be used by the alphameric code. This allows an 80-column card, containing only characters from some 40-character set, to be stored as 100 digits in a sector of disk storage. For example, the word "WALK" would be stored as "61$\overline{32}$1" using this coding system, called gammameric coding. It may be used with fixed or variable length rdcords since the space saving ratio is constant.

Whether you chose an augmented, normal, or reduced character set code, by thinking of the flag-bit as more than a field mark or algebraic sign indicator, but as a bit just as capable of storing the kind of information normally stored by the 1-. 2-, 4-, and 8-bits, you too will discover new ways of storing data in fewer memory positions in your IBM 1620 system.

We were able, for instance, to store in less than 32000 memory positions in an IBM 1620 a 500 by 500 symmetric matrix whose entries were either zero or one. Only through clever programming can we hope to exploit all features of the machine.

---

The appendix contains listings of two macros— TGS (Transmit Gamma Strip) and Tgf (Transmit Gamma Fill)— which may be added to the SPS2D subroutine set 01 and called upon to perform the conversions. We have found it best to use a mixture of TNS and TGS. The gammameric code, as it appears in a core dump, can be read almost as easily as the alphameric code by anyone familiar with the code. Although the gammameric code is applicable only to data written with characters chosen only from some 40-character set, this is rarely a serious restriction to its use.

## APPENDIX

### What Is the Gammameric Code?

Gammameric coding facilitates compact storage of large texts of alphameric data (such as a name and address file). The character set that can be handled can have at most 40 characters. One version of the code uses the 26 letters of the alphabet, the 10 numerals, a nd the blank, period, comma, and hyphen. Any other character will be treated as blank by this version of the code.

The code will allow one to store 4 alphameric characters in 5 memory positions, 3 in 4, 2 in 3 and 1 in 2.

The alphameric text to be converted into gammameric code is scanned from the left and divided into "words" of 4 characters each, and any remaining characters (from 1 to 3) form a "word". 40 is then subtracted from ea ch alphameric code. If the result is negative, a substitution is performed. For this version of the code -17 becomes 21, -37 becomes 10, -20 becomes 20 and anything else becomes 00.

A gammameric "word" is produced as follows. The numeric parts of the alphameric "word" become the first digits of the gammameric "word". A flag is placed over the digit in the gammameric "word" if the zone digit of the alphameric character from which it came had a 1-bit. To complete the gammameric "word", a last digit is appended whose 1-bit, 2-bit, 4-bit and flag-bit respectively are used to indicate the presence of a 2-bit in the zone digit of the 1st, 2nd, 3rd, or 4th character of the alphameric"word" being converted into gammameric code. See diagram on page 12.

Examples: WALK becomes 613 $\bar{2}$1 RUST becomes $\bar{9}$423$\bar{6}$

H3 becomes 8$\bar{3}$2

### Gammameric Conversion Macros

TGS A,G,N (to convert alphameric into gammameric)

TGF A,G,N (to convert gammameric into alphameric)

N is the number of characters to be converted.

A is the even core storage address of the left-most character in the alphameric data. Flags in the alphameric data do not affect the result produced by TGS. The flag status in the area to receive the alphameric result produced by TGF will be unaltered.

G is the address of the left-most digit in the gammameric data. This may be in an odd or even memory position.

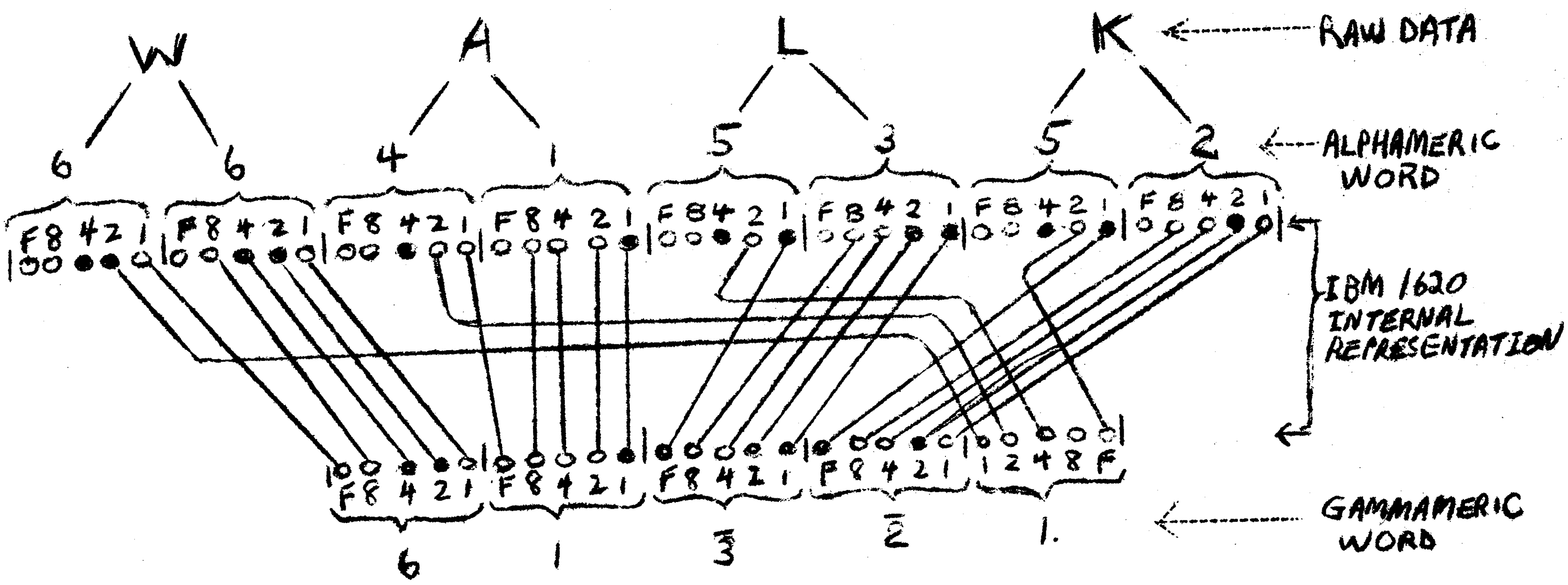
In no case should the da ta in either the gammameric or alphameric records "wrap-around" the end of memory, or else the macros will "hang-up" with a MAR CHECK. See listings (pp8-11).

```
ZZJOB 50
ZZXEQ SPSLIB
*DEFINE OP CODE
           TGS -181
*ENDLIB
ZZZZ
ZZJOB 5
ZZSPS
*ASSEMBLE RELOCATABLE
*LIBR
*ID NUMBER 0118
*STORE RELOADABLE
*OUTPUT CARD
*LIST PRINTER
**                               TRANSMIT GAMMA STRIP (TGS)
00010*     TGS A,G,N  (ASSUMED CALLING STATEMENT)
00020*     TFM 2375,*+19,,SPS2D GENERATED LINKAGE
00030*     B7  TGS
00040*     DSA A,G,N
00050*     DSC 1,-,,END OF GENERATED LINKAGE
00060      DSA TGS
00070TGS   TR  A-4,-2375,,FETCH PARAMETERS
00080      AM  2375,17,10,CALCULATE RETURN ADDRESS
00090LQ    CM  N,0,10
00100      BNP -2375,,,RETURN TO CALLING PROGRAM
00110      TFM C,00,10
00120      TFM P,01,10
00130      TF  K,N
00140      CM  K,4,10
00150      BNH *+24
00160      TFM K,4,10
00170      SM  N,*-*
00180K     DS  5,*,COUNT UP TO 4
00190LP    TD  T-1,-A,,PICK UP ZONE DIGIT
00200      AM  A,1,10
00210      TD  T,-A,,PICK UP NUMERICAL DIGIT
00220      AM  A,1,10
00230      CF  T,,,SAFETY ONLY
00240      BNR *+24,T
00250      TFM T,,10,SUPPRESS TROUBLE
```

```
0260        TFM *+47,TBL
0270        A   *+35,T
0280        A   *+23,T
0290        TF  T,*-*,,ENCODE
0300        TD  -G,T,,PUT OUT THE CODE
0310        AM  G,1,10
0320        BD  *+24,T-1
0330        A   C,P
0340        AM  *+11,*-*,10,GET NEXT POWER OF 2
0350P       DS  2,*,,POWER OF 2
0360        SM  K,1,10
0370        BP  LP
0380        CM  C,08,10
0390        BL  *+36
0400        SM  C,08,10
0410        SF  C
0420        TDM -G,*-*,,PUT OUT THE SUFFIX
0430C       DS  2,*,GAMMAMERIC SUFFIX
0440        AM  G,1,10
0450        B   LO
0460T       DC  3,0,,HOLD ONE ALPHAMERIC CHARACTER
0470*           PERIOD(03=50),BLANK(00=40),COMMA(23=61),HYPHEN(20=60)
0480TBL     DVLC,2, 10,2, 10,2, 10,2,-10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0490        DVLC,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0500        DVLC,2, 00,2, 10,2, 10,2, 01,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0510        DVLC,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0520        DVLC,2, 10,2, 11,2, 12,2, 13,2, 14,2, 15,2, 16,2, 17,2, 18,2, 1
0530        DVLC,2, 10,2,-11,2,-12,2,-13,2,-14,2,-15,2,-16,2,-17,2,-18,2,-1
0540        DVLC,2, 10,2, 10,2, 02,2, 03,2, 04,2, 05,2, 06,2, 07,2, 08,2, 0
0550        DC  2,-00
0560        DVLC      ,2,-01,2,-02,2,-03,2,-04,2,-05,2,-06,2,-07,2,-08,2,-0
0570        DVLC,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0580        DVLC,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 10,2, 1
0590A       DS  5,,SENDING ADDRESS
0600G       DS  5,,RECEIVING ADDRESS
0610N       DS  5,,NUMBER OF CHARACTERS
0620        DS  1,,CATCH RECORD MARK AT ENTRY TIME
0630        DEND 01 18 1 ,,,TGS
777
```

```
ZZJOB 50
ZZXEQ SPSLIB
*DEFINE OP CODE
              TGF -191
*ENDLIB
ZZZZ
ZZJOB 50
ZZSPS
*ASSEMBLE RELOCATABLE
*LIBR
*ID NUMBER 0119
*STORE RELOADABLE
*OUTPUT CARD
*LIST PRINTER
 *                                   TRANSMIT GAMMA FILL (TGF)
00010*       TGF A,G,N    (ASSUMED CALLING STATEMENT)
00020*       TFM 2375,*+19,,SPS2D GENERATED LINKAGE
00030*       B7  TGS
00040*       DSA A,G,N
00050*       DSC 1,-,,,END OF GENERATED LINKAGE
00060        DSA TGF
00070TGF     TR  A-4,-2375,,FETCH PARAMETERS
00080        AM  2375,17,10,CALCULATE RETURN ADDRESS
00090LO      CM  N,00,10
00100        BNP -2375,,,RETURN TO CALLING PROGRAM
00110        TF  K,N
00120        CM  K,04,10
00130        BNH *+24
00140        TFM K,4,10
00150        SM  N,*-*
00160K       DS  5,*,COUNT UP TO 4
00170        TF  *+35,G
00180        A   *+23,K
00190        TD  C,*-*,,PICK UP GAMMAMERIC SUFFIX
00200        TDM C-1,0,11
00210        BNF *+36,C
00220        CF  C
00230        AM  C,08,10
00240        A   C,C,,2 TIMES C
00250        A   C,C,,4 TIMES C
00260        TFM *+35,TBL1
00270        AM  *+23,*-*
00280C       DS  2,*,HOLD 1 GAMMAMERIC SUFFIX
00290        TF  W-1,*-*
```

```
00300LP      TFM BAS,TBL2
00310        BD  *+24,W-4
00320        AM  BAS-1,4,10
00330        TD  *+59,-G
00340        BNF *+36,*+47
00350        CF  *+35
00360        AM  BAS-1,2,10
00370        AM  BAS,*-*,10
00380        A   BAS,*-1
00390        AM  G,1,10
00400        TF  T,-BAS
00410BAS     DS  5,*
00420        CF  T-1
00430        BNF *+24,-A
00440        SF  T-1,,,PRESERVE FLAG STATUS
00450        TD  -A,T-1
00460        AM  A,1,10
00470        BNF *+24,-A
00480        SF  T
00490        TDM -A,*-*
00500T       DS  2,*,HOLD 1 ALPHAMERIC CHARACTER
00510        AM  A,1,10
00520        TR  W-4,W-3
00530        SM  K,1,10
00540        BP  LP
00550        AM  G,1,10
00560        B   LQ
00570W       DC  5,-,,HOLD TABLE 1 ENTRY
00580TBL1    DVLC,4,1111,4,0111,4,1011,4,0011,4,1101,4,0101,4,1001,4,0001
00590        DVLC,4,1110,4,0110,4,1010,4,0010,4,1100,4,0100,4,1000,4,0000
00600*            PERIOD(03=50),COMMA(23=61),BLANK(00=40),HYPHEN(20=60)
00610TBL2    DVLC,2,00,2,41,2,42,2,43,2,44,2,45,2,46,2,47,2,48,2,49
00620        DVLC,2,03,2,51,2,52,2,53,2,54,2,55,2,56,2,57,2,58,2,59
00630        DVLC,2,20,2,23,2,62,2,63,2,64,2,65,2,66,2,67,2,68,2,69
00640        DVLC,2,70,2,71,2,72,2,73,2,74,2,75,2,76,2,77,2,78,2,79
00650A       DS  5,,RECEIVING ADDRESS
00660G       DS  5,,SENDING ADDRESS
00670N       DS  5,,NUMBER OF CHARACTERS
00680        DS  1,,CATCH RECORD MARK AT ENTRY TIME
00690        DEND 01 19 1 ,,,TGF
777
```



This illustration shows how alphameric-to-gammameric conversion is accomplished with the 40-character set consisting of 26 letters (A to Z), 10 numerals (0 to9), and 4 special characters: blank, hyphen, ampersand, and slash. Vertical strokes indicate memory position boundaries. Check bits are not shown.

**LONG ISLAND UNIVERSITY • COMPUTER LABORATORY**

Dr. Harold Joseph Highland, Director

# CO/STATS — COMPUTER ORIENTED STATISTICAL TEACHING AND TESTING SERIES

....Computer Program Package for Teaching Introductory Statistics....

by Dr. Harold Joseph Highland
Director, Computer Laboratory
The Brooklyn Center of
Long Island University

presented at the Eastern-Midwestern
Common Conference, October 8, 1965
at the Americana Hotel, New York City

## Computer's Role in Education

At The Brooklyn Center of Long Island University we refer to our computer complex as the "Computer Laboratory." This was done with prescience since we view the computer and its concomitant equipment as an experimental laboratory in science and business.

We have a 1620 Mark I unit with 20K memory and the basic peripheral equipment, such as 026 key punches, 082 sorter and 407 printer. We operate in an atmosphere similar to that found on the campus of many liberal arts colleges. There are some aficionados but there are many more inimical faculty members clothed in the fig leaves of admiration of the past. We operate a hybrid open=and=closed shop. Because of my other post, that of Director of the Office of Instructional Services, our computer laboratory performs some administrative work. The confidential nature of this work requires a "closed shop," wherein students are not permitted in the computer laboratory. On the other hand, during the remainder of the time, we operate an "open shop," wherein students have free access to all equipment, including the console of the 1620.

In addition to using the computer for faculty and student research, it is also used for the teaching of two introductory courses in computer science, and it is integrated into the teaching of operations research, which I teach at the graduate level, marketing, management and statistics. Furthermore, our computer laboratory is used for test scoring and analysis of classroom tests, finals and comprehensive tests. It is because of our testing and research programs that we started building an extensive statistical program library.

## Computer and the Teaching of Statistics

Until this semester I presented the common statistics lecture which all students attended simultaneously. Since other teachers taught the quiz sections in statistics, I used the lecture to coordinate the teaching of statistical concepts and techniques. The use of the computer in the quiz sections is viewed as an attempt to maximize the teaching process and to increase its efficiency. Specifically, **CO/STATS - Computer Oriented Statistical Teaching and Testing Series -** was prepared:

- in an effort to improve the teaching of statistical principles and basic concepts, and
- to reduce and possibly eliminate the traditional number pushing associated with the conventional statistical laboratory.

**CO/STATS** was prepared for students who will be using statistics in their professional and business careers and not for the training of statisticians! By using this computer program package, we hope:

- to acquaint the students with the applications of limitations of the computer in statistical analysis,
- to illustrate the sensitivity of data to analysis, something that is virtually overlooked in the conventional teaching of statistics in most of the colleges,
- to train the students in the evluation of data printouts, a simulation of real-life activity which they will experience later in their work,
- to introduce students in the use of program libraries | at this stage our own and some of the Users Group programs | and to teach them how to prepare data for use with these programs, and
- to make them aware of the need to consider the type and form of output so that they can request the data in the shape best needed by them in their work when they work with computer personnel on the job.

To a large extent, many of these objectives are achieved through the addition of a myriad of comment cards with the programs. Furthermore, some of the programs are 'loosely' written and could be tightened up for shorter running time and appear more professionally written, following the techniques of Iverson, for example. However, many have been left in this "crude" form since they are easier for the student to understand and to follow the individual statements.

Some of these points can be illustrated best by the accompanying samples from the CO/STATS programs.

- Illustration 1 | page 4 | is a sample of a basic statistical analysis program for use with frequency array data. Each program contains a series of comment cards for "Identification of Variables." Every label used in the program is identified to make it easier for the student to follow the logic of the program. In this instance, there is also the warning note about program input - the output cards from LIU/26.1.01 - and the note limiting the analysis to no more than 15 class intervals.

- Illustration 2 | page 5 | is a sample of information flow contained within the program. Since these students are not being trained as statisticians, we spend only a short period of time in both lecture and quiz classes on the analysis of skewness, $\beta_1$ , and kurtosis, $\beta_2$. Classroom instruction is reinforced by inclusion in the program of comment cards about both of these statistical measures. Furthermore, I have included for the students and some of the faculty too, the reference for testing the significance of both $\beta_1$ and $\beta_2$ .

- Illustration 3 | page 6 | is the printout of our basic statistics program, LIU/26.0.01 used for the analysis of raw data. It is the companion program for LIU/26.0.02 which is used to provide almost identical analysis of frequency array data. You will note that the output is in F-format, which is easily understood even by the beginning student, or for that matter, any faculty member for whom the computer laboratory does research analysis. However, along the bottom line of the printout are four values in E-format, namely, $\Sigma X$, $\Sigma X^2$, $\Sigma X^3$ and $\Sigma X^4$. They were prepared in this format to produce a compact single line printout and, at times, this one line is removed if we feel that the receipient will encounter difficulty in reading these data. Furthermore, note that we have prepared variance, $\sigma^2$, the standard deviation, $\sigma$ , and coefficient of variation, V, for both N and N-1 since we have requested for analysis in both forms.

```
C      BASIC STATISTICS FOR FREQUENCY ARRAY DATA
C
C      ¤ CO/STATS 26.0.02 ¤                      DR. HAROLD JOSEPH HIGHLAND
C
C      ¤ COMPUTER-ORIENTED/                       DIRECTOR, COMPUTER CENTER
C      STATISTICAL TEACHING                          THE BROOKLYN CENTER OF
C      AND TESTING SERIES ¤                          LONG ISLAND UNIVERSITY
C
C
C      ¤¤¤USE WITH OUTPUT OF LIU/26.1.01 AS INPUT FOR THIS PROGRAM¤¤¤
C      ¤¤¤LIMITED TO 15 CLASS INTERVALS MAXIMUM
C
C
C      IDENTIFICATION OF VARIABLES
C      BLCI     LOWER LEVEL OF CLASS INTERVALS
C      CF       CUMULATIVE FREQUENCY
C      D        DIFFERENCE IN CLASS INTERVALS - FIRST CI IS BASE --ZERO--
C      D2       SQUARE OF D
C      DEL1     DIFFERENCE FOR MODE COMPUTATION
C      DEL2     DIFFERENCE FOR MODE COMPUTATION
C      F        FREQUENCY OF CLASS INTERVAL
C      FD       F * D
C      FD2      F * D2
C      I        CONTROL FOR COUNT
C      ICDE     ID CODE
C      ID       CARD NUMBER OF EACH INPUT CARD
C      IDMX     CLASS INTERVAL OF MAXIMUM FREQUENCY
C      IN       CONTROL FOR COUNT
C      MFC      MAXIMUM FREQUENCY COUNT
C      SAVE     QUARTILE DEVIATION
C      SF       SUM  OF  F  OR  N
C      SFD      SUM  OF  FD
C      SFD2     SUM  OF  FD2
C      SIZE     SIZE OF CLASS INTERVALS
C      SKP      SKEWNESS COEFFICIENT --- PEARSON
C      SN       NUMBER
C      ULCI     UPPER LIMIT OF CLASS INTERVALS
C      V        COEFFICIENT OF VARIATION
C
C
    2 FORMAT%I5¤
    4 FORMAT%28HSEQUENCE ERROR- REINITIALIZE¤
    6 FORMAT%36HBIMODAL DISTRIBUTION - NO MODAL DATA¤
```

```
C     BASIC STATISTICS FOR RAW DATA ANALYSIS
C
C     ¤ CO/STATS 26•0•01 ¤                      DR• HAROLD JOSEPH HIGHLAND
C
C     ¤ COMPUTER-ORIENTED/                       DIRECTOR, COMPUTER CENTER
C     STATISTICAL TEACHING                          THE BROOKLYN CENTER OF
C     AND TESTING SERIES ¤                          LONG ISLAND UNIVERSITY
C
C
C     ¤ NOTES ABOUT SKEWNESS ¤
C
C     MOST SKEWED CURVES IN SOCIAL SCIENCE, EDUCATION AND BUSINESS ARE
C     SKEWED TO THE RIGHT, OR POSITIVELY SKEWED•
C
C     ¤ BETA 1 ¤   VALUE IS ZERO FOR A NORMAL CURVE•   THE RANGE IS
C     FROM MINUS 3 TO PLUS 3•
C
C     ¤ BETA 2 ¤   THE NORMAL CURVE, MESOKURTIC, HAS A VALUE OF PLUS 3•
C     EXCESSIVE PEAKEDNESS, A LEPTOKURTIC CURVE, HAS A BETA 2 VALUE OF
C     LESS THAN PLUS 3•   ON THE OTHER HAND, A SPREAD CURVE, PLATYKURTIC,
C     HAS A BETA 2 VALUE OF LESS THAN PLUS 3•
C
C     TO TEST THE SIGNIFICANCE OF BOTH BETA 1 AND BETA 2 REFER TO
C     E• S• PEARSON AND H• O• HARTLEY   BIOMETRIKA TABLES FOR STATISTICIANS
C     VOLUME 1   CAMBRIDGE UNIVERSITY PRESS   1954   SEE PAGES 183-184
C     FOR TABLES OF
C     UPPER 0•10 AND 0•02 LIMITS OF BETA 1, AND
C     UPPER 0•05 AND 0•01 LIMITS OF BETA 2
C
C
   10 FORMAT%42HTHIS IS PROGRAM/LIU 26•0•01/STAT-EATKS HJH¤
   16 FORMAT%14HENTER Z VALUES¤
   18 FORMAT%3F10•4¤
   40 FORMAT%5HRANGE,5X,F12•2/¤
```

```
LIU COMPUTER CENTER

STATISTICAL ANALYSIS LIU 26•0•01/HJH

NUMBER           11•
MEAN             14•4090
MAXIMUM X        20•00            MINIMUM X        11•10
RANGE             8•90

RELATIVE SKEWNESS * BETA 1         •417558
KURTOSIS          * BETA 2        2•003520

                                         N                N-1

VARIANCE                              8•5022            9•3524
STANDARD DEVIATION                    2•9158            3•0581
COEFFICIENT OF VARIATION             20•2362           21•2239

GAUSSIAN LIMITS OF DISTRIBUTION
    68•72% *1 SIGMA*              11•4932      17•3249
    95•45% *2 SIGMA*               8•5773      20•2408
    99•73% *3 SIGMA*               5•6615      23•1566


STANDARD ERROR OF MEAN ¤N¤      •8791
                       ¤N-1¤    •9220

CRITICAL LIMITS OF MEAN
5%LEVEL            12•6859       16•1322
1%LEVEL            12•1408       16•6773
 •1%LEVEL          11•5166       17•3015

SUMX 15•8500E&01   X2, 23•7736E&02   X3, 37•1270E&03   X4, 60•2431E&04
```

The companion program, LIU/26.0.02, about which we just spoke has been used in our statistics classes to aid teachers in scoring the statistical laboratory projects of the students. We found, as undoubtedly every teacher before us has, that a common laboratory project given to a class often results in cooperative common effort. Either the students divided the calculator work among themselves or one student does the job and the other ride on his coat tails.

In an attempt to make certain that each student benefits from working with calculators in the statistics laboratory, we developed a series on individual student projects. Each student is given a common worksheet for layout of his work and indicating what statistical measures are necessary for the completion of the project. Although some duplication of raw data is necessary because of the large number of students we have taking statistics, we shuffle the cards before the printout if it is necessary to prepare more than one copy of a printout of raw data. Furthermore, the students are told the lowest level of the lowest class interval and are instructed to use a specific class interval.

The teacher does not have to work each project out individually to check the accuracy of the work of each student. Instead, LIU/26.0.02 is used and provides the teacher with a complete array and the common statistical measures required by the project. He uses this printout, see Illustration 4 | page 8 | to grade the student's laboratory project and gives the student a copy of the printout so that he has the worksheet and answers.

## Program Range of CO/STATS

Although CO/STATS was designed for use in statistics classes, it has also been used in the introductory course in computer science. Likewise, the same program, LIU/26.2.01, Central Tendency - Mean, has been used to acquaint statistics students with introductory FORTRAN.

The program to compute a simple arithmetic mean is written in FORTRAN for card input and output. Each step is explained by using comment cards so that the student knows what is being done. In practice, the program has been processed with a trace so that computer science students can follow the machine operation. Illustration 5 | page 9 | is a sample of this FORTRAN=teaching and statistical methods program.

```
ILLUSTRATION 4                                              CO/STATS  8


LIU COMPUTER CENTER


LIU/26.0.02/HJH

CODE       13

        CLASS INTERVAL        FREQ       D     D2        FD          FD2
        .000 -      3.999      10.       .      .         .            .
       4.000 -      7.999      25.      1.     1.       25.          25.
       8.000 -     11.999      40.      2.     4.       80.         160.
      12.000 -     15.999      15.      3.     9.       45.         135.
      16.000 -     19.999      10.      4.    16.       40.         160.


          SUMS OF DATA        100.                     190.         480.


   □□□OUTPUT ANALYSIS□□□


     MEAN                          9.600   COEFFICIENT OF VARIATION       45.452
     MEDIAN                        9.500   SKEWNESS - PEARSON              -.068
     MODE□ADJUSTED□                9.500   SKEWNESS-QUARTILE               -.218
     STANDARD DEVIATION            4.363   K-VALUE-CENTR                   8.800
     QUARTILE DEVIATION            3.200   Q1          5.6000  Q3         12.000
```

ILLUSTRATION 5

```
C       CENTRAL TENDENCY - MEAN
C
C       ¤ CO/STATS 26.2.01 ¤                    DR. HAROLD JOSEPH HIGHLAND
C
C       ¤ COMPUTER-ORIENTED/                     DIRECTOR, COMPUTER CENTER
C       STATISTICAL TEACHING                       THE BROOKLYN CENTER OF
C       AND TESTING SERIES ¤                       LONG ISLAND UNIVERSITY
C
C
C          INITIALIZATION OF PROGRAM ¤ SET VALUES EQUAL TO ZERO
    13 SUMX#0.
       COUNT # 0.
       AV#0.
C          FIRST CARD CHECK -- TELLS MACHINE TO CHECK FOR CARD TO READ
       IF%SENSE SWITCH 9¤ 1,1
C          INSTRUCTIONS ARE READ IN FORMAT -- 000.00 WITH CARD PUNCHED
C          WITH THE VALUES IN THE FIRST FIVE COLUMNS WITH NO DECIMAL POINT
     1 READ 2,X
     2 FORMAT%F5.2¤
C          START OF ACTUAL CALCULATIONS -- SUM OF THE X-VALUES, COUNTING
C          OF THE NUMBER OF CARDS READ ¤ EACH CARD HAS ONLY ONE VALUE ¤
       SUMX # SUMX&X
       COUNT # COUNT & 1.
C          LAST CARD CHECK -- IF NO MORE CARDS, GO TO STATEMENT #3.
C          OTHERWISE, GO TO STATEMENT #1, THAT IS READ THE NEXT CARD
       IF%SENSE SWITCH 9¤3,1
C          AFTER LAST CARD IS READ -- THE COMPUTATION OF THE AVERAGE
C          OR MEAN OF THE VALUES IS DONE
     3 AV # SUMX/COUNT
C          PUNCH AVERAGE IN FORMAT 00000.00 AND TOTAL NUMBER OF CARDS
C          OR NUMBER OF VALUES READ IN FORMAT -- 000. AND TOTAL OF ALL
C          VALUES OF SUM OF X IN FORMAT 00000000.00
       PUNCH 4,AV, SUMX,COUNT
     4 FORMAT%7HAVERAGE,F8.2,10X,6HTOTALX,F11.2,10X,12HTOTAL NUMBER,F4.0¤
       PAUSE
C          AFTER THE MACHINE READS A  ¤PAUSE¤  STATEMENT, IT HALTS AND WAIT
C          IF YOU WISH TO ENTER NEW DATA, YOU PRESS ¤START¤ ON THE CONSOLE
C          AND THE PROGRAM FOLLOWS THE  ¤GO TO 13¤  INSTRUCTION - BACK TO
C          THE BEGINNING OF THE PROGRAM AND FOLLOWS THROUGH
       GO TO 13
       END
C          AN  ¤END¤  STATEMENT IS REQUIRED IN EVERY FORTRAN PROGRAM
```

At the other extreme in introductory statistics are programs for testing significance, both t and F tests, intracorrelation and multiple correlation programs. In addition, special programs with applications for business and economics majors, such as index number construction and linear regression analysis, as well as programs for education and psychology majors, complex analysis of variance and Spearman rank order correlation, are included in the series.

CO/STATS is a dynamic series. We are adding about a program a week to those already debugged and operating. Among the programs now in "student use," that is usable by students without any additional help, are:

- Basic Statistics for Raw Data Analysis - LIU/26.0.01
- Basic Statistics for Frequency Array Data - LIU/26.0.02
- Sturges' Rule and Frequency Distribution - designed to indicate the correct number of class intervals according to Sturges' rule and the setting up of data in frequency array; these output cards serve as input for a series of the 26. programs.
- Frequency Array Display - histogram presentation of frequency array compiled in preceeding program.
- Data Conversion to Ordered Array - transformation of raw data into an ordered array using "bubble sort" method developed by Kenneth E. Iverson. [1]
- Central Tendency - Mean - combination of FORTRAN programing and statistics for arithmetic average.
- Positional Measures - Median and Quartiles - for use with raw data and includes standard deviation, coefficient of skewness based on Pearson formula as well as quartile distribution, interquartile range, etc.
- Arithmetic, Geometric and Harmonic Means - for use with raw data too illustrate characteristics of these measures.

[1] For "bubble sort" method, see K. E. Iverson, "Programming Notation in Systems Design," IBM **Systems Journal**, Volume Two, June 1963, page 120; also see K. E. Iverson, **A Programming Language**, New York: John Wiley & Sons, 1962.

- Dispersion and Variability - Standard Deviation - similar to arithmetic mean program, combining FORTRAN teaching with basic statistics; requires two passes and verifies N.

- Hypothesis Testing: Tests of Significance - T-test - this permits the use of the t-test with a series of arrays and obtains the intra t-tests. See Illustration 6 |page 12| for sample of output of this program.

- Hypothesis Testing: Tests of Significance - F/ratio - three forms of input are possible under control of header card; this was done with coordinate this program with others used for research and teaching in our computer laboratory.

- Linear Regression and Correlation Coefficient - printout includes correlation coefficient, test of significance, associated and unassociated variation, formula for forecast and permits the running of projected estimates with the series.

- Spearman Rank Oder Correlation - conventional educational and psychological correlation program.

- Spearman Product Moment Correlation - for use by education and psychology teachers and students.

- Intracorrelation Analysis/ Four variables - intracorrelation of all combinations produced by this program.

- Multiple Corrleation Analysis - simple multiple correlation program for use with up to six variables.

- Least Squares Trend Line - for use by business students in analysis and forecasts in times series analysis.

- Simple Index Numbers - for use with three items for a maximum of 15 time periods, producing aggregate, geometric, harmonic and average of relatives indexes.

- Weighted Index Numbers - likewise for use with three items for a maximum of 15 time periods, producing Paasche, Laspeyres, Fisher, average cost and average of relatives indexes.

- Chi-Square $\chi$ Test - for use with a 2x2 contingency table.

- Normal Curve Generation for Frequency Array Data - obtains theoretical frequency by class interval for frequency array data.

ILLUSTRATION 6

LIU COMPUTER CENTER

```
LIU/26.4.01/T-TEST - RATIO ANALYSIS/HJH

TEST  DATA          --MEAN--            VARIANCE         NUMBER             T
FOR SERIES        ¤A¤        ¤B¤       ¤A¤        ¤B¤   ¤A¤     ¤B¤

    1 -  2       3.20       2.33      1.03        .56  110.    115.        7.307
    1 -  3       3.20       2.33      1.03        .56  110.    115.        7.307
    1 -  4       3.20       3.67      1.03        .91  110.    113.       -3.548
    1 -  5       3.20       4.10      1.03        .70  110.    112.       -7.182
    1 -  6       3.20       4.21      1.03        .86  110.    114.       -7.745
    1 -  7       3.20       3.45      1.03        .86  110.    111.       -1.903

    2 -  3       2.33       2.33       .56        .56  115.    115.         .000
    2 -  4       2.33       3.67       .56        .91  115.    113.      -11.760
    2 -  5       2.33       4.10       .56        .70  115.    112.      -16.735
    2 -  6       2.33       4.21       .56        .86  115.    114.      -16.815
    2 -  7       2.33       3.45       .56        .86  115.    111.       -9.963

    3 -  4       2.33       3.67       .56        .91  115.    113.      -11.760
    3 -  5       2.33       4.10       .56        .70  115.    112.      -16.735
    3 -  6       2.33       4.21       .56        .86  115.    114.      -16.815
    3 -  7       2.33       3.45       .56        .86  115.    111.       -9.963

    4 -  5       3.67       4.10       .91        .70  113.    112.       -3.577
    4 -  6       3.67       4.21       .91        .86  113.    114.       -4.305
    4 -  7       3.67       3.45       .91        .86  113.    111.        1.741

    5 -  6       4.10       4.21       .70        .86  112.    114.        -.931
    5 -  7       4.10       3.45       .70        .86  112.    111.        5.471

    6 -  7       4.21       3.45       .86        .86  114.    111.        6.118
```

51K

□ Under construction at this time is a testing generator program which will require that students write the necessary statistical formulas, having been given the labels for basic statistics terms, and which will produce numerical output. The student's printout can be compared with the test printout to determine his accuracy in statistical manipulation and his ability in FORTRAN programing.

Several related programs, outside the 26=series, are usable in the teaching of statistics. Two of these, for example, involve Gamma distributions as well as Poisson distributions, the necessary elements in queueing theory analysis. One of these, LIU/350.21, "Basic Queueing Model," produces an analysis of arrival and waiting time rates. Its format in printout is easily understandable so that it can be used in business management courses with students who have not had FORTRAN. See Illustration 7 | page 14 | for sample printout. The other program, LIU/350.20, "Queueing Similation Model," is more in the realm of applied statistics, but like its companion, just mentioned, is usable in both statistics and management courses. See Illustration 8 | page 15 |

## Limitations of Computer in Teaching Statistics

Essentially the computer console and peripheral equipment should be part of a classroom design or else there should be a remote console in the classroom if the educational objectives of CO/STATS are to be achieved. While considerable program can be made by using this method, there are several constraints which are present on any campus.

- The available of computer time as well as its cost may require only limited application of the CO/STATS programs.
- Time and energy of students are also limiting factors; this method is somewhat slower than conventional teaching if we use standard textbooks and lecture method; use of programed textbooks and visuals in teaching may make it possible to proceed at a faster rate in the classroom especially since we have high motivation on the part of students since the computer is used.
- Finally, time and energy on the part of the teacher is also a constraining factor. Until the teacher really organizes his materials and integrates these with the computer programs and classroom discussions, the use of CO/STATS puts a heavy drain upon the teacher for the course.

ILLUSTRATION 7

```
LIU COMPUTER CENTER

BASIC QUEUING MODEL/LIU/350.21/H.J.HIGHLAND

         AVERAGE ARRIVAL RATE          2.00
         AVERAGE SERVICE RATE          3.00

   MEAN TIME BETWEEN ARRIVALS           .50
         MEAN NUMBER IN QUEUE          1.33
            MEAN WAITING TIME           .66

        MEAN NUMBER IN SYSTEM          2.00
          MEAN TIME IN SYSTEM          1.00

        UTILIZATION PARAMETER           .66 PERCENT
         PERCENTAGE IDLE TIME           .33 PERCENT

   PROBABILITY OF N-UNITS IN THE SYSTEM   CUMULATIVE
       QUEUE LENGTH       PROBABILITY
            NONE          .3333          .3333
             1            .2222          .5555
             2            .1481          .7037
             3            .0987          .8024
             4            .0658          .8683
             5            .0438          .9122
             6            .0292          .9414
             7            .0195          .9609
             8            .0130          .9739
             9            .0086          .9826
            10            .0057          .9884
            11            .0038          .9922
            12            .0025          .9948
            13            .0017          .9965
            14            .0011          .9977
            15            .0007          .9984
            16            .0005          .9989
            17            .0003          .9993
            18            .0002          .9995
            19            .0001          .9996
            20            .0001          .9997
```

ILLUSTRATION 8

```
LIU COMPUTER CENTER

QUEUING SIMULATION MODEL/LIU/350.20/HJH

NO.  ARRIVAL                     SERVICE
       TIME     -SERVICE-         TIME     IDLE   WAITING     QUEUE
              BEGINS    ENDS                   --TIME--      LENGTH


 1       0        0        6        6        0        0         1
 2       4        6       12        6        0        2         3
 3       9       12       21        9        0        3         2
 4       9       21       29        8        0       12         1
 5       9       29       36        7        0       20         0

                     TOTAL         36        0       37

              AVERAGE LENGTH OF QUEUE      1.40
              AVERAGE WAITING TIME         7.40
              AVERAGE SERVICE TIME         7.20

              UTILIZATION OF SYSTEM     100.00 PERCENT
              IDLE TIME FACTOR             .00 PERCENT
```

## Availability of CO/STATS Programs

Integration of the computer and the teaching of statistics is progressing on several campuses. Among those with which I am comparatively familiar are the work done by E. E. Remmenga and Wade Halvorson and William Owen at Colorado State University, Thomas E. Kurtz at Dartmouth and Nat Goldfarb at Hofstra University.

I am beset by several decisions in making this series available to other institutions. First, the series is not finalized, but this will no longer by a problem by the end of the year.

Second, I am torn between making these available as part of a book, which one of the publishers would like to produce, or turn them over as a package to the 1620 Users Library. My interest in CO/STATS is that of a teacher and not a programer. Although publishing another book is looked upon with favor in the acedmic community, I feel that these programs are so fundamental that they would do more good if they were available on cards immediately and be accompanied by special manuals. Until that decision is made, however, I should be happy to make CO/STATS available directly from the LIU Computer Laboratory to any of you who wish to use these programs at your school. I am preparing a brief bibliography on the series and shall forward this upon request. Those wishing any specific program, or even the series, will then be able to obtain the necessary cards from my office.

There is, however, one condition which I place upon my sending of the program cards to any individual. I should like to hear from you and learn about your experiences in teaching statistics with computers. True maximization of teaching effort is a cooperative venture; maybe you'd like to join with me in forming a group to advance this area.

Dr. Harold Joseph Highland
Director, Computer Laboratory
The Brooklyn Center
Long Island University
Brooklyn, New York 11201

User #1429

Presented to Eastern-
Midwestern Users Group
Conference, New York
City, October 7, 1965

# BATCH LOAD AND GO STEPWISE MULTIPLE LINEAR REGRESSION PROGRAM

by

Kenneth L. Daniels
Instructor in Mathematics
North Dakota State University
Fargo, North Dakota

## Introduction

It frequently happens in an experimental situation that one is concerned with the problem of estimating or predicting the value of one variable from a knowledge of other variables. It is desirable to express the relationship in mathematical form by determining an equation connecting the variables. This equation, called a regression equation, is of the form

$$X_1 = b_1 + b_2 X_2 + b_3 X_3 + b_4 X_4 \ldots\ldots\ldots$$

where the variable $X_1$ is called the dependent variable and the variables $X_2$, $X_3$ ..... $X_n$ are called independent variables. In addition to determining the equation that best fits the data supplied, it is advantageous to have some measure of the goodness of fit of this equation. Some means is necessary to determine if the best estimate of the dependent variable comes from a regression equation containing one variable or several variables or even if any estimate of the dependent variable is significant.

The purpose of this program is to provide enough information to determine which variables, if any, add significantly to the total regression and to provide the equation of best fit including those variables.

## Discussion of the Program

At N.D.S.U. a considerable amount of regression data is processed. The program that was being used to process this data had a rather complex header, transformation and input format. In addition, the program would not allow batch running of data. Therefore, it took a considerable amount of time to process data, because time was wasted reading in the program for each run. Since N.D.S.U. has an open-shop policy, people who had a need for the results of the program were required to know how to run it. Much time was wasted by these people in trying to figure out how to run the program and prepare the input data. In order to minimize the total running time by allowing batch compiling and to provide a larger group of users with an easy-to-understand program, it was decided to write a new version of this program.

The program was written in Fortran II in order to simplify coding and to allow easy changing of the transformations available in the program. The program is written for 40K 1620 with card I/0 and makes extensive use of the Column Subroutine, No. 1.6.086 in the Users Group library. The Column Subroutine provides the capability of examining any card column of an input card before deciding on which format the input card should be read in. This allowed a simplification of the amount of data required in the header card by providing an easy method to differentiate a transformation card from a data card without indicating in the header card that a transformation was to take place.

The solving of simultaneous equations was done by means of a matrix inversion subroutine which was written in Fortran II and compiled separately from the main program. This was done in order to allow easy replacement of the matrix inversion subroutine if desired. The changing of the transformation by the user, although easy to do in the source program, would require recompiling of the main program.

The program was written to allow only ten variables (one dependent and nine independent) to be processed. This may be increased by increasing the numbers in the Dimension statement and by changing an IF statement at the beginning of the program. However, this program is using all but about 2000 locations in memory and care should be taken so that overflow does not occur. If a 60K memory is available the dimension statements may be increased to take advantage of the additional memory. The only restriction on this point would be the size of the matrix which the matrix inversion subroutine is capable of handling accurately.

## Input

The first input card (the header card) is written in the following format:

Col. 1 (1) An asterisk if the program is to give full output, or
(2) a dollar sign if the program is to inhibit the stepwise regression output and only give the standard statistic output.

Col. 2-3 The number of variables (right justified) included in the regression.

Col. 4-80 Any information desired to identify the data. This information will be reproduced on the first output card.

Transformation cards

Transformation cards may be included if desired. If they are included they are in the following format. One card is used for each transformation desired.

| | |
|---|---|
| Col. 1-2 | The number of the variable to be transformed (right justified) |
| Col. 3-4 | The type of transformation (See paragraph on Description of Transformation.) |
| Col. 5-10 | Information included in these columns is ignored. |
| Col. 11-36 | Transformation constants, if needed, written in either E or F format. Must include decimal point and must be right justified only in the case of E or F format. |
| Col. 37-80 | Not used. |

Data Cards

Data is read in either E10.0 or F10.0 format, a maximum of 8 variables per card. The first variable is the dependent variable. A second card may be used for each set of observations if more than 8 variables are used. If more than one set of data is to be processed, the header card for the next set of data serves as a trailer card for the preceding set of data. If only one run is included, data input is terminated by a last card indication.

## Description of Transformations

1. EXCH, EX, EXCHANGE - to exchange any variable with the dependent variable

Example #1: 02EXCH

Variable number 2 becomes the dependent variable and variable number 1 becomes the independent variable.

Example #2:

```
*05    HEADER CARD

08EXCH         10.
```

According to the header card, 5 variables are included in the regression (1 dependent and 4 independent) however, the variable desired as the dependent variable is listed as variable number 9 on the input card. The original dependent variable will be moved to position 9 and hence be ignored in the regression. The 10. in the transformation constant position indicates that 10 variables are being read in although the header card indicates that only 5 are being processed. The transformation constant must be included whenever the number of variables included is greater than the number listed on the header card and if the additional variables listed requires another card of input data for each set of observations. The transformation constant indicates the total number of variables that are being read in which may be larger than the number actually processed.

2. LOG - The transformation will substitute the log of a given variable for the actual variable.

   Example: 04LOG

   Each observation of variable number 4 is replaced by the log of the observation.

3. LN - This transformation is the same as 2 except the number is replaced by the natural log.

4. SQ, SQR, or SQRT - This transformation replaces the variable with the square root of the variable.

   Example: 04SQ or 04SQR or 04SQRT

   Each observation of variable number four is replaced by its square root.

5. ** - This transformation replaces the variable with the variable to any power.

   Example: 02** .5

   Each observation of variable number 2 is replaced by the square root of the observation.

   01** 2.0

   Each observation of variable number 1 (dependent variable) is replaced by the observation squared.

6. * - This transformation replaces the variable with any number times the variable.

   Example: 04* 3.25

   Each observation of variable number 4 is multiplied by 3.25.

   03* 4.7E-09

   Each observation of variable number 3 is replaced by .000000004 times the observation.

```
C     STEPWISE REGRESSION  K  DANIELS  NORTH DAKOTA STATE UNIVERSITY
      DIMENSION X(10),SUMX(10),PRODX(10,10),VALUE(10),ER(10),IDENT(10),
     1R(10,11),VAR(10),ANS(10),ALPHA(19),SD(10),KTYPE(10),CONST(10),SE(1
     10)
      COMMON VALUE,ER,R
      NUMB=0
      KOD23=1
300   KOUNT=0
      N=NUMB
      IF(SENSE SWITCH 9)7196,7196
C     READ IN THE TOTAL NUMBER OF VARIABLES  10 MAXIMUM, THIS INCLUDES
C     THE DEPENDENT VARIABLE.  THIS NUMBER SHOULD BE IN COLUMNS 2 AND 3
C     COLUMN 1 SHOULD CONTAIN AN * OR $ IF MORE THAN ONE
C     SET OF DATA IS TO BE RUN.  COLUMNS 5 THROUGH 80 SHOULD CONTAIN THE
C     DATE, PROBLEM NUMBER, NAME AND ANY OTHER PERTINANT INFORMATION.
 7196 GO TO(400,4112),KOD23
 400  READ 1,KOL,N,(ALPHA(I),I=1,19)
 1    FORMAT(A1,I2,19A4)
      KOL23=KOL
 4112  KOD23=1
      NN=N
      KOL=KOL23
      NOTR=1
      PUNCH 595,(ALPHA(I),I=1,19)
 595  FORMAT(3X,19A4)
      IF(N-1)731,731,730
730   IF(N-10)732,732,731
731   TYPE=.484459
      GO TO 66
732   K=N-1
      XN=N
C     READ IN THE DATA CONSISTING OF THE DEPENDENT VARIABLE IN COLUMNS
C     1-10 AND THE REMAINING NUMBERS IN SUCCESSIVE COLUMNS IN GROUPS OF
C     10.  GO TO A SECOND CARD IF NECESSARY BUT START EACH NEW DEPENDENT
C     VARIABLE IN COL 1-10 OF THE CARD.
      DO 4 I=1,N
      SUMX(I)=0.0
      DO 4 J=1,N
4     PRODX(I,J)=0.0
C     TRANSFORMATIONS.  TRANSFORMATION CARDS FOLLOW THE HEADER CARD ONE
C     CARD FOR EACH TRANSFORMATION DESIRED.  THE TRANSFORMATIONS ALLOWED
C     ARE LOG, LN, SQRT,VARIABLE TIMES ANY CONSTANT,(*) VARIABLE TO ANY
C     POWER(**) AND EXCHANGE ANY VARIABLE WITH THE DEPENDENT VARIABLE
C     (EX).  THE TRANSFORMATION CARD CONTAINS VARIABLE NUMBER IN COL 1
C     AND 2 THE TYPE OF TRANSF STARTING IN COL 3 AND THE TRANSFORMATION
C     CONSTANT IF ANY STARTING IN COL 10 WRITTEN WITH DECIMAL POINT OR
C     IN E FORM.
      DO 376 I=1,N
376   KTYPE(I)=1
      DO 2015 I=1,10
71    KOL3=COLUMN(3)
      IF(KOL3-3)2016,2016,2018
2018  IF(KOL3-70)2017,2016,2016
 2017  IF(KOL3-10)1017,2016,7017
 7017 IF(KOL3-20)3017,2016,3017
 3017 READ 2099,ID1,KIND,CONST(ID1)
      GO TO(4647,4648),NOTR
 4647 PUNCH 4649
      NOTR=2
 4649 FORMAT(15HTRANSFORMATIONS/)
 4648 PUNCH 2099,ID1,KIND,CONST(ID1)
```

```
2099   FORMAT(I2,A2,7XE25.0)
       IF(KIND-53560)76,5356,76
76     IF(KIND-53550)60,5355,60
60     IF(KIND-62580)61,6258,61
61     IF(KIND-14140)62,1414,62
62     IF(KIND-14000)63,1400,63
63     IF(KIND-45670)64,4567,64
5356   KTYPE(ID1)=2
       GO TO 2015
5355   KTYPE(ID1)=3
       GO TO 2015
6258   KTYPE(ID1)=4
       GO TO 2015
1414   KTYPE(ID1)=5
       GO TO 2015
1400   KTYPE(ID1)=6
       GO TO 2015
4567   KTYPE(ID1)=7
       IF(ID1-N)1018,1018,1019
1019   NN=ID1
1018   IF(CONST(ID1))64,2015,1020
1020   IF(CONST(ID1)-10.)1022,1022,64
1022   IF(CONST(ID1)-1.)2015,64,1025
1025   NN=CONST(ID1)
       IF(NN-N)2015,2015,64
2015   CONTINUE
 366   TYPE=.44416341
       GO TO 66
64     TYPE=.635955
66     PUNCH 65,TYPE
65     FORMAT(10XA4,2X5HERROR,58X1H-)
69     KOL1=COLUMN(1)
       IF(KOL1-14)67,300,67
 67    IF(KOL1-13)677,300,677
 677   READ 68
68     FORMAT(1H )
C      THE PROGRAM WILL STOP READING DATA WHEN A LAST CARD INDICATION IS
C      RECEIVED OR WHEN IT READS A CARD WITH AN  * OR $  IN COLUMN ONE
C      THE CONTENTS OF THIS CARD WILL BE STORED AND USED AS THE HEADER CARD FOR
C      THE NEXT SET OF DATA. THE PROGRAM WILL NOT STOP BETWEEN SETS OF DATA
C      IF PROGRAM SWITCH 1 IS ON
       IF(SENSE SWITCH 9)614,69
 2016  IF(SENSE SWITCH 9)8,7
7      KOL1=COLUMN(1)
       IF(KOL1-14)402,114,402
 402   IF(KOL1-13)444,114,444
 444   READ 9,(X(I),I=1,NN)
9      FORMAT(8E10.0)
       KOUNT=KOUNT+1
C      COMPUTE THE SUMS OF EACH VARIABLE. THE SUM OF THE CROSS PRODUCTS
C      OF EACH PAIR OF VARIABLES AND SUM OF SQUARES.  PRODX(3,4) MEANS
C      SUM OF X(3)*X(4)
       CO TO(8015,8016),NOTR
 8016  DO 72 I=1,N
       KTR=KTYPE(I)
       GO TO (72,1012,1013,1014,1015,1016,1017 ),KTR
1012   X(I)=.43429448*LOG(X(I))
       GO TO 72
1013   X(I)=LOG(X(I))
       GO TO 72
1014   X(I)=X(I)**.5
```

```
      GO TO 72
1015  X(I)=X(I)**CONST(I)
      GO TO 72
1016  X(I)=X(I)*CONST(I)
      GO TO 72
1017  WORK=X(I)
      X(I)=X(1)
      X(1)=WORK
72    CONTINUE
 8015 DO 10 I=1,N
      SUMX(I)=SUMX(I)+X(I)
      DO 10 J=I,N
10    PRODX(I,J)=X(I)*X(J)+PRODX(I,J)
      GO TO 2016
C     PUNCH OUT THE SQUARES AND CROSS PRODUCTS
 114  READ 1,KOL23,NUMB,(ALPHA(I),I=1,19)
      KOD23=2
8     PUNCH 2224,KOUNT
2224  FORMAT(I4,13H OBSERVATIONS)
      PUNCH 2222
 2222 FORMAT(/17HSUMS OF VARIABLES)
      PUNCH 2223,(SUMX(I),I=1,N)
2223  FORMAT(5E16.8)
      PUNCH 98
 98   FORMAT(/26HSQUARES AND CROSS PRODUCTS)
      DO 74 I=1,N
74    PUNCH 41,(PRODX(I,J),J=I,N)
      COUNT=KOUNT
C     COMPUTE AVERAGES
      DO 7777 I=1,N
7777  SD(I)=SUMX(I)/COUNT
      PUNCH 8888
 8888 FORMAT(/8HAVERAGES)
      PUNCH 2223,(SD(I),I=1,N)
C     COMPUTE VARIATIONS
      DO 2225 I=1,N
      DO 2225 J=I,N
2225  PRODX(I,J)=PRODX(I,J)-SUMX(I)*SUMX(J)/COUNT
      DO 2226 I=1,N
      VAR(I)=PRODX(1,I)
2226  SD(I)=PRODX(I,I)**.5
      PUNCH 2228
2228  FORMAT(/18HSTANDARD DEVIATION)
      PUNCH 2227,(SD(I),I=1,N)
2227  FORMAT(5E16.8)
      PUNCH 2229
2229  FORMAT(/35HRESIDUAL SQUARES AND CROSS PRODUCTS)
      DO 2230 I=1,N
2230  PUNCH 41,(PRODX(I,J),J=I,N)
      PUNCH 2764
2764  FORMAT(/12HCORRELATIONS)
      DO 2231 I=1,K
      IX=I+1
      DO 2231 J=IX,N
      PRODX(I,J)=PRODX(I,J)/(SD(I)*SD(J))
2231  PRODX(J,I)=PRODX(I,J)
      DO 2232 I=1,N
2232  PRODX(I,I)=1.0
      DO 2233 I=1,N
2233  PUNCH 41,(PRODX(I,J),J=I,N)
      STERR=SD(1)/(COUNT-1.)**.5
      PUNCH 2234,STERR
 2234 FORMAT(/21HSTD ERR OF DEP VAR = ,E14.8)
      IF(KOL-13000)613,3131,613
C     COMPUTE THE COEFFICIENTS FOR THE SIMPLE EQUATIONS AND THEIR F
C     VALUE THEN PUNCH OUT THE EQ WITH THE HIGHEST F VALUE.
C     THEN PROCEED BY BUILDING ONTO THE BEST.
 613  IDENT(1)=1
      DO 617 I=2,N
  617 IDENT(I)=0
      DO 136 KK=2,N
      K1=KK-1
      EXPL=0.0
      DO 126 I=2,N
      IDENT(KK)=I
C     CHECK TO SEE IF VARIABLE NUMBER I HAS ALREADY BEEN USED
      IF(K1-1)615,615,616
 616  DO 87 JJ=2,K1
      IF(IDENT(JJ)-I)87,126,87
 87   CONTINUE
 615  DO 815M=2,KK
      IDN=IDENT(M)
      ER(M-1)=PRODX(1,IDN)
      DO815 MM=M,KK
      IDK=IDENT(MM)
      R(M-1,MM-1)=PRODX(IDK,IDN)
 815  R(MM-1,M-1)=PRODX(IDK,IDN)
      IF(K1-1)819,819,820
 820  CALL SOLUTN(R,ER,K1,VALUE,KERR)
      GO TO (305,300),KERR
 819  VALUE(1)=ER(1)
 305  DO 909 L=2,KK
      IDN1=IDENT(L)
 909  VALUE(L-1)=VALUE(L-1)*SD(1)/SD(IDN1)
C     COMPUTE THE EXPLAINED VARIATION AND COMPARE IT WITH THE LARGEST OF
C     THE PREVIOUS EXPLAINED VARIATION FOR THE SAME NUMBER OF VARIABLES
      XPL=0.0
      DO 97 IZ=2,KK
      IDN=IDENT(IZ)
 97   XPL=VALUE(IZ-1)*VAR(IDN)+XPL
      IF(EXPL-XPL)101,126,126
 101  EXPL=XPL
      ID=I
      DO 77 I2=1,K1
      SE(I2)=R(I2,I2)
 77   ANS(I2)=VALUE(I2)
 126  CONTINUE
      UNEX=VAR(1)-EXPL
      IDENT(KK)=ID
      COEF=0.0
      XKK=KK
      STERR=(UNEX/(COUNT-XKK))**.5
      DO 8866 L=1,K1
      IDN=IDENT(L+1)
      SE(L)=STERR/SD(IDN)*SE(L)**.5
 8866 COEF=SUMX(IDN)*ANS(L)+COEF
      COEF=(SUMX(1)-COEF)/COUNT
C     COMPUTE F AND PUNCH OUT F AND THE COEFFICIENTS OF THE BEST
C     EQUATION WITH THE GIVEN NUMBER OF UNKNOWNS
      F=EXPL*(COUNT-XKK)/((XKK-1.)*(UNEX))
      IF(KK-2)201,201,200
  200 F1=(EXPL-XPLN)*(COUNT-XKK)/UNEX
```

```
201     XPLN=EXPL
        PUNCH 1802,K1
1802    FORMAT(//10HSTEP NO.  I2)
        PUNCH 2234,STERR
        PUNCH 1002,COEF
        PUNCH 1001
        PUNCH 17,(IDENT(L+1),ANS(L),SE(L),L=1,K1)
        IF(KK-2)202,202,203
203     PUNCH 209,F,F1
1001    FORMAT(/10X11HVARIABLE NO,15X4HCOEF15X17HSTD ERROR OF COEF)
209     FORMAT(3HF= ,E15.8,5X15HIMPROVEMENT F= ,E15.8)
        GO TO 136
202     PUNCH 208,F
208     FORMAT(3HF= E15.8)
136     CONTINUE
1002    FORMAT(8HCONSTANT,5XE14.8)
17      FORMAT(15XI2,13XE14.8,15XE14.8)
3131    PUNCH 249
 249     FORMAT(79X1H-)
2830    IF(SENSE SWITCH 1)300,614
614     PAUSE
        GO TO 300
 41     FORMAT(5E16.8)
        END
```

```
C      SUBROUTINE FOR SOLVING SIMULTANEOUS EQUATIONS BY MATRIX INVERSION
C      METHOD  10 UNKNOWNS MAXIMUM
       SUBROUTINE SOLUTN(R,ER,K,VALUE,KERR)
       KERR=1
       DIMENSION R(10,11),ER(10),VALUE(10)
       COMMON VALUE,ER,R
       KX=K+1
       DO 33 I=1,K
       DO 48 L=1,K
48     R(L,KX)=0.0
       R(I,KX)=1.0
       IF(R(I,1))34,101,34
34     T1=R(I,1)
       DO 35 J=1,KX
       TR=R(I,J)/T1
35     R(I,J)=TR
       IX=0
       IF(I-K)37,38,101
38     MX=I-1
       MY=1
       GO TO 39
37     MY=I+1
       MX=K
 39    DO 40 L=MY,MX
       IX=IX+1
       T1=R(L,1)
       DO 40 J=1,KX
       TR=R(I,J)*T1
40     R(L,J)=R(L,J)-TR
       IF(I-1)101,44,45
 45    IF((K-1)-IX)101,44,38
44     DO 46 L=1,K
       DO 46 J=1,K
       NU=J+1
46     R(L,J)=R(L,NU)
33     CONTINUE
       DO 54 I=1,K
53     VALUE(I)=0.0
       DO 54 J=1,K
       T1=R(I,J)*ER(J)
54     VALUE(I)=VALUE(I)+T1
       RETURN
101    KERR=2
       RETURN
       END
```

```
* 3     SAMPLE DATA
 64.       57.       8.
 71.       59.       10.
 53.       49.       6.
 67.       62.       11.
 55.       51.       8.
 58.       50.       7.
 77.       55.       10.
 57.       48.       9.
 56.       52.       10.
 51.       42.       6.
 76.       61.       12.
 68.       57.       9.
        SAMPLE DATA
  12 OBSERVATIONS

SUMS OF VARIABLES
   .75300000E+03   .64300000E+03   .10600000E+03

SQUARES AND CROSS PRODUCTS
   .48139000E+05   .40830000E+05   .67960000E+04
   .34843000E+05   .57790000E+04
   .97600000E+03

AVERAGES
   .62750000E+02   .53583333E+02   .88333333E+01

STANDARD DEVIATION
   .29803522E+02   .19720977E+02   .62981478E+01

RESIDUAL SQUARES AND CROSS PRODUCTS
   .88825000E+03   .48175000E+03   .14450000E+03
   .38891700E+03   .99166700E+02
   .39666670E+02

CORRELATIONS
   .10000000E+01   .81964483E+00   .76981687E+00
   .10000000E+01   .79840751E+00
   .10000000E+01

STD ERR OF DEP VAR =   .89861005E+01


STEP NO.   1

STD ERR OF DEP VAR =   .53991481E+01
CONSTANT     -.36234766E+01

          VARIABLE NO              COEF               STD ERROR OF COEF
                2             .12386963E+01               .27377690E+0
F=   .20470855E+02


STEP NO.   2

STD ERR OF DEP VAR =   .53632176E+01
CONSTANT      .36512741E+01

          VARIABLE NO              COEF               STD ERROR OF COEF
                2             .85460836E+00               .45166423E+0
                3             .15063353E+01               .14142665E+0
F=   .10940248E+02      IMPROVEMENT F=   .11344373E+01
```

## Mathematical Discussion

The following is a list of the output available along with some of the formulas used

K = number of observations

N = number of dependent plus independent variables

Sums of variables

Averages

Sums of squares and cross products

Residual sums of squares and cross products

$$\sum x_i x_j = \sum X_i X_j - \frac{\sum X_i \cdot \sum X_j}{K}$$

Standard deviation

$$\sigma_i = \sqrt{\sum x_i^2}$$

Coefficient of correlations

$$r_{ij} = \frac{\sum x_i x_j}{\sqrt{\sum x_i^2 \cdot \sum x_j^2}}$$

Standard error of the dependent variable

$$S_1 = \frac{\sigma_1}{\sqrt{K-1}}$$

The following calculations are repeated for each entering variable at each step.

The partial regression coefficients $b_{1,i}$ are computed by inverting the matrix of the correlation coefficients included in the regression and multiplying this by the matrix consisting of the corre.ation coefficients of the dependent variable and each independent variable included in the regression.

thus $\quad b_2 = b_{1,2} \frac{\sigma_1}{\sigma_2}$

and $$b_i = b_{1,i} \cdot \frac{\sigma_1}{\sigma_i} \qquad i \geq 2 \qquad i \in {}^{+}I$$

Constant

$$b_1 = \bar{X}_1 - \sum b_i \bar{X}_i \qquad i \geq 2 \qquad i \in {}^{+}I$$

The explained variation for step i is computed as follows

$$V_i = b_i \sum_{i=2}^{N} x_i^2.$$

The unexplained variation for step i is given by

$$U_i = \sum x_1^2 - V$$

Standard error of the dependent variable at each step

$$s = \sqrt{U_i/N-M}$$ M = the number of variables in the regression at the given step.

Of the variables remaining at each step only the variable providing the largest explained variation is included in the regression.

The following calculations are made once for each step.

Standard error of the regression coefficients

$$s_{bi} = \frac{s}{\sigma_i} \sqrt{Q_{ii}}$$

$Q_{ii}$ = the ith main diagonal element of the inverted matrix of the correlation coefficients.

Fisher F test

$$F = \frac{V_i/(M-1)}{U_i/(K-M)}$$

$F_i$ = improvement F for the entering variable

$$F_1 = \frac{V_{i+1} - V_i}{U_{i+1}/(K-M)}$$

All output from the program is in the form of cards. If an error should appear in the header card any transformation card or in a data card, a card will be punched out indicating the type of error and then the remaining cards will be read in at full reader speed until the header card for the next batch of data is encountered. The last output card for each run will contain a minus sign in card column 80. This may be used to cause the IBM 407 to start a new page.

In order to save as much space in memory as possible the same array was used in storing averages and standard deviation. Also the same array was used to store the sums of squares and cross products, residual sums of squares and cross products and correlation coefficients. In addition, a common area was set up for some of the arrays used in the subroutines and the main program. The calculation time for the program is approximately 5 minutes for five variables and about 20 minutes for 10 variables. This does not include the time needed to read the input data.

## Bibliography

Anthony Ralston and Herbert S. Wilf, Mathematical Methods for Digital Computers, Wiley Publishing Co., 1960.

Murray R. Spiegel, Theory and Problems of Statistics, Schaum Publishing Co., 1961.

# A NEW APPROACH TO INVESTMENT ANALYSIS

There are three commonly used approaches to investment analysis:

1) The Accounting Method

2) The Discounted Cash Flow Method

3) The Present Value Method

These three approaches meet most of the standards which can be set for a complete and logical approach to investment analysis. However, they are lacking in two areas:

1) The treatment of the Cost of Money, which is unique for each approach and causes the three methods to rank projects in different relative orders.

2) The nature of the final result or index figure generated, which causes the absolute values computed to differ sharply from method to method.

The Equivalent Annual Amount (EAA) method offers the needed improvement. It permits the user to select that Cost of Money which will correctly reflect the financial setting of a project. EAA further expresses its answer in terms of the Rate of Return which is the most commonly used and most widely understood measure of economic performance.

The basis of the EAA approach is the "Equivalent Annual Amount", a value which may be understood as the time adjusted average of a series of values. Mathematically the EAA can be expressed as:

$$EAA = \frac{i(1+i)^n}{(1+i)^n - 1} \sum_{j=1}^{j=n} R_j \frac{1}{(1+i)^j}$$

where n = life of the project
i = cost of capital
Rj = values in the time series to be averaged.

To calculate the EAA Rate of Return, it is first necessary to calculate the EAA for the projected time series describing Net Income, Depreciation, and Unrecovered Investment Balance. The EAAs are then related to each other as if they were the values of an income statement to express the EAA Rate of Return.

An analysis of the mathematical realtionships between the different methods reveals that EAA ranks projects basically in agreement with the Present Value method. The superiority of this ranking pattern has been discussed in several publications.

The Rate of Return as computed by the EAA method is identical to the Accounting Rate of Return when the cost of money is selected to be 0%. The Rate of Return as computed by the EAA method is identical to the Discounted Cash Flow answer when the cost of money is selected to equal the Project Rate of Return. This means EAA is not in disagreement with the methods now in use but simply adds a new dimension of flexibility by giving the user the choice of selecting the Cost of Money.

EAA thus proves to be a concept on a level one step above the Accounting Method and the Discounted Cash Flow Method, because EAA can describe the general investment analysis model and can express the Accounting Method and the Discounted Cash Flow Method as special cases within the general framework. EAA could well be the first step to a unified and comprehensive theory of investment analysis.

For a more detailed description of the EAA system, refer to "The Equivalent Annual Amount Method - A New Approach to Investment Analysis" by L. C. Raney, K. A. Rist, and H. A. Wiebe, NAA Bulletin, April 1965, Section 1.

K. A. Rist

# A RELOCATABLE SPS SUBROUTINE FOR EDITING AND ROUNDING OUTPUT DATA FOR SCIENTIFIC TABLES AND SIMILAR APPLICATIONS

W. N. Tuttle, General Radio Company

## ABSTRACT

The need is discussed for a general-purpose subroutine for fitting an output quantity to an allocated space and rounding to the required number of digits. An SPS subroutine is described which gives unbiased rounding and shifts the decimal point to handle as wide a range of values as possible without resorting to exponential format. The subroutine is relocatable and is added to the regular fixed-length subroutine deck.

# A RELOCATABLE SPS SUBROUTINE FOR EDITING AND ROUNDING OUTPUT DATA FOR SCIENTIFIC TABLES AND SIMILAR APPLICATIONS

W. N. Tuttle, General Radio Company

## Introduction: The Need for a General-Purpose Subroutine for Editing Output Data

It seems to me that it is no longer enough for a computer to give us the right answer. It should put the answer in the form we want it in, either for transmittal to a non-programming reader, or for publication. The form of the output should, without editing, meet the standards of the scientists and of the professional journals. I think that by now we should be able to publish a book of tables directly from the computer output and avoid the danger of introducing errors in a separate editorial process.

Let's look at some of the problems we run into when we make a table of computed quantities and don't want to resort to exponents. One of the most bothersome is that we have to allow more space than we like, either because of uncertainty about the range that will be required, or because of the limitations of the available formats. If we use the Fortran conversion specification 10F4 we would get, over the range of exponents that can be covered,

```
    .0000
    .0001
    .0012
    .0123
    .1234
   1.2345
  12.3456
 123.4567
1234.5678
```

As a first improvement we can make it possible to maintain precision with small values by shifting the decimal point to the left when there are leading zeros. This keeps the same number of significant figures until the available width is used up,

```
.00001234
 .0001234
  .001234
   .01234
    .1234
   1.2345
  12.3456
 123.4567
1234.5678
```

We can make a second improvement by reducing the number of decimal places for large numbers to keep within the specified width. This means shifting the decimal point to the right. This avoids excess figures and greatly extends the range of values that can be covered.

```
.000000000
.000000001
.000000012
.000000123
.000001235
.000012346
 .00012346
  .0012346
   .012346
    .12346
   1.23457
  12.34568
 123.45678
1234.56780
12345.6780
123456.780
1234567.80
12345678.0
123456780.
        OV
```

This shows the range of outputs when both left and right shifts are used together. Here the nominal number of decimal places is 5. For small numbers the left shift keeps 5 significant figures until the width is used up and then a constant number of decimal places until the field is all zeros. For large numbers a constant number of decimal places is kept as long as possible, and then the decimal point is shifted to the right until the decimal point occupies the last position in the output area. Note the enormous range of values that can be covered.

A particularly useful specification is available in the common case where the left shift is not needed. If, for example, four places is enough for small numbers and four significant figures for large numbers, we can use the right shift only and allow a width of only 5 positions. This gives

```
.0000
.0001
.0012
.0123
.1234
1.234
12.34
123.4
1234.
```

This specification is particularly useful when many columns of data are required on a page and full use must be made of the entire width.

This last example would require a width of 6 positions for negative numbers. A minor feature of the program is that it checks for the sign and allows an extra space for positive numbers. The Fortran F specification does not do this.

For many applications in engineering, science and statistics tables are needed with only 3 or 4 significant figures, but these figures must be accurate, and proper rounding rather than truncation is essential. It is not satisfactory to give more figures than needed and let the user do his own rounding. This wastes the user's time in addition to wasting space.

Many of you have used the rounding system in which 5 is added in the first remainder column. This is not hard to do for a fixed number of retained digits, but it becomes messy when the number of retained digits may vary from 1 to 7 with an 8-digit mantissa, as in a general-purpose output data conversion program. For this reason truncation is the rule in the IBM subroutines.

Since rounding is needed, let's do the job right. The "half-add" rounding procedure tends to make the average of the rounded values slightly greater than the average of the unrounded values. All changes are upwards so the method has been called "up-rounding". Small differences in averages are frequently important, particularly in statistics, and rounding should be done in such a way that the rounded values are not biased, either up or down. Unbiased rounding is the same as "up-rounding" except when the remainder is exactly one half. In this case the rounding is up when the next digit is even, but down when the next digit is odd.

| | Original Number | Biased Rounding | Unbiased Rounding |
|---|---|---|---|
| | 0.5 | 1. | 0. |
| | 1.5 | 2. | 2. |
| | 2.5 | 3. | 2. |
| | 3.5 | 4. | 4. |
| | 4.5 | 5. | 4. |
| | 5.5 | 6. | 6. |
| | 6.5 | 7. | 6. |
| | 7.5 | 8. | 8. |
| | 8.5 | 9. | 8. |
| | 9.5 | 10. | 10. |
| Total | 50.0 | 55. | 50. |
| Average | 5.0 | 5.5 | 5.0 |

Here the average with biased rounding is 10 per cent greater than the average of the original numbers. This is an extreme case because the remainder is not usually exactly .5 and because more than one figure is usually retained, but the difference is sometimes significant in practice, and unbiased rounding is desirable both in scientific work and in statistics. In accounting it is a matter of convention, and biased rounding is traditional.

One final capability would be desirable in an output data-conversion subroutine. This is a means for rounding to the nearest integer and omitting the decimal point in the output.

## Description of the Subroutine

A flexible general-purpose SPS II subroutine has been written in which the features described above have been incorporated. The subroutine performs, optionally, under control of a code operand, the desired combination of the following operations:

a. Determining the number of significant figures that can be used with the specified width of the output area. If the number is positive or the decimal point omitted, additional space is made available.

b. Rounding the mantissa to the required number of figures using either conventional "up-rounding" or unbiased rounding.

c. Moving the decimal point to the right when the specified position would cause an overflow.

d. Moving the decimal point to the left when the specified position would result in loss of significant figures because of leading zeros.

e. Alternatively omitting the decimal point and rounding to the nearest integer.

f. Converting the edited value to alphameric form and transmitting to the specified output area.

g. Transmitting "OV" to the output area when the specified conversion causes an overflow.

## Use of the Subroutine

The subroutine is added to the regular fixed-length subroutine deck and a library card is added to the processor. The subroutine is loaded automatically whenever the macro FLA (float-to-alphameric) is used in the program. Five operands are used as follows:

```
FLA A,B,C,D,E
```

where,

A is the address of the alphameric field,
B is the address of the floating-point number,
C is the alphameric width of the output area,
D is the nominal or uncorrected number of decimal places, and
E is the code specifying the type of conversion.

If the code operand is zero, or omitted, then unbiased rounding is used, and both right and left decimal point shifts occur if the specified decimal places cause overflow or loss of significant figures. Thus

```
FLA   OUT1,NMBR1,10,5,0
```

specifies that the output area, OUT1, is of alphameric width 10, the quantity NMBR1 is to be converted and transmitted to OUT1, and that 5 decimal places will normally be used. The code operand is 0, so the decimal point will be shifted as required to accommodate as wide a range of numbers as possible.

The code operand handles other specifications by a figure 1 in the tens, hundreds, or thousands positions. Code 10 causes omission of the left shift of the decimal point when there are leading zeros, code 100 causes omission of the right shift on overflow, and code 1000 calls for conventional "up-rounding" instead of unbiased rounding. The digits can be combined as desired, so that, for example, code 1010 calls for up-rounding with right shift only. Note that the code digit in the units position must always be zero.

The decimal point is omitted by making the decimal point operand negative. (Zero calls for a decimal point with no figures following.) Thus

```
FLA   OUT5,NMBR5,10,-1,
```

calls for omission of the decimal point and rounding to the nearest integer. In this example the code operand is omitted, so is taken as 0. Note that the fourth comma is always required.

## Additional Information

The subroutine is for use with fixed-length 8-digit mantissas only. Considerable rewriting would be necessary to adapt it to variable-length mantissas, because the PICK subroutine, which is used throughout to save storage, is quite different in the variable-length subroutine set. For the same reason rewriting would be necessary for SPS II-D. Two versions are available, one requiring no special features and the other requiring only indirect addressing. The former uses 2145 core positions and the latter 1977. Several approaches were tried in the programming in order to reduce the storage requirements to as low a value as possible with the options provided.

WNT:mao

# CLEARTRAN

Francis W. Winn
COMPUTER LANGUAGE RESEARCH
2501 Cedar Springs Road
Dallas, Texas
Area Code 214 RIverside 7-1621

541

# ABSTRACT

CLEARTRAN is a system for compiling FORTRAN statements to yield an object program having maximum efficiency. The object program generally occupies less than one-half the core space and usually executes twice as fast as the MONITOR II system. Programs involving substantial amounts of subscripted variables may execute in as little as one tenth the usual time.

"Infinite" programs may be compiled by virtue of "instant" linkage from disk and the use of optional advanced language concepts.

"My 1620 can draw circles around your 1620" aptly describes the Format capability. Equations can be "plotted" on the printer. Information can be extracted from a card read or a card may be re-read by any Format number. Complete printer control is available with FORMAT statements. Printing of the results of one problem may be obtained while computing the next set of answers.

Error analyses are exceedingly thorough at both the compile and execution stage. For example, unidentified variables, and out-of-range subscripts are called out at both compile and execute time. The object program seldom blows up during execution. A tract routine is available for presentation of both the name of the variable and its value as calculated.

# CLEARTRAN VERBS

- ACCEPT
- ASSIGN
- BRANCH BACK
- CALL
  - EXIT
  - INTERRUPT
  - LINK
  - PDUMP
- COMMON
- CONTINUE
- DEFINE
  - ADDRESSES
  - DISK
  - DISK ADDRESS
  - FAST LOG
  - SIZE
  - START
- DIMENSION
- DO
- DO BACK
- END
- EQUIVALENCE
- FETCH
- FIND
- FORMAT
- FUNCTION
- GO TO
- HOLD
  - CARD IMAGE
  - ERROR MESSAGE
  - PRINT SKIP
  - PRINT ROUNDING
- IF
- INTEGER
- PAGE
- PAUSE
- PERFORM
- PRINT
- PUNCH
- READ
- REAL
- RECORD
- RELEASE
  - CARD IMAGE
  - ERROR MESSAGE
  - PRINT SKIP
  - PRINT ROUNDING
- RETURN
- ROUTINE
- SET
- STACK
- STOP
- STORE
  - ADDRESSES
  - CONSTANTS
  - NAMES
- SUBROUTINE
- TAG
- TRACE
  - PRINT, TYPE, PUNCH
  - OFF
- TYPE
- ZIP

# IN-LINE ROUTINE

A group of FORTRAN statements prefaced by ROUTINE is defined as an in-line routine. The routine is given a name with up to six alphanumeric characteristics, the first of which must be alphabetic. A routine is normally entered by means of the PERFORM command and the normal exit is by BRANCH BACK.

The PERFORM command generates a BTM (branch and transmit immediate) type of instruction, with the return address being carried to the routine for use when a BRANCH BACK instruction is encountered. If the PERFORM command includes a statement number, the BRANCH BACK will be to the address of the statement number specified; otherwise, the return address will be that of the statement following PERFORM.

The data and variables used in a routine are identical to those of a mainline program. A routine may be located anywhere in the program except within the confines of another routine. The normal exit from a routine is by the BRANCH BACK command, of which several may be used if desired. A direct entry to any numbered statement of the routine may be used. In this event, the BRANCH BACK exit address from the routine will be that specified by the PERFORM command last used to enter the routine.

The address of a routine may be stored in a subscripted array by the STORE ADDRESSES command. This makes it possible to PERFORM a computed address, e.g., PERFORM RUTEN(J).

While within a routine, one may PERFORM other routines provided the chain of addresses required to return to the mainline program is not broken. If there is a need to break the chain, the address of the ROUTINE where the break is to occur may be saved by including the name of the routine as the third operand of the PERFORM command.

By this means, one may perform a ROUTINE from within the routine itself, if desired. Examples are illustrated in the sample programs.

## DEFINITION OF VARIABLES AND ASSIGNMENT OF ADDRESSES

A variable is "defined" if it is encountered to the left of an equal sign, in a READ, ACCEPT or FETCH statement, or in one of the following: COMMON, DIMENSION, EQUIVALENCE, INTEGER, REAL, STACK.

### DIMENSION

The DIMENSION statement is used to define the size and the number of words in an array. The number of subscripts which may be used is not limited to three. The length of the fields in an array can be made different from normal by placing an intergal number in front of each element of the list. The minimum length field is two. There is no maximum limit to the length of the field; however, the practical limit for use in conjunction with printer commands is 288.

### COMMON

This command is identical to IBM's.

### STACK

The STACK command is the opposite of COMMON; i. e., the variables in the list are assigned sequentially ascending addresses, while those in COMMON have descending addresses. A dimensioned variable can be equivalenced to the first element of a list previously stacked making it possible, thereby, to refer to the list as an array. The STACK command is especially convenient for use in conjunction with the deferred PRINT command described later.

### TAG

If it is desired to determine all the positions in a program where reference to a particular variable is made, the TAG command may be used. Up to five variables may be tagged at one time; e. g.

```
TAG, V, VOICE, A20, I, YOU
```

### REAL

This verb defines a variable as a floating point type even though the initial letter might be I, J, K, L, M or N.

### INTEGER

This command defines a variable as a fixed point type even if the initial letter is other than I, J, K, L, M or N.

## FIELD LENGTH

As with the DIMENSION command, the length of a variable can be changed from the normal length by placing a number immediately after the STACK, REAL or INTEGER commands.

## ALTERNATE DIMENSIONING

The list in the COMMON, STACK, REAL or INTEGER commands can be subscripted as in a DIMENSION statement if the variable is to be dimensioned. (The variable must not then appear in a DIMENSION statement.)

## STORE

The STORE command may be used to store at compile time three types of data: addresses, fixed or floating constants with sign, or alphanumeric data. The name of the field where the data are to be stored in core is the first element of the list; the remaining elements are stored in first and successively higher addresses.

## ASSIGN

This command may be used to move addresses in core at object time.

## DEFINE

The DEFINE ADDRESSES command may be used to specify indirect addresses where the actual address of a subroutine, routine, statement number, constant, etc., may be found. This command is very useful for a communication link between two programs which may be in core simultaneously.

## TRACE

The commands TRACE PRINT, TRACE TYPE, TRACE PUNCH, and TRACE OFF may be used to follow the path of a problem through a program. The TRACE PRINT statement calls in a relocatable subroutine which prints the name of the variable on the lefthand side of the equal sign in each arithmetic statement, together with its numerical value. Three variables and their values are printed on each line. Switch 4 activates the TRACE subroutine at object time.

## CALL INTERRUPT

The CALL INTERRUPT statement results in storage on disk of the core immage of the program and data, together with the address at which the interrupt occurred. The CALL INTERRUPT command may be selected by program calculations or by turning on a programmed sense switch. The program may be restored and execution continued by loading a single INTERRUPT RESTART card. If data remain to be read, the last card read, together with cards not yet read, should be set aside for reloading. Obviously, data stored by RECORD commands may be lost unless the disk pack is set aside.

## PRINTER DUMP

If it is desired to dump the contents of core on the printer during the execution of a program, one may use the following command:

```
CALL PDUMP (N1, N2)
```

N1 and N2 may be absolute addresses or they may be the names of variables. CALL PDUMP pulls in a disk utility program which prints core between the limits specified, with 100 digits per line grouped by tens, with core addresses conveniently shown. Control returns to the program after execution.

## ZIP

The ZIP statement used immediately preceding a DO will speed up the evaluation of arithmetic expressions involving subscripted variables. The conditions where ZIP may be used are:

1. Indexing must be under DO loop control only.
2. Each arithmetic statement must be complete without having to use more than one continuation card.
3. DO's may be nested not more than three deep.
4. The number of subscripted words may not exceed 20.

(The above limits are tentative.)

Example:

```
       ZIP
    11 DO 3 J=1, 10
    12 DO 3 K=3, 5
    13 DO 3 L=1, JIM, 2
     1  A(J) = A(J) + B(J, K, L) +AB(I)
       IF (C(J,K,L) -50. )2, 3, 2
     2  C(J,K,L) = D(J,K+1, L-1) +F +A(J)
     3  CONTINUE
```

Note there are four subscripted "words" in the above, under ZIP control. A(J) is one word, B(J,K,L), C(J,K,L) and D(J,K+1, L-1) are the others. Each word must consist of not more than 12 characters including the two parentheses. AB(I) is not a "word" because its index, I, is not under DO control.

Another ZIP could be used after statement No. 3. The DO's must have numerical starts and numerical increments. The DO's may have variable upper limits; i.e., JIM in statement 13.

## I/O FEATURES

The input or output achieved by execution of conventional I/O FORTRAN statements is identical to that from IBM compilers. Certain additional features are available, as follows:

1. B-TYPE, G-TYPE, and J-TYPE FORMAT
2. ALPHABETIC SUBSTITUTION IN E & F OUTPUT
3. AUTOMATIC E & F ROUNDING
4. AUTOMATIC PAGE SKIP
5. CARD IMAGE REREAD
6. DEFERRED PRINTING
7. FORMAT OVERRIDE
8. NON FORMAT READ
9. PLOTTER SIMULATION
10. PLUS SIGN (+) SPACE IGNORE
11. COMPLETE PRINTER CONTROL

## B-TYPE FORMAT

B-type words (B for Beta) can be read or written with a non-standard length designated by the FORMAT statement. The length may be a single character, B1, or up to 80 characters for reading the entire card, B80, or B144 for printing 144 characters. The variable where such a word is stored normally is dimensioned so as to have its word length equivalent to that used in the FORMAT. For example B1 words could be read into a dimensioned variable with a word length of at least 2. B80 words should be read into fields which are at least 160 digits long.

A single B-type character is stored as two digits in core, the left one of which is flagged. If stored in a standard fixed-point field of four digits, only the two positions to the right side of the four are used. When printing or punching such a word, by a B-type FORMAT which is identical in length to that used to read the word, conventional output is obtained. However, if the length designated by the FORMAT statement used to read the word is less than the length used in output, the character will be positioned incorrectly by the difference in the two lengths. This can be corrected by changing the output FORMAT statement so that the two lengths are equal, using X-type FORMAT to make up the differences where required for spacing purposes.

## G-TYPE FORMAT

This is identical to F-type FORMAT except that the first blank following the field is considered as a decimal. G-type FORMAT is normally used for reading only.

## J-TYPE FORMAT

This is identical to I-type FORMAT except that the first blank to the right of the field terminates reading. This permits left-justified, fixed-point fields.

## ALPHABETIC SUBSTITUTION IN E & F OUTPUT

On occasions it may be desirable to substitute a short word or blanks in place of an E or F output field. This can be done by setting the floating-point variable to be listed equal to a special alphabetic field. The two leftmost characters of the alphabetic field must be the decimal point equivalent (03). The decimal point is not printed but the alphabetic characters which represent the remaining portion of the word will be printed. As an illustration,

```
        IF (A) 2, 1, 2
    1   A = WORD(1)
    2   PRINT 100, A
  100   FORMAT (F10. 0)
        STORE NAMES (WORD(1), . NONE, .    , . ALL )
```

will cause the word NONE to be printed if A were zero at the IF statement. Blanks would be printed if A were set equal to WORD(2) and the word ALL would be printed if set equal to WORD(3).

## AUTOMATIC E & F ROUNDING

Rounding of E and F output is automatic. If automatic rounding is not desired, it can be bypassed by the command: HOLD ROUNDING; and restored later if desired by: RELEASE ROUNDING.

## AUTOMATIC PAGE SKIP

A skip to a new page is automatic when the bottom line of the page is sensed (printer indicator 34). This feature can be eliminated by the command: HOLD SKIP (and restored by RELEASE SKIP). This might be desirable when using plotter simulation, or when page skip is under program control.

## CARD IMAGE REREAD

The usual READ statement with a FORMAT number will cause a new card to be read and data extracted therefrom in accordance with the FORMAT specifications. It is possible to "read" the same card again, using different FORMAT statements if desired, by two different methods:

1. Place an X (for extra) after the FORMAT number in the READ statement.
2. Use the command, HOLD CARD IMAGE, followed by a conventional READ statement.

The latter causes a one-time skip of the normal procedure whereby a new card is read for each READ statement, making it possible to reread the last read card. The HOLD CARD IMAGE command can be nullified by the command RELEASE CARD IMAGE.

## DEFERRED PRINTING

Some problems require that all or almost all of the calculations be completed prior to doing any output. With CLEARTRAN it is possible to print the results of one set of calculations while calculating the following set. By this means, printing and calculations can go on simultaneously with a considerable savings in time. Deferred output can be obtained by placing the letter "S" (for Save) after the FORMAT number of an output statement; e.g., PRINT 102S, List. This command will be ignored at object time prior to execution of a SAVE command.

The output commands utilizing this feature can be placed at selected positions in the mainline of the program where recycling does not occur. Alternately, all PRINT S commands can be placed in a ROUTINE using a computed GO TO to execute successive statements. The ROUTINE could be executed by randomly placed PERFORM statements. When used in a ROUTINE, the "printer busy" indicator should be tested to save time (if the printer is busy an immediate exit from the ROUTINE should be made.)

The SAVE (V1, V2) commands result in a transfer of that portion of core image lying between the address of variable V1 and variable V2 to a safe place in memory. This includes V1, but not V2. The variables to be listed by S type output commands should be in contiguous memory locations for the least space requirements. The STACK or COMMON commands are used to achieve the desired order. All the variables to be listed should be in either COMMON or STACK, but not part in one and part in the other.

The deferred output command can be made to list current values if a zero is used in a SAVE statement; i.e., SAVE (0). The original status can be restored by using a negative number in a SAVE statement; e.g., SAVE (-1).

## FORMAT OVERRIDE

The list of an I/O statement is under control of the specifications set up in the FORMAT statement. This normally requires that the number of items in a subscripted list be identical to the "repeat" number of the corresponding element in the FORMAT statement.

With CLEARTRAN, an I/O list may involve subscripts using a variable index. The corresponding "repeat" number of the FORMAT specification should be greater than the maximum possible value of the index(99 is tops). If the repeat number happens to be less than the index variable, FORMAT control will pass to the next element of the FORMAT statement.

## NON FORMAT READ

The preparation of input data to be read under FORMAT control requires extra care to insure proper positioning of data on the punched card. This problem can be circumvented by using the READ statement without a FORMAT number. Data of the E, F, I and A type can be read without a FORMAT number. One or several spaces are used to separate data fields. All 80 columns of a card may be used. A relocatable library subroutine examines the input data and discriminates between E, F, I or A data. The F-type conversion results from a decimal point in a numeric data field. The E-type results if a decimal point and the letter E are found. The I-type is generated when there is no decimal point in a numeric field. The A-type is obtained if none of the above conditions are met.

An "input error" is called out if the variable being read is in the wrong mode. The unread portion of the card is typed and a BRANCH TO the program starting address occurs.

One card may contain information which is read by several READ statements. After all the fields on a card are read, the next item on a list will cause a new card to be read, even if the item is in the middle of a list. A record mark in column one of a card calls EXIT.

## PLOTTER SIMULATION

The SET command is identical to the PRINT command with the exception that printing does not occur. The SET command is normally used to build an image which is to be held in position for additional modifications. If an X follows the FORMAT number of a SET command, "extra" information can be placed in the image without destruction of information previously placed (except that which is overlaid). By this means, it is possible to build up a complex line of information which is to be printed after all the information is in place.

The X specification in a FORMAT statement is used to position or space adjacent fields. In CLEARTRAN one may use the X specification followed by a fixed point variable in parenthesis; e. g., X(N1). This specification will result in a number of spaces equal to the value of the fixed point variable N1.

One may use the space supress character (+) in column one to print one set of characters on top of another. This character should be erased (1H0) prior to the final PRINT command.

After a line of data is in position, it is printed by a PRINT command, the FORMAT number of which is followed by the letter X.

The first 80 characters of an image can be punched by placing the letter X after the FORMAT number of punch statement.

A program to illustrate these features is attached. This program plots three equations, coordinate grids, and prints alphabetic information simultaneously. Another program "draws" a picture of a heat exchanger tubesheet.

## PLUS SIGN (+) SPACE IGNORE

It is not necessary to provide space for the plus sign when printing or punching. This makes it possible to put additional information on a card when punching, or to pack E or F fields adjacent to other fields when printing. An error may occur if the E or F fields are negative, since space must be provided for the minus sign.

## COMPLETE PRINTER CONTROL

A complete set of printer controls is available with CLEARTRAN. The following is a list of the printer controls which are achieved by placing a Hollirith character in column one:

| Before Printing | | After Printing | |
|---|---|---|---|
| + | Space supress | | |
| J | one space | S | one space |
| K | two spaces | T | two spaces |
| L | three spaces | | |
| 1 | skip to channel 1 | A | skip to channel 1 |
| 2 | skip to channel 2 | B | skip to channel 2 |
| 3 | skip to channel 3 | C | skip to channel 3 |
| 4 | skip to channel 4 | D | skip to channel 4 |
| 5 | skip to channel 5 | E | skip to channel 5 |
| 6 | skip to channel 6 | F | skip to channel 6 |
| 7 | skip to channel 7 | G | skip to channel 7 |
| 8 | skip to channel 8 | H | skip to channel 8 |
| 9 | skip to channel 9 | I | skip to channel 9 |
| = | skip to channel 11 | | skip to channel 11 |
| @ | skip to channel 12 | ) | skip to channel 12 |

(Any other character may result in a runaway carriage.)

FORMAT statements with multiple slashes which are executed only by PRINT commands are automatically compiled so as to take advantage of fast printer spacing insofar as possible.

The "printer busy" indicator (35) can be sensed by use of the statement: IF (SENSE SWITCH 35) N1, N2. A BRANCH TO statement N1 occurs if the indicator is on (buffer is unavailable for loading); N2 if off (buffer can be loaded).

Similarly, 33, 34 and 25 can be used to sense respectively channel 9, channel 12 , and printer check indicator on the 1443.

# AN APPROACH TO TIME SERIES ANALYSIS

by R. Mennell and J. Turney
Management Research Department
H. P. Hood & Sons
Boston, Mass.

One approach to time series analysis is to state hypotheses and test their validity. This is generally an undirected, subjective method. What is really needed is a method which tells what relationships are significant, the method not to be dependent on the analyst's ability to hypothesize. To a large degree spectral analysis, when carefully constructed and interpreted, provides a way of pointing out relationships within and between time series. This paper discusses time series analysis in general, describes the use of spectral analysis to develop models, describes a new computer program for spectral analysis, and presents the first step in the analysis of an example series.

## Introduction

It is a simple task to propose situations wherein a researcher has available a series of data and wishes to accurately estimate, or forecast, the value of additional terms in the series. Within the firm there is a need to estimate what demand for a product line will be in the coming month for profit planning, what demand for a product will be during the next day or week for production scheduling, and what demand for a product on a route will be for the next day for vehicle loading.

Hopefully, at the least, the available times series is the correct one with which to work.[1] If analysis is confined to the series and a forecasting method is used, the result is a "naive" forecast in the sense that neither causal factors nor even simply correlated factors are included in the model.

The fundamental premise of this paper is that the basic and first step in developing a forecast for a series is to understand the series itself. Analysis should be performed which will lead directly to a model of the series that is "near best". By best is meant a model that removes relationships to the extent that the residuals (differences between the series and estimates of the series) contain no more systematic relationships which can be removed. The residuals are then "white noise" -- a random series; in other words, a time series of uncorrelated random variables with zero means and common variance.[2]

[1] Unfortunately, one may often have only the system's response rather than the prime series; for example, sales or shipments rather than customer demand. In this case the researcher must either find some way to collect the prime series, or establish that the response series is closely enough related to the prime series to be a solid foundation for analysis.

[2] Of course, a polynomial model can always be found which fits the time series arbitrarily well. In time series analysis, we do not go that far. We "fit" only those components of the series that one has a right to; i.e. the components that have relationship and will therefore be helpful in forecasting.

Near best gives expression to the fact that significance (not necessarily in the statistical sense) must be established in a relationship before it is included in the model. Therefore, the near best model gives residuals that are nearly white noise.

Our intention herein is to describe an approach to analyzing time series and present some tools with which to work. Before doing so, we should explore the environment within which time series analysis is useful.

A first thought is that a non-"naive" forecast may be an improvement over a "naive" forecast. By considering other series for inclusion in a model of the prime series, one may be able to achieve a white noise residual with lower variance. For example, product sales in the future may be partially dependent on population, income, and weather. The method used here can be extended to incorporate other series rather than simply the prime series. If the residuals then have lower variance than those resulting from the analysis of the prime series, the new model will give better forecasts. The residuals of the prime series are the standard for comparison.

The analysis of time series leads to a model to be used in developing a forecast. The model is not a forecasting technique itself. Another way to say this is that the model of a time series can be applied very well to the sample of the series already observed. However, when one forecasts, he wishes to estimate terms in the series that have not yet been observed. This fact leads to interesting questions. Since the terms have not been observed, how can one be assured that the causal system itself will not be different? For instance, the product price may change with respect to the competitor's price, or the product may enter a different phase in its life--from a new product to a "mature" product. The price change, if temporary, results in a transient series superimposed on the prime series.

Mechanical methods of forecasting usually accept new terms in a time series to improve the estimates of coefficients in the underlying model. This device automatically includes some ability to respond to changes in the causal system. Even a simple moving average forecast gradually tracks a change in the average level. The forecaster must make a compromise between obtaining quick response to real changes and stability if there is no real change. However, if the model of the time series employed in the forecasting method is incorrect, the forecast cannot be expected to give good results. Forecasting must start with time series analysis to ensure a sound base.

## LIST OF TERMS

| | |
|---|---|
| $\bar{X}$ | Average value of series |
| C | Constant |
| $\omega$ | Frequency |
| v | Lag in number of terms of series |
| $\lambda_v$ | Lag window of the windowed spectrum - also referred to as a kernel |
| m(t) | Mean value function |
| $\bar{f}_t(\omega)$ | Normalized sample spectral density for frequency (or period $\frac{\omega Q}{2}$) |
| N | Number of values in series |
| P | Period measured in number of terms of the series |
| Y(t) | Residuals |
| $\rho(v)$ | Sample Autocorrelation coefficient for lag v |
| R(v) | Sample Autocovariance for lag v |
| Q | Spectral computation number - number of points in interval 0 to $\pi$ of $\omega$ that spectral density is computed for |
| M | Truncation point number of sample autocorrelations used in computing the spectrum |
| x(t) | Value of series at time t |

## Approach to Time Series Analysis:

We have stated that we want to construct a model of the relationships in a time series to the extent they have meaningful existence. The absolute limit of relationship occurs when the residuals are white noise. Practically, the limit is reached when there are no additions to the model which can be demonstrated as significant in making the residuals more white. At that point, all the insight the series itself can provide for our understanding will have been obtained.

The standard model used for time series is

$$x(t) = m(t) + Y(t) \tag{1}$$

where the mean value function (m(t)) is the observed relationship in the data, and therefore

$$m(t) = E\left[x(t)\right] , \tag{2}$$

and the residuals are

$$Y(t) = x(t) - m(t) . \tag{3}$$

We proceed stagewise to improve the model (m(t)) such that Y(t) approaches white noise. A white noise residual is defined by

$$\begin{aligned} E\left[Y(t)\right] &= 0 \\ E\left[Y^2(t)\right] &= \sigma^2 \\ E\left[Y(t)\ Y(t+v)\right] &= 0 \qquad \text{for } v = 0, 1 - - - N \end{aligned} \tag{4}$$

That is, the residuals have 0 mean, variance $\sigma^2$ and no autocovariance (or autocorrelation).

The analysis is based on the assumption that the residuals are always covariance stationary.

$$\mathrm{Cov}\left[Y(t) - m(t),\ Y(t+v) - m(t+v)\right] = R(v) \quad \text{for all values of t and v} \tag{5}$$

That is, only v, and not t, determines the covariance. In practice the series may not be covariance stationary. However, the time series analysis will still provide insight into the series.

Knowing something about the system which generated the series and by plotting and observing the series a model can be proposed, albeit partial and possibly faulty. One approach then is to state hypotheses after observation and test their validity. Our original statement of intent is a rejection of this approach. We want to have a method of analysis which tells us what relationships are significant. We should not need to depend on our ability to hypothesize. To a large degree spectral analysis, when carefully constructed and interpreted, provides a method of digging out the significant relationships for examination.

As a first step we form the sample autocovariance for lag period v

$$R(v) = \frac{1}{N}\sum_{t=1}^{N-v} x(t)\ x\ (t+v) \qquad v = 0,\ 1,\ 2,\ ----\ M \tag{6}$$

and the sample autocorrelation coefficient for lag v

$$(v) = \frac{R(v)}{R(0)} = \frac{\sum_{t=1}^{N-v} x(t)x(t+v)}{\sum_{t=1}^{N-v} x^2(t)} \qquad v = 1,\ 2,\ ----\ M \tag{7}$$

The autocovariance and autocorrelation are analogous to the sums of squares and correlation coefficient respectively. Here, though, the calculations are performed on all pairs of data with a time difference of v periods. When either R(v) or $\rho(v)$ are plotted, relatively high absolute values indicate a relationship between terms t and t+v.

Actually the contribution (power) of each lag (frequency) can be viewed more readily by calculating the normalized windowed sample spectral density (or simply spectral density);

$$\bar{f}_T(\omega) = \frac{1}{2}\pi\rho(0) + \frac{1}{\pi}\sum_{v=1}^{M}\lambda_v\,\rho(v)\cos\frac{vk\pi}{Q} \tag{8}$$

The spectral density is obtained by representing the sample autocorrelation function as the Fourier transform. Then by inversion the spectral density is expressed as a function of the sample autocorrelation. The autocorrelation has been used rather than the autocovariance so that the results obtained from different time series can be portrayed in a standard form. The frequency $\omega$ is equal to $\frac{k}{2Q}$ where k is denotes an interval for computation (k=1, 2, - -, Q).

M is the truncation point of computation and $\lambda_v$ is the lag window; in our case the Parzen Window;

$$\lambda_v = 1\text{-}6\ \left(\frac{v^2}{M}\right) + 6\left|\frac{v}{M}\right|^3\ ;\ \left|\frac{v}{M}\right| \leq 0.5 \tag{9}$$

$$= 2\ \left(1-\left|\frac{v}{M}\right|\right)^3\ ;\ 0.5 \leq \left|\frac{v}{M}\right| \leq 1.0$$

All plots of spectral density are the natural logarithm of 1000 times $\bar{f}_T(\omega)$. For white noise the expected value of $\bar{f}_T(\omega)$ is 5.07.

## Development of Computer Programs

Early in our study of spectral analysis, a search was made for available programs. We could only consider programs which could be run, or adapted to run, on a 20K 1620 with one disk drive. The libraries of larger computer systems were examined, but did not yield programs which seemed analytically correct or mechanically practical. The basis for developing tools had to be adapted from the three spectral analysis programs available in the 1620 Users' Group library. The programs are listed and described in Figure 1.

Only program 6.0.126 could be run without any modification, so it was the first to be evaluated. The autocorrelation coefficient is unbiased and in the traditional statician's form. However, there are several objections to using it for time series analysis. The first objection is that it includes automatic mean subtraction. In general, one would like to perform spectral analysis on the raw data and/or the detrended data. Mean detrending is, of course, only one form of detrending. One may want to subtract any mean value function --- linear, sinusoid, etc. Therefore, detrending should be performed separately; an operation on the data rather than a specific adjustment in the analysis routine.

A second objection to this program is that the divisor N - v is used rather than N. While the object of the use of N - v is to give an unbiased estimate of the true covariance, the more important fact is that the use of N leads to non-negative estimates of the spectral density function. In addition, the use of N gives a positive definite function to be used to estimate a positive definite function and thus allows one to interpret the Fourier transform of the covariance function as a spectral density function. Finally, the "biased" estimate (using the divisor of N) seems[3] to give smaller mean square error than the unbiased estimate.

Program 6.0.151, like 6.0.126, uses the "unbiased" estimate of autocovariance and does not calculate the autocorrelation coefficient. It was felt that the autocorrelation gives important information by itself and should be included. At least it is easy to understand, even though it does not give an altogether clear idea of the frequencies present nor their power. This program, like the other two, does not permit more than one truncation point to be used. It does use a lag window (Blackman and Tukey).[4]

While program 6.0.147 suffers from deficiencies of the order of importance of those in 6.0.151, it seemed easier to work with and modify. The first change removed the mean detrending. The second step was to replace the autocovariance term in the spectral density function by the sample autocorrelation. This change normalized the spectral density so that results of different time series analyses can be compared. The normalization is, of course, scaling by dividing all R(v) by R(0) --- the sample variance.

[3] Reference 1, p. 940

[4] Program 6.0.126 computes spectral density without any lag window. In addition to other deficiencies, this resulted in the program being completely abandoned.

It was also decided to write general purpose routines for plotting to graphically present results as a very useful adjunct to the program. Natural logarithms of the spectral density were plotted for two reasons.[5] First, log plotting magnifies the result where values are small as an aid to interpretation. Second, a confidence band about the log of spectral density is linear. Before taking logarithms the spectral density was multiplied by 1000 so that the graph would have a standard range of 0 to 10.

The variance of estimate of the spectral density function depends on the lag window selected, the sample size, and the truncation point. For a high truncation point the variance is high, but the bias is low, and conversely for a low truncation point. One way to get more information for this situation of conflict is to use several truncation points. In the revised program the density functions for up to three truncation points are calculated and the three functions are plotted together.

The first analysis computed by the revised program was for a test series composed of a pure constant plus a 12 period sine (N=148). The Bartlett Lag Window ($\lambda v=1-\frac{v}{M}$) from the original program was used. With the exception of the lag window, the formulae were those given for the new program 6.0.166 in Figure 1. Figure 2 is a plot of the density functions for this series. The density is computed for each frequency in terms of K, where K is an interval between $\omega=0$ and $\pi$, K=1, 2, ---, Q. Intervals can be converted into period, P=2Q/K, or frequency, $\omega=K/2Q$. The abscissa of K is cumbersome, so it was decided to convert to period. In Figure 2 there is indication of high power at 12 months (2 x 48/8 = 12). The constant shows up as high power at low frequency, K=1 and 2. Most surprising is the systematic oscillation at higher frequencies. For the next analysis the Parzen lag window was used, with the result shown in Figure 3. Now the spurious power at the higher frequency is eliminated. Next the pure constant term was removed from the sine test data. Thus Figure 4 shows the spectral density for a pure sine series with a period of 12. This reference will be helpful for comparison with spectral densities of seasonal monthly data.

Two other standard spectral densities are included for reference. Figure 5 is an analysis of random data, or white noise, and Figure 6 depicts a constant series. For the case of random data the expected value of the spectral density (for all frequencies) is in $(\frac{1000}{2\pi}) = 5.07$, which is in agreement with the test result.

After performing the test analyses just described, which indicated that the program was working properly, attention was directed to improving the computational efficiency. Before performing the summation $\sum_{v=1}^{M} \lambda_v \cos \frac{K\pi v}{MC} \rho(v)$, all $\lambda_v \rho(v)$ are computed once and tabled to use for all K = 1, 2, ---, Q.

[5] Reference 1, p. 941

In addition, before calculating any density points the values of cos $(\frac{\pi}{Q})$ and sin $(\frac{\pi}{Q})$ are determined, then all subsequent cosine terms are computed recursively. The recursion is based on the relations:

$$\cos (A+B) = \cos A \cos B - \sin A \sin B, \text{ and}$$

$$\sin (A+B) = \sin A \cos B + \cos A \sin B,$$

where A is the previous phase angle and B is the constant increment $\frac{\pi}{Q}$. Since the values of A and B terms are known at each interval, the new term A+B can be computed directly. This is much faster than a method which must redefine the cosine term for each interval.

These two features, along with other smaller improvements, resulted in running times 1/3 as long as were required in the original version. Several runs have been made with 141 data points, 48 lag intervals, 48 density points, and truncation points of 12, 24, and 48. For these dimensions, the autocorrelation calculations take five minutes, and the spectral analysis takes fifteen minutes, including plot output. On a model II 1620 with floating point hardware these times could be expected to be reduced again by 2/3, to about two and five munutes respectively.

The separation of the plotting portion into a set of general routines, each of which was designed for maximum efficiency, resulted in very rapid plot output. Other parts of the program, such as computation of inner products, could be similarly separated and improved. Such additional refinements may be implemented if use of the program becomes extensive. The program and plot routines are available from the IBM 1620 Users' Group library as 6.0.166 and 1.6.123 respectively.

## Example Series

A sample of 141 observations x(t) was used in testing of the described approach. The data are monthly product sales, which may be expected to contain both trend and seasonal components. Figure 7 is a plot which includes the raw data (plot character D), the linear trend (plot character T), and the residual (plot character R). The residual is on an enlarged scale relative to the data and trend. Examination of the residual indicates a definite 12 month seasonal component, as expected.

Figure 8 includes the autocovariance and autocorrelation of the data with itself at each of 48 lag periods. Figure 9 lists spectral density for truncations of 12, 24, and 48 at each lag interval. Frequency (F) for each point is computed as F = 2MC/I (in this example F = 96/I), with F = 999. indicating the zero lag term. The frequency thus represents the number of time units per cycle.

Figure 10 is a plot of the spectral densities, with conversions as described earlier. Truncation points of 12, 24, and 48 are plotted as A, B, and C respectively. Numbers at the left are the frequencies with their fractional part truncated. This plot clearly indicates density peaks at frequencies of 12 and 3, and lesser peaks at 6 and 2½. This suggests investigation of a model of the time series taking into account 12 and 3 month prior data. After several stages of model formulation, use of regression analysis to derive coefficients for the model, and spectral analysis of residuals, a satisfactory model with near-white noise residuals was achieved. This model contained sine and cosine terms for a 12 month seasonal component with increasing amplitude, plus weighting factors for 1, 3, and 6 month prior data points.

The use of spectral analysis to develop time series models is still largely empirical. For example, a 12 period density peak could lead to a simple weighting factor for 12 month prior data points, to a sine-coside model with a 12 month period, or to a double sine-cosine model which includes changing amplitude. This does at least narrow the search, and experience in analysis should lead to good individual judgment in particular cases.

References:

1. E. Parzen, "An Approach to Empirical Time Series Analysis", Radio Science Journal of Research, Vol. 68D No. 9, September, 1964.

2. R. G. Brown, "Smoothing, Forecasting, and Prediction", Prentice-Hall, Englewood Cliffs, N. J., 1963.

3. T. Yamane, "Statistics, An Introductory Analysis", Harper and Rowe, New York, 1964.

4. R. M. Stephenson, "Autocorrelation and Spectral Analysis for a 20K 1620", 1620 General Program Library No. 6.0.126.

5. S. R. Kimbleton, "1620 Spectral Analysis Program", 1620 General Program Library No. 6.0.151.

6. W. Lawton and F. Puff, "Autocorrelation and Power Spectrrum", 1620 General Program Library No. 6.0.147.

7. J. E. Turney, "Autocorrelation - Spectral Analysis Program", 1620 General Program Library No. 6.0.166.

8. J. Warshawsky, "Autocorrelation and Power Spectral Density", 1620 Users Group Meeting, September, 1961.

| Program No. | Autocovariance | Autocorrelation Coefficient (Sample Correlation Function) | Spectral Density |
|---|---|---|---|
| 6.0.126<br>in Ralston & Wilf<br>20K Storage | $R(v)=\frac{1}{Nv}\sum_{t=1}^{N-v}[x(t)-x^*][x(t+v)-x^*]$<br>where $X^*=\frac{1}{N-v}\sum_{t=1}^{N-v}x(t)$ | $\rho(v)=\frac{\sum_{t=1}^{N-v}[x(t)-x^*][x(t+v)-x^*]}{\left\{\sum_{t=1}^{N+v}[x(t)-x^*]^2\sum_{t=1}^{N-v}[x(t+v)-x^*]^2\right\}^{\frac{1}{2}}}$<br>where $X^*=\frac{1}{N-v}\sum_{t=1}^{N-v}x(t)$ | $\bar{F}_T(\omega)=\rho(0)+2\sum_{v=1}^{M}\rho(v)\cos\left(\frac{\pi K v}{M}\right)$<br>comment; no lag window, no plot, for one value of M |
| 6.0.147<br>Bartlett Lag Window<br>40K Storage | $R(v)=\frac{1}{N}\sum_{t=1}^{N-v}[x(t)-\bar{x}][x(t+v)-\bar{x}]$<br>where $\bar{x}=\frac{1}{N}\sum_{t=1}^{N}x(t)$ | $\rho(v)=\frac{R(v)}{R(0)}$<br>$=\frac{\sum_{t=1}^{N-v}[x(t)-\bar{x}][x(t+v)-\bar{x}]}{\sum_{t=1}^{N-v}[x(t)-\bar{x}]^2}$<br>where $\bar{x}=\frac{1}{N}\sum_{t=1}^{N}x(t)$ | $f_T(\omega)=\frac{1}{\pi}\left[R(0)+2\sum_{v=1}^{M}\lambda_v R(v)\cos\left(\frac{\pi K v}{M}\right)\right]$<br>$\lambda_v=1-\frac{v}{M}$ , $\omega=\frac{v}{2M}$<br>comment; for one value of M, no plot |
| 6.0.151<br>Blackman & Tuckey Lag Window<br>40K Storage | $R(v)=\frac{1}{N-v}\sum_{t=1}^{N-v}x(t)\,x(t+v)$ | — — — | $f_T(\omega)=.23\,f_{T-1}(\omega)'+.54\,f_T(\omega)'+.23\,f_{T+1}(\omega)'$<br>$f_T(\omega)'=R(0)+2\sum_{v=1}^{M}\cos\left(\frac{\pi K v}{M}\right)R(v)$<br>comment; for one value of M, no $\rho$ (v) calculated, no plot |
| 6.0.166<br>Parzen Lag Window | $R(v)=\frac{1}{N}\sum_{t=1}^{N-v}x(t)\,x(t+v)$ | $\rho(v)=\frac{R(v)}{R(0)}$<br>$=\frac{\sum_{t=1}^{N-v}x(t)\,x(t+v)}{\sum_{t=1}^{N-v}[x(t)]^2}$ | $\bar{F}_T(\omega)=\frac{1}{2\pi}\rho(0)+\frac{1}{\pi}\sum_{v=1}^{M}\lambda_v\cos\left(\frac{\pi K v}{Q}\right)\rho(v)$<br>$\lambda_v=1-6\left(\frac{v}{M}\right)^2+6\left\lvert\frac{v}{M}\right\rvert^3$ , $\left\lvert\frac{v}{M}\right\rvert\le 0.5$<br>$=2\left(1-\left\lvert\frac{v}{M}\right\rvert\right)^3$ , $0.5\le\left\lvert\frac{v}{M}\right\rvert\le 1.0$<br>$\omega=\frac{v}{2Q}$ |

Figure 1.

```
DENSITY PLOT

MIN #   .11498900E-02
MAX #   .45255369E&01



001.              .              .              .       A   BC .              .
002.              .              .        C     .      AB      .              .
003.              .              .           CB .     A        .              .
004.              .              C              .   A          .              .
005.              .              .      CB      . A            .              .
006.              .              .C             B              .              .
007.              .              .             A. BC           .              .
008.              .              .            A .  B    C      .              .
009.              .              .            A . C            .              .
010.              .             C.            B .              .              .
011.              .              .  CB       A  .              .              .
012.              .     C        .         A    .              .              .
013.              .           CB .     A        .              .              .
014.              .C             BA             .              .              .
015.              .      ACB     .              .              .              .
016.            C .              .              .              .              .
017.              .   ACB        .              .              .              .
018.          C   .         B    .              .              .              .
019.              .  C B       A .              .              .              .
020.        C     .            A .              .              .              .
021.              . CB       A   .              .              .              .
022.       C      .     BA       .              .              .              .
023.             AC B            .              .              .              .
024.      C       .              .              .              .              .
025.            ACB              .              .              .              .
026.    C         .   B          .              .              .              .
027.            CB.       A      .              .              .              .
028.   C          .       A      .              .              .              .
029.           CB .      A       .              .              .              .
030.   C          . BA           .              .              .              .
031.          ACB .              .              .              .              .
032.  C           .              .              .              .              .
033.         ACB  .              .              .              .              .
034. C            .B             .              .              .              .
035.         C B  .    A         .              .              .              .
036. C            .     A        .              .              .              .
037.         CB   .   A          .              .              .              .
038. C            BA             .              .              .              .
039.        ACB   .              .              .              .              .
040.C             .              .              .              .              .
041.       ACB    .              .              .              .              .
042.C            BA              .              .              .              .
043.        CB    .  A           .              .              .              .
044.C             .   A          .              .              .              .
045.        CB    .  A           .              .              .              .
046.C            BA              .              .              .              .
047.       ACB    .              .              .              .              .
048.C             .              .              .              .
```

Figure 2.  12 Period Sine, Bartlett Lag Window

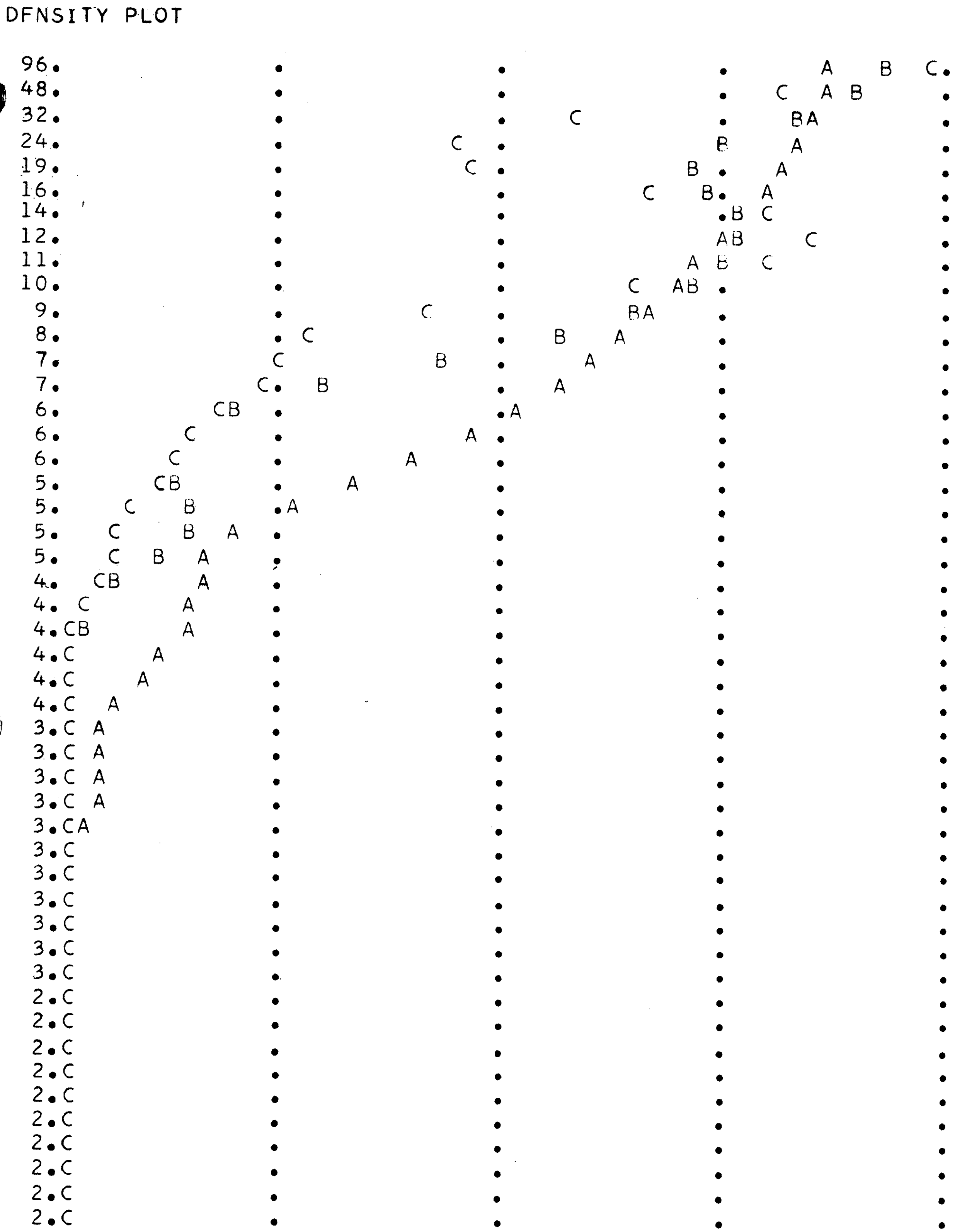


Figure 3. 12 Period Sine, Parzen Lag Window

```
DENSITY PLOT


 PARZEN BETAS

096.              .     C B      .           A  .              .              .
048.              .        C  B  .            A .              .              .
032.              .        C     .    B         A              .              .
024.              .          C   .           B  .A             .              .
019.              .              .   C          .  B           .              .
016.              .              .              .  CA B        .              .
013.              .              .              .   A    B  C  .              .
012.              .              .              .   A    B    C.              .
010.              .              .              .   A    B C   .              .
009.              .              .              . C A B        .              .
008.              .              .C             . BA           .              .
008.              .    C         .          B   .A             .              .
007.              .  C           .  B          A.              .              .
006.              . C    B       .           A  .              .              .
006.         C  B .              .        A     .              .              .
006.     C B      .              .     A        .              .              .
005.    CB        .              . A            .              .              .
005.   C   B      .           A  .              .              .              .
005.C        B    .      A       .              .              .              .
004.C       B     A              .              .              .              .
004.C    B A      .              .              .              .              .
004.C             .              .              .              .              .
004.C             .              .              .              .              .
004.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C A           .              .              .              .              .
003.C  A          .              .              .              .              .
003.C   A         .              .              .              .              .
003.C  A          .              .              .              .              .
002.C A           .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .
                                                                              /1
```

Figure 4. 12 Period Sine, Zero Mean, Parzen Lag Window

```
DENSITY PLOT
096.              .              .         CB   .              .              .
048.              .              .          B  C.              .              .
032.              .              .          B C .              .              .
024.              .              .         AC   .              .              .
019.              .              .       C B    .              .              .
016.              .              .    C   BA    .              .              .
013.              .              .     C  BA    .              .              .
012.              .              .         C    .              .              .
010.              .              .         B C  .              .              .
009.              .              .         AC   .              .              .
008.              .              .         CB   .              .              .
008.              .              .         AC   .              .              .
007.              .              .         AB C .              .              .
006.              .              .         AB C .              .              .
006.              .              .         BC   .              .              .
006.              .              .       CB     .              .              .
005.              .              .     C BA     .              .              .
005.              .              .    C  BA     .              .              .
005.              .              .      BC      .              .              .
004.              .              .      BAC     .              .              .
004.              .              .      BC      .              .              .
004.              .              .     CB       .              .              .
004.              .              .    CB        .              .              .
004.              .              .    CA        .              .              .
003.              .              .    CA        .              .              .
003.              .              .  CBA         .              .              .
003.              .              . C BA         .              .              .
003.              .              .    C         .              .              .
003.              .              .    B C       .              .              .
003.              .              .    BC        .              .              .
003.              .              .   CB         .              .              .
003.              .              . C B          .              .              .
002.              .              . C B          .              .              .
002.              .              .   BAC        .              .              .
002.              .              .   BAC        .              .              .
002.              .              .   CA         .              .              .
002.              .              .C B A         .              .              .
002.              .             C.   B A        .              .              .
002.              .              .C   BA        .              .              .
002.              .              .     CB       .              .              .
002.              .              .      A BC    .              .              .
002.              .              .       AB  C  .              .              .
002.              .              .       AB C   .              .              .
002.              .              .      CAB     .              .              .
002.              .              .      CB      .              .              .
002.              .              .      CB      .              .              .
002.              .              .      CA      .              .              .
002.              .              .      BC      .              .              .
```

/1

Figure 5. "White Noise" (Normally Distributed Random Numbers)

```
DENSITY PLOT

096.              .              .              .        A    B. C            .
048.              .              .              .       CA  B  .              .
032.              .              .      C       .       B      .              .
024.              .            C .              . B   A        .              .
019.              .         C    .        B     .    A         .              .
016.              .        C    B.              . A            .              .
013.              .   C B        .              A              .              .
012.              .C B           .          A   .              .              .
010.              CB             .      A       .              .              .
009.             C.B             .  A           .              .              .
008.           C  . B         A  .              .              .              .
008.         C    . B    A       .              .              .              .
007.        C    B.A             .              .              .              .
006.       C BA   .              .              .              .              .
006.      C A     .              .              .              .              .
006.     CA       .              .              .              .              .
005.    CA        .              .              .              .              .
005.    CA        .              .              .              .              .
005.  C  BA       .              .              .              .              .
004.  C  B A      .              .              .              .              .
004. C B     A    .              .              .              .              .
004. C        A   .              .              .              .              .
004.C         A   .              .              .              .              .
004.C        A    .              .              .              .              .
003.C       A     .              .              .              .              .
003.C     A       .              .              .              .              .
003.C  A          .              .              .              .              .
003.CA            .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
003.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
002.C             .              .              .              .              .
```

**Figure 6. Constant Series**

```
AUTOCORRELATION-POWER SPECTRUM
TEST DATA FOR SPECTRAL ANALYSIS PROGRAM     12/18/64
001.  T            .              .      R       .              .              .
002.  T            .              .    R         .              .              .
003.  T            .              .   R          .              .              .
004.  DT           .              .   R          .              .              .
005.  DT           .              .  R           .              .              .
006.  D T          .              .    R         .              .              .
007.   DT          .              .     R        .              .              .
008.   DT          .              .   R          .              .              .
009.  D  T         .              .R             .              .              .
010.     TD        .              .         R    .              .              .
011.   D  T        .             R.              .              .              .
012.  D   T        .           R  .              .              .              .
013.  D    T       .        R     .              .              .              .
014.   D   T       .          R   .              .              .              .
015.     D  T      .              .R             .              .              .
016.       DT      .              .     R        .              .              .
017.       DT      .              .    R         .              .              .
018.         TD    .              .         R    .              .              .
019.         T D   .              .           R  .              .              .
020.          T    .              .       R      .              .              .
021.        D T    .              . R            .              .              .
022.          DT   .              .      R       .              .              .
023.       D   T   .           R  .              .              .              .
024.       D    T  .         R    .              .              .              .
025.       D    T  .        R     .              .              .              .
026.        D    T .          R   .              .              .              .
027.           D T .              .  R           .              .              .
028.         D   T .        R     .              .              .              .
029.           D  T.              .R             .              .              .
030.            D T.              .R             .              .              .
031.               TD             .          R   .              .              .
032.               T              .        R     .              .              .
033.            D  .T          R .               .              .              .
034.            D  .T         R  .               .              .              .
035.         D     . R            .              .              .              .
036.        D     R. T            .              .              .              .
037.          D    .  R           .              .              .              .
038.            D  .  T   R       .              .              .              .
039.            D  .  T R         .              .              .              .
040.               .D  T          R              .              .              .
041.              D.   T  R       .              .              .              .
042.               .   DT         .    R         .              .              .
043.               .   DT         .    R         .              .              .
044.               .    DT        .     R        .              .              .
045.               .D    T R      .              .              .              .
046.               .D     TR      .              .              .              .
047.               .D    RT       .              .              .              .
048.               DR     T       .              .              .              .
049.               .D  R   T      .              .              .              .
050.               .   D   T  R   .              .              .              .
051.               .R       T     .              .              .              .
052.               .  D R   T     .              .              .              .
053.               .    D    R    .              .              .              .
054.               .     D   TR   .              .              .              .
055.               .          T   .       R      .              .              .
056.               .        D T   .R             .              .              .
057.               .     D  R  T  .              .              .              .
```

Figure 7. Sample Series Data

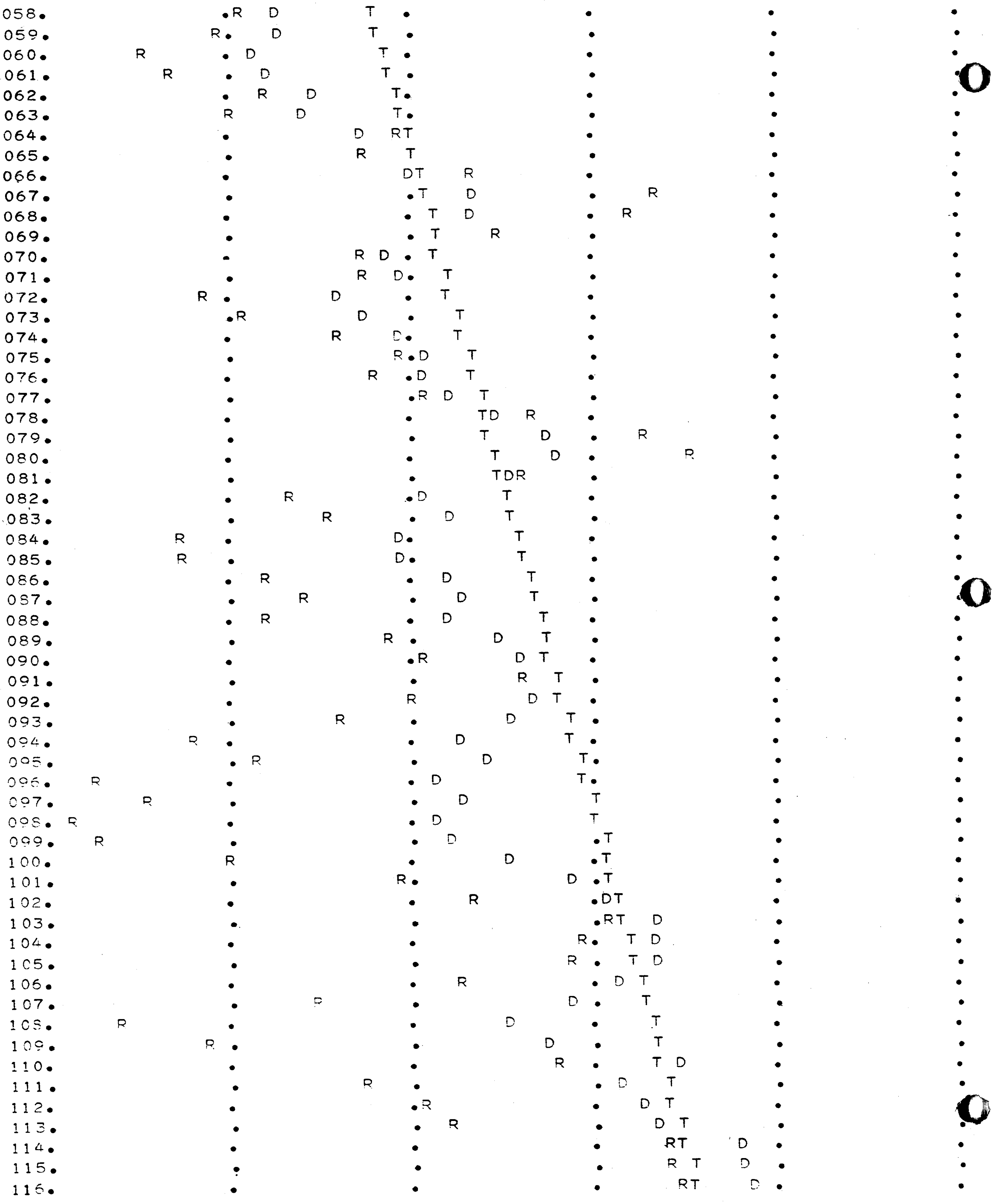


Figure 7. Sample Series Data (continued)

```
117.               .               .             R .         T D   .               .
118.               .      R        .         T     .               .
119.               . R             .       D  T    .               .
120.               .         R     .               .    D    T     .               .
121.               .             R .               .       D  T    .               .
122.               .               .           R   .           TD  .               .
123.               .        R      .               .     D     T   .               .
124.               .               .               R            T D.               .
125.               .               .               .  R         T  .D              .
126.               .               .               .      R      T .  D            .
127.               .               .               .             T .     R  D      .
128.               .               .               .              TR      D        .
129.               .               R               .           D  T.               .
130.               .               .            R  .               TD              .
131.               .               .           R   .               TD              .
132.       R       .               .               .    D          T               .
133.               .               R               .             D .T              .
134.               .               .        R      .               .TD             .
135.               .               .     R         .               .DT             .
136.               .               .               .   R           . T   D         .
137.               .               .               .     R         .  T   D        .
138.               .               .               .   R           .  T . D        .
139.               .               .               .               .   T       R   D
140.               .               .               .               .   T  R       D.
141.               .               .               .               .R  T         D .
```

Figure 7. Sample Series Data (continued)

```
AUTOCORRELATION-POWER SPECTRUM
TEST DATA FOR SPECTRAL ANALYSIS PROGRAM      12/18/64
SAMPLE SIZE #  141
M #    48
          K          PHI%K⌑                  RHO%K⌑
          0       .61918826E&05            1.00000000
          1       .49629587E&05             .80152661
          2       .39625253E&05             .63995484
          3       .30105374E&05             .48620711
          4       .18852726E&05             .30447486
          5       .11471034E&05             .18525922
          6       .10399243E&05             .16794961
          7       .90171652E&04             .14562881
          8       .15269558E&05             .24660606
          9       .26045408E&05             .42063794
         10       .34045049E&05             .54983356
         11       .41392950E&05             .66850346
         12       .46054632E&05             .74379045
         13       .40432239E&05             .65298781
         14       .31725964E&05             .51237993
         15       .22425711E&05             .36217920
         16       .11780641E&05             .19025943
         17       .59452975E&04             .09601760
         18       .28123651E&04             .04542019
         19       .44975980E&04             .07263700
         20       .10222909E&05             .16510178
         21       .19470050E&05             .31444475
         22       .26368626E&05             .42585797
         23       .34144464E&05             .55143913
         24       .38007251E&05             .61382383
         25       .33412656E&05             .53962030
         26       .24516484E&05             .39594555
         27       .15777787E&05             .25481405
         28       .60014975E&04             .09692524
         29       .18969563E&04             .03063618
         30       .12834516E&04             .02072797
         31       .26921270E&04             .04347832
         32       .60884212E&04             .09832908
         33       .15170095E&05             .24499971
         34       .23122131E&05             .37342650
         35       .29949677E&05             .48369258
         36       .32601485E&05             .52651975
         37       .28405066E&05             .45874684
         38       .21030386E&05             .33964445
         39       .13310755E&05             .21497104
         40       .49601110E&04             .08010667
         41      -.18197957E&03            -.00293900
         42      -.16281521E&04            -.02629494
         43      -.25759475E&03            -.00416020
         44       .42860210E&04             .06921999
         45       .10374279E&05             .16754644
         46       .17207375E&05             .27790215
         47       .23431276E&05             .37841925
         48       .25892057E&05             .41816130
```

Figure 8. Sample Series, Autocovariance and Autocorrelation

```
PARZEN BETAS

      F         BETA%1,N¤        BETA%2,N¤        BETA%3,N¤
999.000     .80011024E&00   .13340027E&01   .23465504E&01
 96.000     .78929509E&00   .12202052E&01   .17034202E&01
 48.000     .75812768E&00   .93653168E&00   .67958946E&00
 32.000     .71015539E&00   .61916708E&00   .20900758E&00
 24.000     .65041706E&00   .39948586E&00   .97214880E-01
 19.200     .58440389E&00   .33340729E&00   .88525320E-01
 16.000     .51704732E&00   .38584086E&00   .23442209E&00
 13.714     .45199957E&00   .46911768E&00   .65136868E&00
 12.000     .39135833E&00   .50167397E&00   .93366472E&00
 10.666     .33583873E&00   .44995445E&00   .63677465E&00
  9.600     .28526015E&00   .33467733E&00   .19462367E&00
  8.727     .23913551E&00   .20591381E&00   .35159730E-01
  8.000     .19715263E&00   .10791720E&00   .22684310E-01
  7.384     .15940835E&00   .56990130E-01   .23143450E-01
  6.857     .12636459E&00   .42773160E-01   .29081020E-01
  6.400     .98600030E-01   .44901620E-01   .53479060E-01
  6.000     .76493680E-01   .48369660E-01   .72477750E-01
  5.647     .59980890E-01   .47891110E-01   .55966280E-01
  5.333     .48474860E-01   .44032130E-01   .36976540E-01
  5.052     .40971290E-01   .38747370E-01   .35184100E-01
  4.800     .36282580E-01   .33713860E-01   .29622420E-01
  4.571     .33306580E-01   .30005890E-01   .22685510E-01
  4.363     .31234280E-01   .27654190E-01   .26523880E-01
  4.173     .29635800E-01   .25683580E-01   .31906040E-01
  4.000     .28415290E-01   .23160020E-01   .29076180E-01
  3.840     .27673250E-01   .20261370E-01   .18901550E-01
  3.692     .27539780E-01   .18261970E-01   .96773900E-02
  3.555     .28041400E-01   .18647580E-01   .10495150E-01
  3.428     .29040560E-01   .22253980E-01   .18848840E-01
  3.310     .30254160E-01   .28783770E-01   .24704920E-01
  3.200     .31329000E-01   .36620530E-01   .32181690E-01
  3.096     .31937310E-01   .43126840E-01   .51423100E-01
  3.000     .31856520E-01   .45685250E-01   .65480820E-01
  2.909     .31011180E-01   .43080200E-01   .55028230E-01
  2.823     .29471850E-01   .36240840E-01   .34862590E-01
  2.742     .27420720E-01   .27727820E-01   .21366200E-01
  2.666     .25100700E-01   .20319860E-01   .13447380E-01
  2.594     .22764670E-01   .15661660E-01   .96615800E-02
  2.526     .20635200E-01   .13759420E-01   .10297030E-01
  2.461     .18879240E-01   .13476650E-01   .14642140E-01
  2.400     .17596420E-01   .13569170E-01   .17766000E-01
  2.341     .16817560E-01   .13519190E-01   .14771150E-01
  2.285     .16510480E-01   .13632320E-01   .10184480E-01
  2.232     .16590680E-01   .14454840E-01   .99279900E-02
  2.181     .16936510E-01   .16105920E-01   .13757660E-01
  2.133     .17407460E-01   .18118480E-01   .20514060E-01
  2.086     .17864730E-01   .19835710E-01   .25282660E-01
  2.042     .18191330E-01   .20876450E-01   .21663380E-01
  2.000     .18309310E-01   .21207760E-01   .17402390E-01
```

Figure 9. Sample Series, Spectral Densities

```
DENSITY PLOT


 PARZEN BETAS

096.               .              .              .    A   B C   .              .
048.               .              .              .   CA B       .              .
032.               .              .         C    .  BA          .              .
024.               .              .   C          B   A          .              .
019.               .              .   C         B.  A           .              .
016.               .              .          C   B A            .              .
013.               .              .              .B  C          .              .
012.               .              .              A B   C        .              .
010.               .              .             A.B  C          .              .
009.               .              .         C A B.              .              .
008.               .           C  .         BA   .              .              .
008.               .       C      .    B    A    .              .              .
007.               .       C      B       A      .              .              .
006.               .         C  B .     A        .              .              .
006.               .            B C   A          .              .              .
006.               .             B. C            .              .              .
005.               .             BCA             .              .              .
005.               .           CBA.              .              .              .
005.               .           CA .              .              .              .
004.               .         CBA  .              .              .              .
004.               .       C BA   .              .              .              .
004.               .        CBA   .              .              .              .
004.               .        BAC   .              .              .
004.               .       B C    .              .              .
003.               .      C  A    .              .              .              .
003.               . C    B  A    .              .              .              .
003.               . C    B  A    .              .              .              .
003.               .      CB A    .              .              .              .
003.               .        CB    .              .              .              .
003.               .          CB  .              .              .              .
003.               .          A BC.              .              .              .
003.               .          A  B.C             .              .              .
002.               .          A B C              .              .              .
002.               .         ACB  .              .              .              .
002.               .       C B    .              .              .              .
002.               .   C  B A     .              .              .              .
002.               . C  B  A      .              .              .              .
002.               . C B   A      .              .              .              .
002.               .   BC A       .              .              .              .
002.               .   B C        .              .              .              .
002.               .   BCA        .              .              .              .
002.               . C B A        .              .              .              .
002.               . C  BA        .              .              .              .
002.               .   C B        .              .              .              .
002.               .     AC       .              .              .              .
002.               .     AB C     .              .              .              .
002.               .      AC      .              .              .              .
002.               .     CAB      .              .              .              .
```

Figure 10. Sample Series, Spectral Density Plot