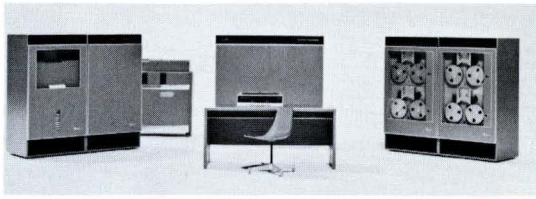
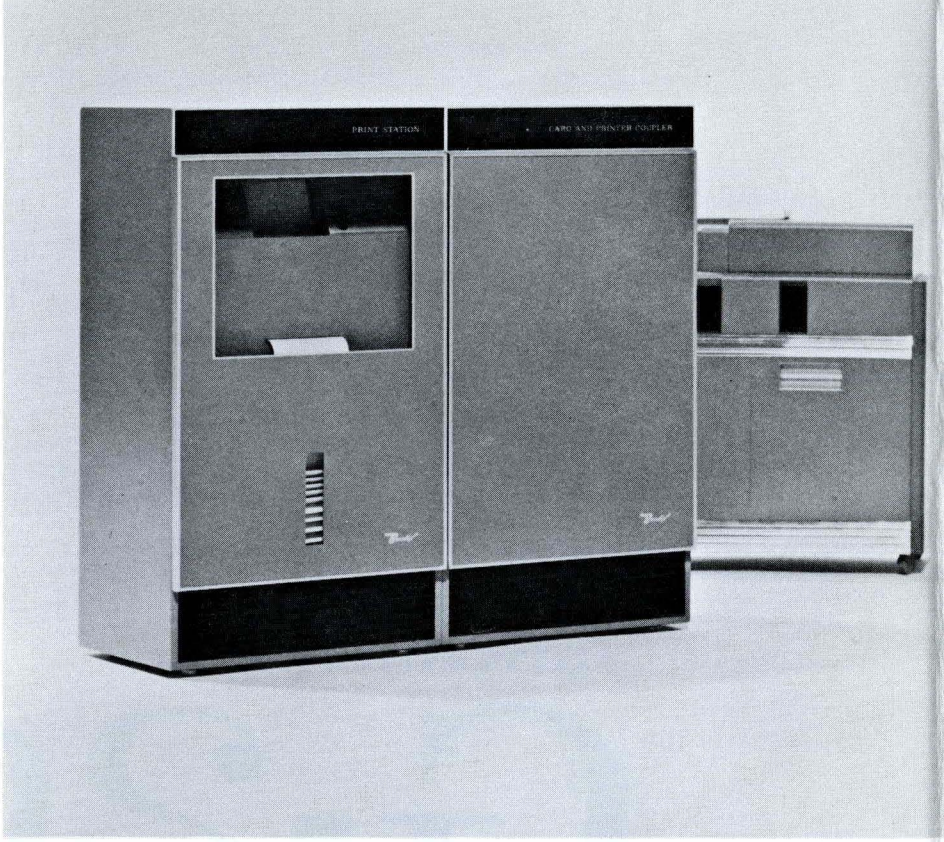
Introduction To Bendix G-20 Programming

SPAR
ALCOM
COBOL
EXECUTIVE

Introduction to G-20 Programming



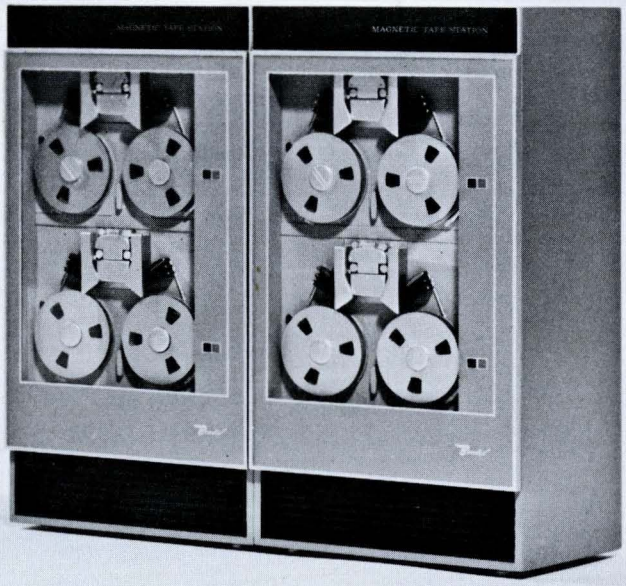


G-20 Programming

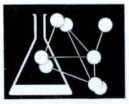
The Bendix G-20 general purpose, automatic, data processing system represents a decisive advancement in computing machinery. Associated with the equipment is a sophisticated programming system that fully uses the features built into the physical equipment. The equipment and the programming system complement each other and give the user powerful computing facilities at minimum cost.

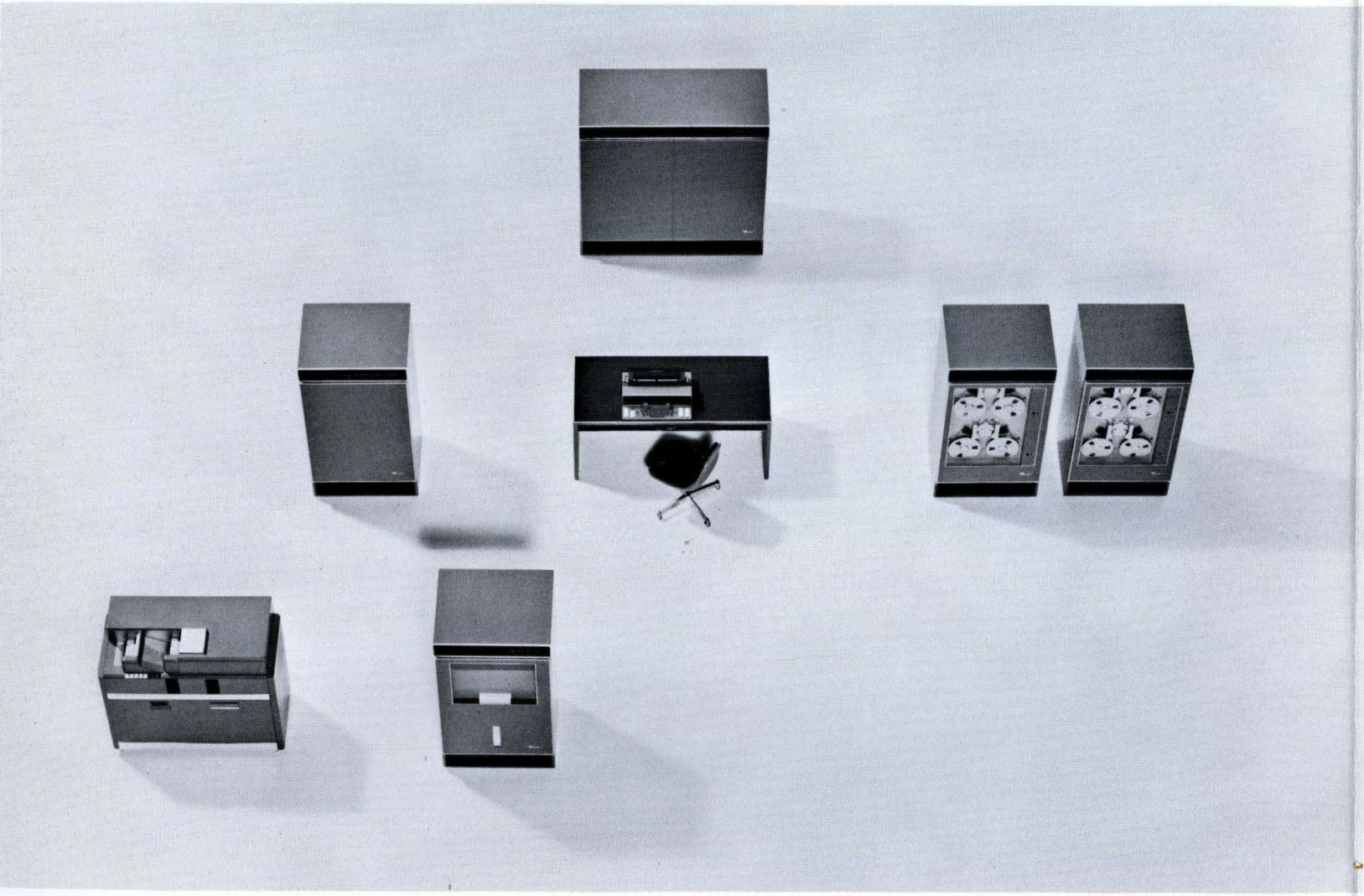
SPACE, the G-20 Programming System, is a comprehensive system that includes a symbolic programming assembly routine, an algebraic compiler, a business compiler incorporating COBOL, and an Executive routine. As additional programming routines are developed, they will be made compatible with the SPACE system. In addition to SPACE, Bendix provides a self-contained assembly program, SNAP, which is particularly designed for those users having a modest configuration of equipment. SNAP includes all elements necessary for the translation and execution of programs.

The information contained in the following pages describes the Bendix programming systems. For a description of the performance characteristics of the equipment in the G-20 System, see A Technical Introduction to the Bendix G-20 System.



Systems



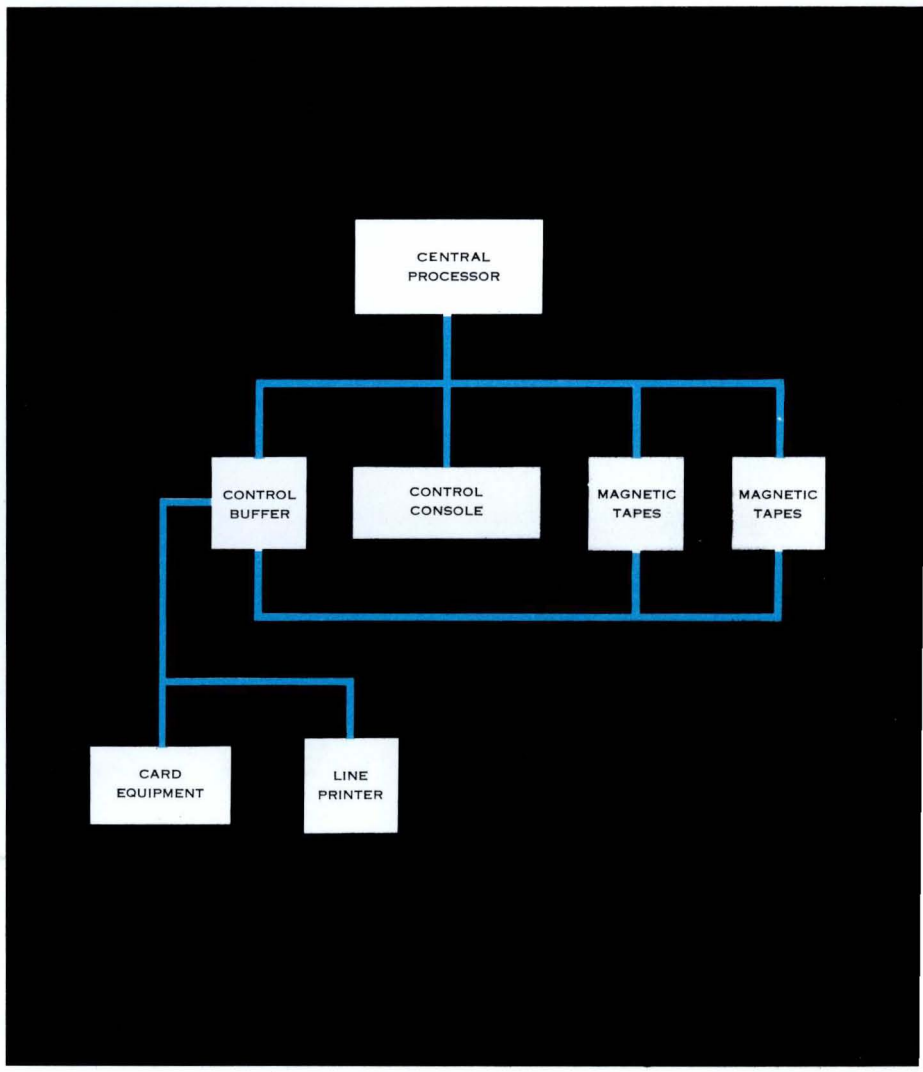


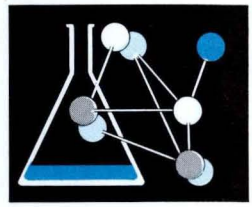
G-20 System Configurations

G-20 system configurations vary in a modular fashion from the minimum system listed below to systems of virtually unlimited size. A minimum system consists of a central processor containing 4,096 words of core memory storage, the associated console, and paper tape or card input/output equipment. From the minimum configuration, the user may expand the system by the addition of magnetic tapes, line printers, control buffers, data communicators and additional core storage.

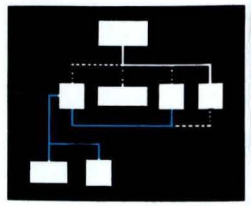
A control buffer added to the system permits the independent use of various elements of peripheral equipment either "on-line" or "off-line". A data communicator added to the system permits simultaneous reading, writing and computing with complete input/output independence. A control buffer can control equipment connected to two communication lines; a data communicator can control equipment connected to 4 communication lines.

Magnetic tape units and control buffers can be switched to secondary lines under program control. A fully buffered, solid state, line printer is available which connects directly to a communication line and has direct communication with the central processor and other controlling units.





G-20 "SPACE"



SPAR

GENERAL PURPOSE LANGUAGE... SPAR, the Symbolic Programming Assembly Routine, gives the programmer direct control of all operations built into the G-20 system. SPAR is ideal for scientific and logical programs which are most conveniently written in "machine language". In the use of the G-20 in on-line computation and control, SPAR is an easy-to-use tool which provides maximum flexibility and efficiency in the processing of data.

ALCOM

SCIENTIFIC LANGUAGE... ALCOM, an ALgebraic COMpiler based on the international ALGOL, gives the scientist and engineer an easy method of expressing problems for the G-20 system. The scope of ALCOM programs covers any problem readily expressed in algebraic and logical formulas. For example, an engineer may easily program matrix operations in the ALCOM language. ALCOM does not require a specialized or previous knowledge of computers or programming systems.

COBOL

BUSINESS LANGUAGE... COBOL is the COmmon Business Oriented Language to describe business problems for computers. As a business programming language, COBOL has received nation-wide acceptance. The COBOL language includes full flexibility in the use of alphabetic words, decimal numbers and special characters. Inventory accounting, production scheduling, payrolls and billing are among the many applications for which COBOL is particularly designed.

PROGRAMMING

EXECUTIVE

SYSTEM CONTROL... The Executive Routine directs the operations of the SPACE Programming System in conjunction with the G-20 computer. The Executive Routine coordinates the computer activity in the translation and execution of all problems in a direct and efficient manner. The Executive Routine has the ability to direct parallel processing of programs and data by spreading the tasks among the many processing elements which comprise the system. For example, the Executive Routine may direct the printing of a report while the central processor independently processes another program.

SPAR COBOL

SPACE Programming

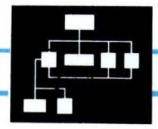
ALCOM EXECUTIVE

RELOCATABLE PROGRAMS

All translators in the SPACE system produce a program assembled in a relocatable form as output. The Executive Routine can allocate the relocatable program to any available memory locations. The relocatable form allows one source language program to incorporate a program written in a different language of the SPACE system. Thus, the programmer has at his disposal a constantly expanding library of sub-programs. In order to efficiently use the library, the Executive Routine can handle all the details necessary to locate the library program.

COMMON METHOD OF INPUT AND OUTPUT

The advanced interrupt facilities of the G-20 system permit internal computation simultaneously with input and output operations. Card reading and punching equipment, magnetic tape units, and controlling units can interrupt the central processor for required servicing. The separate translators in the SPACE Programming System allow the programmer to write simple input/output statements which are translated into the common language for input/output control.



EXPANDABILITY WITHIN "SPACE" SYSTEM

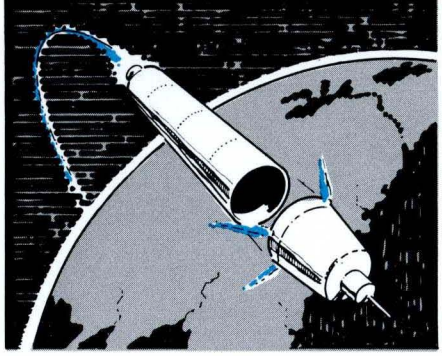
In designing the SPACE system, Bendix provided for the addition of translators for future programming languages. The SPACE system has one major requirement for future translators—the output must follow the rules prescribed for the linkage to the Executive Routine at the beginning and end of the program run.

Since programs written in the present languages will always be available to future translators, the programmer has at his disposal a constantly expanding library of languages.

EQUIPMENT REQUIRED

In order to use the SPACE Programming System, an equipment system configuration must have a central processor with 8,192 words of core memory, its associated control console, two magnetic tape units, a card reader and punch, and a line printer. One magnetic tape unit holds all elements of the SPACE system and also contains the library of programs that have been assembled in relocatable form. A running program may use the space on the tape not needed by the system and library for intermediate storage if necessary. The various routines in the system use the other magnetic tape units during the actual translation of the program languages into relocatable machine commands. The Executive Routine requires only the system tape during program execution. The remaining tapes are available for program use.

Another magnetic tape unit added to the system increases efficiency in program translation. Adding a data communicator or a control buffer increases the overall efficiency by allowing card operations, line printing, and parallel processing to occur simultaneously with computations in the central processor.



General Purpose Language

SPAR

SPAR, the Symbolic Programming Assembly Routine, allows the programmer to use the full capabilities of the G-20 equipment, yet releases him from the burden of many bookkeeping details inherent in machine language programming. The programmer references variables in symbolic notation, and uses subscripts and indirect and relative addresses with ease. SPAR is particularly designed to express problems requiring direct control of all operations of the G-20, and which are not easily written in the algebraic and logical notation of ALCOM. In order to use SPAR with maximum efficiency, the programmer must be familiar with machine commands.

SPAR uses a single address command structure similar to the one used in the G-20 processor, which specifies an operation code and an operand. The operand is normally combined with the contents of the accumulator leaving the results in the accumulator.

Operation Codes

The SPAR operation codes include the complete set of operations of the G-20 processor. See A Technical Introduction to the Bendix G-20 System for a list of the mnemonic codes and a brief description of their functions. Another set of SPAR codes, called macros, allows the programmer to include functions used repeatedly which may require more than a single G-20 command. SPAR contains a built-in set of macros defined particularly for efficient input/output operations.

Each macro code causes a sequence of G-20 commands to be inserted in the program and results in the proper operation performed under control of the Executive Routine.

The macro codes which control input/output through the Executive Routine include:

INPUT I (Unit/List);

The macro inserts the commands to initiate input from the peripheral equipment designated as Unit. The processing of input proceeds under control of a previously defined format designated by I. The central processor stores the input in the memory locations specified in the List.

OUTPUT I (Unit/List);

The macro inserts the G-20 commands to initiate output from the memory locations specified in the List. The central processor sends the words of output to the designated Unit in the prescribed format.

SPAR has built-in formats for transmitting data in Hollerith or row binary form. The programmer may define other formats.

READY Address

The macro inserts the necessary commands to determine if the input/output operations using the memory locations defined by Address are complete. Further processing waits for the completion of the input/output and resumes at the next command in sequence.

The programmer may also define his own macros for use within his current program.

Declarations

Another group of codes, called declarations, appear in the operation field of a SPAR program. The information in the declarations enables SPAR to construct an efficient program. The set of declarations include:

BEGIN, END, ENTRY

The BEGIN and END designate the boundaries of a program or subroutine. The ENTRY indicates the point in the program or subroutine where computation begins.

INDEX, RESERVE

Both declarations reserve memory locations in the program. Index registers use the space reserved by INDEX and data uses the space reserved by RESERVE.

ALF, LWD, SPC, DPC

The declarations indicate that the constants following are in a given format. ALF indicates alphabetic; LWD indicates octal; SPC indicates single precision; and DPC indicates double precision. On the coding sheet, the information is written on the same line as the declaration. SPAR automatically converts single and double precision numbers to computational form.

FORMAT

The FORMAT declaration specifies the form of the data processed in input/output operations. The declaration identifies the type of data and indicates the number of characters and their position.

Operand Assembly

The values which enter into computation are called operands. The translator may place the operand directly in the command using it or may place it in memory locations and refer to it by the address of the location. The value or its address is often a combination of several quantities. The calculation of the value or its address is called operand assembly.

Examples

MPY — 2356.14 means multiply the value in the accumulator by the decimal number 2356.14. SPAR allocates a memory location for the number, enters the number into the location, and places the location address in the command.

MPY — 834 means multiply the value in the accumulator by the integer 834. SPAR places the number 834 in the actual G-20 command.

If the value is not given directly, the programmer must symbolically designate the operand.

MPY ABLE means multiply the value of the accumulator by the contents of a memory location represented symbolically by the name ABLE. SPAR places in the command the address of the variable ABLE.

The name ABLE may refer to the actual value of a variable. The value of the variable must be less than 32,768. In this context:

MPY — ABLE means multiply the value in the accumulator by the integer whose symbolic name is ABLE. SPAR inserts the integer directly into the command.

The programmer may use indirect addressing as follows:

MPY (ABLE) means multiply the value in the accumulator by the number whose address is found in location ABLE.

A command has unlimited indexing facilities. That is, the address of the operand can be modified by the contents of any number of other memory locations. Each location which holds a value that modifies the address of the operand is put in parentheses. If a single memory location from 1 to 63 is used to modify an address, the entire command will fit in a single word position.

MPY ABLE + (I) means multiply the value in the accumulator by the contents of a location whose address is determined as follows:

Add the contents of the index memory location represented by I to the number represented by ABLE. The sum is the address of the operand. Assume that ABLE represents 1300 and the contents of I is 43. The address of the operand is 1343.

The programmer may form complex operand addresses by repeated combinations of these address elements with "+" and "−" signs connecting them.

Illustrative Example

Calculate the absolute value of the innerproduct of two vectors A and B, each having 100 components. The components of the vectors are key-punched on cards in decimal numbers, four per card. The formula is:

$$Y = \left| \sum_{i=1}^{100} A_i B_i \right|$$

The three-letter mnemonic codes in the Operation Field are G-20 machine language code representations. The commands Input, Output, Ready are macros included in the SPAR system. The remaining words in the Operation Field are declarations.

SPAR programming commands are key-punched in cards using a standard algebraic set of Hollerith codes. These cards are entered into the computer under control of the Executive Routine along with the SPAR assembly routine. The assembly routine translates the commands to machine instructions and records them for immediate or future use. The translation process is necessary only once. The machine instructions resulting from the translation are recorded in a compressed, relocatable form on cards for external storage, or in a magnetic tape library of programs and subroutines. Either form may be entered into the computer under the control of the Executive Routine for production runs.

Calculate the absolute value of the innerproduct of two vectors A and B, each having 100 components. The components of the vectors are key-punched on cards in decimal numbers, four per card.

The formula is:

$$Y = \left| \sum_{i=1}^{100} A_i B_i \right|$$

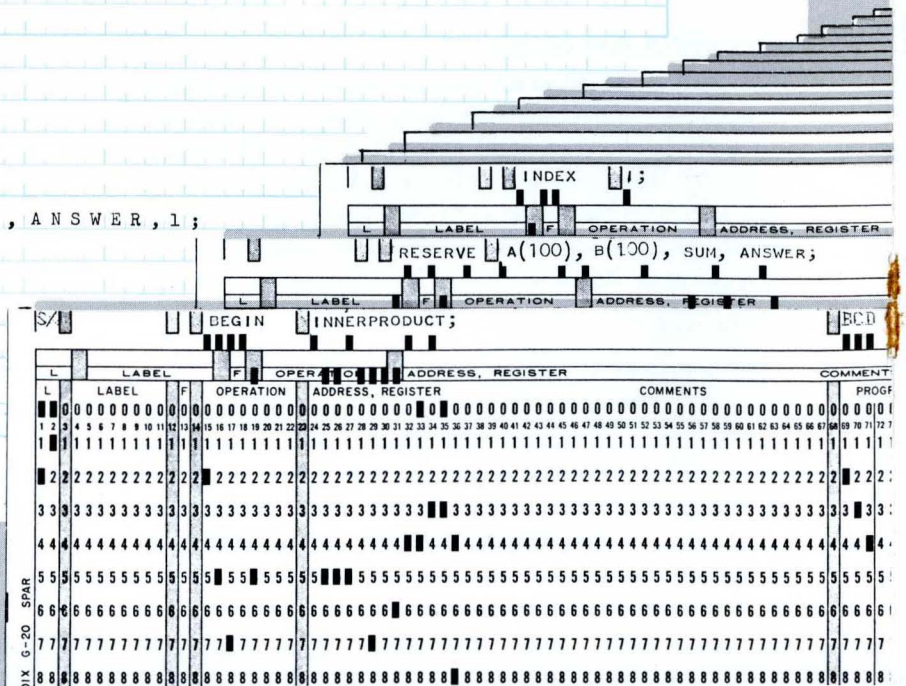
Bendix G-20 SYMBOLIC LANGUAGE INSTRUCTION FORM

	LABEL	OPERATION	ADDRESS, REGISTER	COMMENTS
1	S/	BEGIN	INNERPRODUCT;	
2		RESERVE	A(100),B(100),SUM,ANSWER;	
3		INDEX	I;	
4	BETSY	FORMAT	4F18.3;	
5	CAROL	FORMAT	E12.3;	
6	INNERPRO	ENTRY	; BEGINNING OF PROGRAM	
7		INPUT	BETSY,CARD1,A,100;	
8		INPUT	BETSY,CARD1,B,100;	
9		STZ	SUM;	
10		LXP-	100,I;	
11		READY	A;	
12		READY	B;	
13	START	CLA	A + (I);	
14		MPY	B + (I);	
15		ADD	SUM;	
16		STS	SUM;	
17		SXT-	1, I;	
18		TRA	START;	
19		ADA-	;	
20		STS	ANSWER;	
21		OUTPUT	CAROL,PRIN1,ANSWER,1;	
22		READY	ANSWER;	
23		TRA	(INNERPRO);	
24	/	END	;	
25				

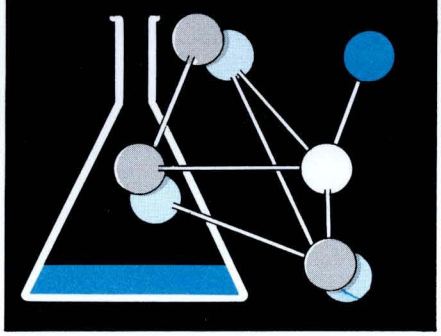
PROGRAM INNERPRODUCT CALCULATION

PROCEDURE

BCD FORM: 1054



SPARALCOMCOBOLEXECUTIVE



Scientific Language

ALCOM

The ALCOM ALgebraic COMpiler permits the expression of scientific and engineering problems in scientific notation and allows the scientist or engineer to program the computer by expressing problems in a familiar notation.

For scientific computation, the G-20 computer equipment provides a number of outstanding features: fast internal computation; floating point arithmetic; single and double precision facilities; 63 special index register locations; and direct, indirect and relative addressing of any memory location. ALCOM incorporates and automatically uses all of these features. The index registers provide easy handling of vectors, matrices and arrays; relative addressing provides for efficient use of subroutines in a program.

Application of ALCOM extends beyond the algebraic field into the area of logical programming through the use of expanded logic facilities. ALCOM's logical programming retains all the simplicity of algebraic programming.

The heart of the ALCOM language is the assignment statement. A typical assignment statement is:

```
Y := A + B * SIN (X);
```

ALCOM automatically generates the G-20 commands which assign to Y the evaluation of the expression to the right of the equality.

Control statements in ALCOM specify the sequence of computation in a program. The words IF, THEN, ELSE, GO TO, and FOR are control statements. An example of an ALCOM control statement is:

```
IF X = ALPHA + 15 THEN GO TO STEP 1
ELSE IF X  >  ALPHA + 15 THEN GO
TO STEP 2;
```

In an ALCOM program, the programmer writes the formulas in their natural order and includes the necessary control statements to direct the proper sequence of computation.

Declarations, another element in an ALCOM program, provide information to the translator concerning the contents of the program. Representative declarations are:

ARRAY and INDEX

The declarations identify and reserve memory space for data and index registers, respectively.

INTEGER, SINGLE, LOGICAL, and BOOLEAN

The declarations INTEGER, SINGLE and LOGICAL specify that the following variable is either an integer, a single precision number, or an octal number. BOOLEAN indicates that a variable has only two possible values.

FUNCTION, PROCEDURE, and ENTRY

FUNCTION and PROCEDURE specify the type of subroutine and ENTRY specifies the entry point for execution of the subroutine.

In addition to the normal algebraic statements, ALCOM contains the logical operators $\wedge$, $\vee$, $\neg$, and \$. The \$ operator represents a shift in the accumulator. These operators, combined with logical variables, form logical assignment statements. For example,

```
F: = ALPHA $ 5 ∧ BETA;
```

means to assign to the logical variable F the result of shifting the variable ALPHA left 5 bits then performing an "extract" operation with BETA.

A programmer may divide an ALCOM program into any number of subroutines, called procedures. Each procedure is a self-contained program requiring only input/output variables for computation. Dividing a program into procedures has a distinct advantage for programs which may be too large for the internal memory of the central processor. Only the procedures necessary to one phase of computation need be in the memory at one time; the other procedures may be read into memory, when needed, replacing those which have already been run. The programmer, through an overlay statement, may automatically cause the overlay to occur at program execution time.

ALCOM example:

Find the temperature at the end of a given interval of time at each of 1000 equidistant points along a bar from temperatures previously recorded at each point.

Use the formula: $Y_i = 2/3\, X_i + 1/6\, (X_{i-1} + X_{i+1})$.

The ends of the bar are kept at 0°F. The interval of time is from 0 to 1 second in increments of 1/600 second.

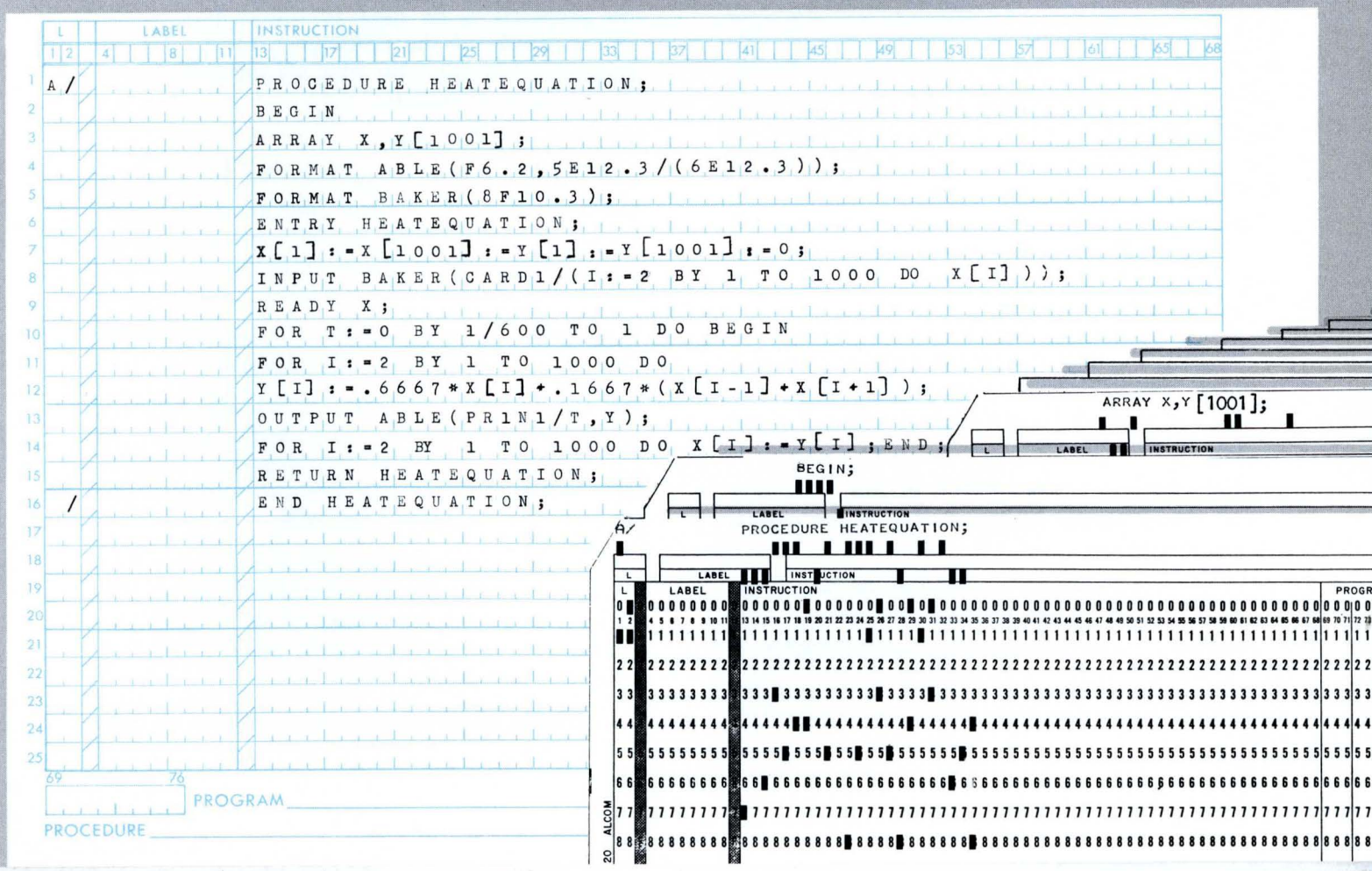
Lines 6 through 15 of the coding sheet are the statements which ALCOM translates into machine language. The other lines are declarations which instruct ALCOM to reserve space for the variables, and determine the format in which ALCOM reads data from cards and the format in which ALCOM prints the results on the line printer.

The information on the coding sheets is keypunched on cards using the existing standard set of Hollerith codes. The cards are read into the G-20 under control of the Executive Routine. The ALCOM translator processes the statements and forms relocatable machine commands. Instructions to the Executive Routine specify whether execution is to occur immediately or whether the program is to be stored for later execution.

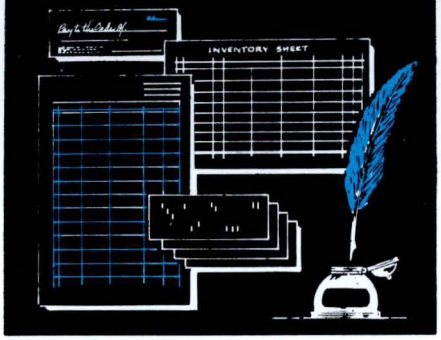
Heat Distribution

Find the temperature at the end of a given interval of time at each of 1000 equidistant points along a bar from temperatures previously recorded at each point. Use the formula: $Y_i = 2/3\, X_i + 1/6\, (X_{i-1} + X_{i+1})$.

The ends of the bar are kept at 0°F. The interval of time is from 0 to 1 second in increments of 1/600 second.



SPARALCOMCOBOLEXECUTIVE



Business Language

COBOL

COBOL is the business data processing language of the SPACE Programming System. The COBOL language is English. COBOL accepts, processes, and transmits data in alphabetic, alphanumeric, or decimal form. The programmer need have no specialized knowledge of either programming or electronic computers. In addition, COBOL gives the business executive the ability to obtain first hand knowledge of business programs processed by the G-20. COBOL permits the rapid generation of reports, special analyses, and surveys in a fraction of the time formerly required.

COBOL is the business language system resulting from the coordinated efforts of the U. S. Government through the Department of Defense, users of computers for business, and computer manufacturers. COBOL uses the features built into the G-20 equipment which make the computer powerful in data processing applications. These features include:

Fast Magnetic Tapes which are capable of read-write speeds up to 120,000 alphanumeric characters per second, or 240,000 decimal digits per second;

Bendix Line Printers which operate at speeds up to 1000 lines per minute with full horizontal and vertical format control without wiring boards;

Data Communicators which permit simultaneous input/output independently of the central processor; and

Flexible equipment configurations which instantly may form off-line sub-systems controlled by a program in the Control Buffer.

The flexibility of the system makes all components available to the central processor when needed.

A Control Buffer may switch tapes to form an off-line system to perform the merge part of a sorting operation. Simultaneously, the central processor computes another totally unrelated problem. Both operations — the one controlled by the buffer and the one in the central processor — continue concurrently with and independently of one another.

Generalized Business Routines

In addition to the COBOL compiler, SPACE contains a set of standard business routines which perform basic business operations. The routines include File Assembly and Maintenance; Generalized Sorting, and Report Generation. These routines are prepared to be used in conjunction with or independently of the COBOL compiler. The language for describing the files, records, and fields is the language of COBOL.

COBOL Language

The COBOL language permits the business programmer to write problem statements in English using ordinary sentence structure and punctuation. He uses English phrases to describe the records and files of his problem.

A COBOL program has four divisions: Identification, Environment, Data and Procedure.

The Identification division contains information to identify a problem. The information enables the Executive Routine to refer to the problem in library searches, in program scheduling and execution. It permits easy external recognition.

The Data division specifies all information pertaining to files necessary to solve the problem. Files cover a broad representation of data which may exist in core memory, on cards, on magnetic tape, or on the line printer.

A symbolic name represents each item of data.

A typical name might be HOURLY-RATE-OF-PAY. The programmer uses English phrases to give information about each item. Such phrases might be: CLASS IS NUMERIC; POINT LOCATION IS LEFT 2 PLACES; SIZE IS 4 CHARACTERS.

The Procedure division contains the English sentences defining the necessary computation. Some key verbs for the division are: ADD, SUBTRACT, MULTIPLY, DIVIDE, OPEN (files), READ, WRITE, CLOSE, MOVE (data), STOP, ENTER, and EXIT.

The programmer may cause a direct transfer from one point in the program to another point by using the word GO. A conditional transfer results by using the word IF followed by an expression which may or may not be true.

COBOL automatically converts all decimal data to computational form and performs necessary editing operations for input and output. Label reading, writing, and checking of tapes is automatic with use of the verbs OPEN and CLOSE.

The Environment division specifies the equipment and its configuration on which a program is to run and the number of peripheral units to which specific files are assigned. The information enables COBOL to generate the most efficient program with the available equipment. In addition, the Environment division specifies the points at which information is saved to enable reruns and to continue operation if the program is interrupted.

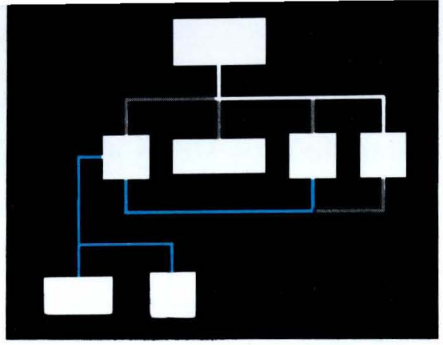
The example illustrates typical sections from the Data and Procedure divisions.

The information on the coding sheets is keypunched on cards using the existing standard set of Hollerith codes. The Executive Routine controls the reading of the cards by the G-20. The COBOL translator processes the statements into a relocatable form. Instructions to the Executive Routine specify whether execution is to occur immediately or whether the program is to be stored for later execution.

An example illustrating typical sections from the Data and Procedure divisions.

```
 1  DATA DIVISION PREPARED FOR BENDIX G-20.
 2  FILE SECTION.
 3  FD  TAPEACCTS; LABEL RECORDS ARE STANDARD; DATA RECORDS ARE ACCOUNTS,
 4      SUMMARY; SEQUENCED ON TBRANCH, TACCT-NO.
 5      01  ACCOUNTS.
 6      02  RCODE; SIZE 1; USAGE IS COMPUTATIONAL; CLASS IS NUMERIC.
 7          88  ACCOUNTS VALUE IS 1.
 8          88  SUMMARY VALUE IS 2.
 9      02  TBRANCH; SIZE 2; USAGE IS COMPUTATIONAL; SYNCHRONIZED RIGHT:
10          CLASS IS NUMERIC.
```

```
 1  PROCEDURE DIVISION
 2  COMMENT. NOTE WE ASSUME THAT THE TAPEACCTS FILE IS IN PROPER SEQUENCE,
 3      AND THAT THE TRANSACTION FILE WILL CONTAIN NO NEW BRANCHES.  A NEW
 4      BRANCH WILL PRODUCE AN ERROR STOP.
 5  BEGIN. OPEN INPUT TAPEACCTS, TRANSACTIONS. OPEN OUTPUT NEW-TAPEACCTS,
 6      EXCEPTIONS. WRITE HEADER-1 FROM C-HEADER-1; WRITE HEADER-2 FROM
 7      C-HEADER-2.
 8  CLEAR. MOVE ZERO TO TSUM. MOVE ZERO TO T-NOACCTS. READ TAPEACCTS.
 9  0060. READ TRANSACTIONS. MOVE A-BRANCH TO LAST-BR. MOVE A-ACCT-NO TO
10      LAST-ACCT-NO.
11  COMPARE-BRANCH. IF TBRANCH IS GREATER THAN A-BRANCH GO TO ERROR-1;
12      OTHERWISE IF LESS THAN A-BRANCH GO TO NO-CHANGES.
13  COMPARE-ACCTS. IF TACCT-NO IS LESS THAN A-ACCT-NO, GO TO NO-CHANGES;
14      OTHERWISE IF EQUAL TO A-ACCT-NO GO TO TEST-TRAN1; OTHERWISE IF
15      OPENING GO TO 0006; OTHERWISE GO TO ERROR-2.
16  TEST-TRAN1. IF CLOSING GO TO 0007; OTHERWISE MOVE ACCOUNTS TO NEW-ACCOUNTS;
17      ALTER SWITCH-1 TO PROCEED TO NEXT-TAPE-ACCT.
18  TEST-TRAN2. IF DEPOSIT GO TO 0001. IF WITHDRAWAL GO TO 0002. IF
19      CHANGE-NAME-AND-ADD GO TO 0005. IF CHANGE-NAME GO TO 0003. NOTE IF
20      OTHER BRANCHES HAVE NOT BEEN TAKEN THIS MUST BE A CHANGE-ADD.  IF
21      A-ADDRESS EQUALS BLANK-ADD GO TO ERROR-4; OTHERWISE MOVE A-ADDRESS
```



SPARALCOMCOBOLEXECUTIVE

System Control

EXECUTIVE

The Executive Routine coordinates SPACE data processing activity in the G-20. The activities controlled by the Executive Routine include communication with the operator to receive new tasks and to give information on the status of current processing. The Executive locates and loads into available memory locations programs and subroutines for computation. It monitors the status of running programs, directs the input/output for maximum data flow, and performs "overlay" when a complete program must be brought into memory in segments.

Many of the tasks that the Executive performs are necessary in any computing installation. Recognizing the common functions and centralizing them in a single modular routine eliminates considerable duplication of effort and programming.

The Executive Routine is divided into two segments, called the Supervisor and the Monitor. The Monitor may be one of several specialized monitors which fit the translator employed. That is, ALCOM, SPAR, and COBOL may have different monitors which are connected to the common Supervisor. Each of these is referred to as the Monitor.

THE SUPERVISOR

The Supervisor is called into core from the system tape whenever major functions must be performed or when extended communications with the operator are required. The routine arranges a schedule for computation, obtains the programs to be run, and prepares them by relocation and parameter assignment. The Supervisor also transmits messages via the console to the operator regarding assignment of reels of magnetic tape and concerning the status of equipment and program computation. In addition, the Supervisor performs diagnostic checks on programs and equipment when problem solution is interrupted.

When all requirements for beginning or continuing computation are satisfied, the Supervisor loads the Monitor and the programs to be executed into core memory. The Supervisor thereby replaces itself.

THE MONITOR ROUTINE

The Monitor is the smaller of the two segments of the Executive Routine. It remains in core during the time problems are being processed. The primary functions of the Monitor are:

To handle program interrupts;
To expedite the flow of information among input/output units; and
To recall the Supervisor when additional control is required.

Interrupt Servicing

The G-20 permits interruption of computation whenever an illegal command is encountered, whenever the results of computation exceed prescribed limits, and at the programmer's option. Additional interrupts provide for input/output servicing and for timekeeping by means of an interval timer included as standard equipment. At the moment of interruption, the G-20 stores the location of the last command executed and transfers control to a fixed location in core memory. This location contains the beginning of a stored program routine, called the Interrupt Service Routine.

The SPACE Interrupt Service Routine provides four classes of exits: to the Input/Output Control routine; to the Monitor; to subroutines requested by the programmer; and to the point in the program at which the interruption occurred.

The Input/Output Control determines the peripheral units requiring service. If the current task is not complete, the I/O routine continues the tasks for the unit; if there are additional tasks to be performed, the I/O routine initiates them.

The interrupts designed for the Monitor include those for timekeeping and calling the Supervisor if illegal operations prevent further problem processing. The Monitor controls parallel processing in the system.

A "flag" on data or commands can cause an interrupt. The programmer may specify entry into one of six special subroutines associated with his problem when a "flag" causes an interrupt. The only requirement is that each of the special subroutines must return control to the Interrupt Service Routine (ISR) to handle other interrupts which may have occurred.

Finally, when all interrupts have been serviced, ISR returns control to program computation. The user has the option of specifying the priority of interrupts when his system is installed and of changing the priority as his needs change.

The SPACE Interrupt Service Routine occupies about 150 memory locations in the central processor. It contains all the coding necessary for determining which interrupt has occurred and for transferring to one of the exits mentioned above. Depending on the priority of request, 150 to 600 microseconds elapse from receipt of interrupt to exit.

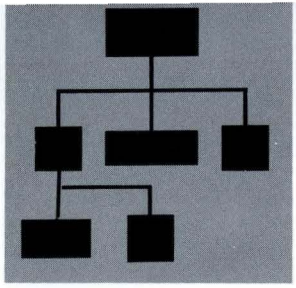
Other programming systems designed to make adequate use of the equipment may use the SPACE Interrupt Service Routine.

Input/Output Control

The Monitor contains the Input/Output Control routine which handles all requirements for data transmission. The routine includes a table containing status information of peripheral units and another table containing the tasks which each unit must perform. When a task is to be done, it is entered into the table and immediately initiated if the specified unit is free. If the unit is a "slow" device such as card equipment or if the task can be done independently of the central processor, computation in the main program proceeds with the task. If the unit is not free to initiate the task, computation proceeds in the main program. Normally, the peripheral units interrupt the computer when they become available for further tasks. Processing of the program halts if the program requires the completion of data transmission before continuing. On receiving a "READY" instruction, the I/O Control routine interrogates the tables for completion of the required tasks. If the tasks are not complete, control is transferred to the monitor for possible alternate problem processing.

The I/O Control requires routines for efficient handling of only the peripheral units required by the problem and routines necessary for automatic data conversion to computational form. The minimum group consists of the console and magnetic tape routines. If the problem requires line printers, card or paper tape equipment and if conversions are required from decimal to computation form, the Supervisor loads these routines with the problem and the Monitor. Thus, SPACE automatically assembles only the programs required for problem solution and for Executive control.

The number of memory locations required by the Monitor varies with the problems being solved. If parts of a problem require only magnetic tapes, no conversions, and no alternate problems in memory for parallel processing, the Monitor needs about 1000 locations including magnetic tape-handling routines. For larger problems requiring additional peripheral equipment and parallel processing of other programs, the Monitor requires more core storage.



G-20

SNAP Programming

SNAP is an independent programming system which may be used with the smallest G-20 system. The SNAP system permits programs to be written in a symbolic language which retains the flexibility of G-20 machine language. SNAP contains all facilities required for program translation, loading, and execution.

The SNAP system consists of:

A convenient source language which is a symbolic form of machine language. This SNAP language, in which programs are coded, combines all features of machine language with an extremely flexible SNAP addressing system.

A Director program through which all operations are performed. The Director contains the SNAP loader and an interrupt service routine. The Director is kept in memory during any SNAP operation.

An assembly routine which translates SNAP source language to machine language.

A group of standard subroutines for input/output operations and debugging.

Labels

Alphanumeric labels may be used in SNAP to represent addresses or numeric values. Convenient symbolic labels may be assigned to commands, constants, and storage areas. These program elements then may be addressed by use of the labels. For example, the constant π may be stored in a program with the label PI. In that program, operations using π may simply address the constant by use of the symbolic address PI.

Relocatability

The SNAP loader has the ability to load programs assembled by SNAP into any area of memory for execution. This feature, called relocatability, means that any number of inter-related programs can be loaded into memory at one time for execution. The first program loaded goes into the first available locations; subsequent routines, as they are loaded, are relocated, by the loader, into adjacent areas of memory.

The SNAP assembler and all subroutines are relocatable. In addition, any program written in SNAP will be relocatable if it uses labels for symbolic addressing.

When a label is used in a relocatable program as an address, that address represents a given number of locations beyond the starting point of the program. Therefore, although an entire program may be relocated in loading, symbolic addressing within a program remains constant relative to that program.

SNAP Command Structure

The source language of SNAP consists of a group of alphabetic commands meaningful to the assembler. These commands consist of the SNAP symbolic form of machine commands and of constant declarations and other instructions to the assembler.

SNAP source language commands have the following form:

Label	Flag	Op Code	Mode	Address-Register

The Label field of commands, constants, and memory reservation commands is used to assign symbolic labels, for use in addressing, to the command, constant, or data storage area.

The Flag field is used to store one or two flags in commands or constants to permit branching in execution.

The Op Code field contains the alphabetic operation codes of SNAP source language. Operation codes consist of instructions to the assembler and of a mnemonic form of machine commands.

Assembler instructions define the beginning and end of a program, reserve and label index registers and storage areas, and control the labels used in symbolic addressing.

Constant declarations permit any number of constants to be stored and assigned labels for symbolic addressing. Numeric constants may be entered in octal or decimal notation in floating or fixed point form. Alphanumeric information and logic words may also be entered as constants. Machine commands are written in a mnemonic form which makes all machine operations available in a simplified notation.

Mode may be indicated in source language in a variety of ways. However, when commands are written in their "normal" mode, no indication of mode is required.

Addressing in SNAP is especially powerful. The features inherent in machine language are combined with arithmetic operations available during assembly to produce an extremely flexible method of addressing. Address notation permits a string of any number of elements to be written in the address of a command. These elements may be numbers or numeric addresses or they may be labels representing numbers or addresses. Such labels may represent relocatable commands, constants, and data storage areas.

These elements of a SNAP address may be connected with symbols indicating addition, subtraction, multiplication, and division. These indicated arithmetic operatons are performed during assembly and the result is output as a number in the assembled, machine language program.

Example of an address string:

BETA+19+ALPHA – $147

In assembly, the decimal number 19 will be converted to octal and added to the sum of the values represented by ALPHA and BETA; from this sum the octal number 147 will be subtracted. The result of this operation will be one number which will be output in the assembled, machine language program.

The Address-Register field of a SNAP source command may contain one string of elements to represent a basic address and another string to refer to an index register modifying that basic address. A comma separates two strings.

Example of a SNAP command using address strings:

```
Label   F  Op Code   M  Address-Register
INPUT      RZO              100
TALLY      RLX                3
                .
                .
                .
           CLA        INPUT+12, TALLY+1
```

In this example, 100 words have been reserved for an input area and the label INPUT is assigned to the first word of the area. Three index registers have been reserved and the label TALLY has been assigned to the first. The third operation code is a machine command in which simple address strings have been used. The basic address referred to is the 13th word of the reserved input area and the index register referred to is the second of the three reserved. The command assembled from the source language command above will clear the accumulator and add to it the contents of the word in the input area which is beyond the 13th word by the amount contained in the second index register reserved.

Assembly

To operate with SNAP, the Director is first loaded into memory. Then, if it is desired to assemble a program written in SNAP, the assembler is loaded

with a console instruction to the Director. The assembler reads cards containing SNAP source language and translates this symbolic program into a machine language program. The result of assembly is a deck of punched cards, called the object language deck, which is in a relocatable machine form. In this form the program may be loaded and executed as required under Director control.

Programs may be assembled on a G-20 system consisting of only a Central Processor with one memory module, a CC-10 Control Console, a reader and punch unit, and a line printer. The assembly process automatically prints a listing of the source language, the resulting object language, and irregularities in the source language. Irregularities are flagged and identified by type to aid in debugging.

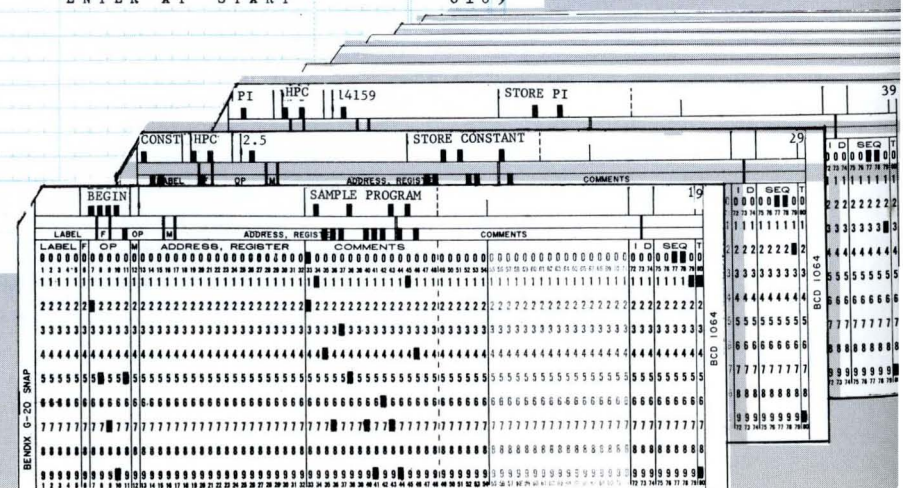
After assembly, programs may be executed on a G-20 system of any size. This flexibility permits programs for large systems to be assembled and partially debugged on a minimum system.

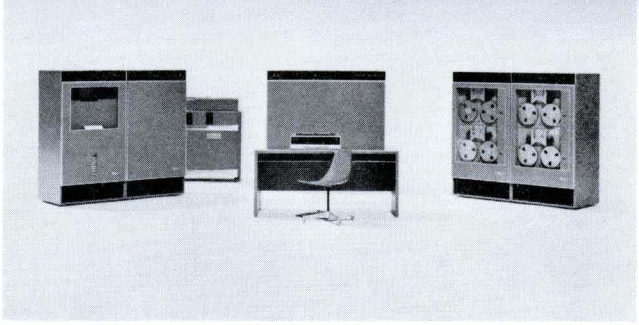
Subroutines

A standard group of subroutines for input/output operations and debugging is available in relocatable binary form for SNAP. In addition, any SNAP program, with minor modifications, may be used as a subroutine of any other SNAP program. Because of the feature of relocatability, any number of standard subroutines, any program, and any number of programs used as subroutines can be loaded into memory for coordinated execution. SNAP source language permits any number of index registers and memory locations to be used in common by two or more programs. These common locations facilitate communication between subroutines and between the control program and any subroutine.

An example of SNAP Programming: Assume that 100 numbers are to be stored in adjacent memory locations. These numbers, if smaller than π, are to be replaced by their squares. Numbers equal to or larger than π are to be replaced by the product of 2.5 and the original number. Input/output commands have been omitted for brevity.

	BEGIN		SAMPLE PROGRAM	0011/9
CONST	HPC	2.5	STORE CONSTANT	0029
PI	HPC	3.14159	STORE PI	0039
DATA	RLS	100	RESERVE DATA LOC	0049
COUNT	RLX	1	RESERVE REG	0059
	LXP-	100,COUNT	LOAD REG	0069
START	CLA	DATA-1,COUNT	ACCESS NUMBER	0079
	FLO	PI	LESS THAN PI?	0089
	TRA	SQARE	YES,GO TO SQARE	0099
	MPY	CONST	NO,TIMES 2.5	0109
	STS	DATA-1,COUNT	REPLACE	0119
	TRA	PLACE		0129
SQARE	MPY	DATA-1,COUNT	SQUARE NUMBER	0139
	STS	DATA-1,COUNT	REPLACE	0149
PLACE	SXT-	1,COUNT	DEC REG,DONE?	0159
	TRA	START	NO,REPEAT	0169
	TRA	CNTRL	YES,GO TO DRECTR	0179
	END	START	ENTER AT START	0189





Other G-20 Programs

Bendix provides its users with other special programs to serve particular functions.

A library of routines is available for the SPACE and SNAP Programming Systems. The library includes elementary mathematical functions:

Sine	Bessel Functions
Cosine	Numeric Integration
Tangent	Square Root
Arctangent	Matrix Handling
Exponential	Polynomial Evaluation
Logarithm	Roots of Polynomials

In addition, the library also contains Number Conversion Routines and General Service Routines.

SNAP to SPACE Relocatable Form

Bendix will provide a program to translate the output of SNAP programs to the SPACE relocatable form. SNAP programs must use symbolic coding exclusively. The user, therefore, does not need to reprogram any error-free problems symbolically coded in SNAP language.

Fortran

Bendix provides a routine to compile for the G-20 any program written in the Fortran II language.

650 Simulator

Bendix will provide a routine to simulate the 650 machine. The G-20 with the simulator may run any error-free program coded for the basic 650 with floating-point, index registers, added drum memory, magnetic tapes, card and tabulating equipment designed for the 650.

Offices:

BOSTON 16

607 Boylston Street
COngress 2-9110

CHICAGO 11

919 N. Michigan Avenue
MIchigan 2-6692

CLEVELAND 13

55 Public Square
CHerry 1-7789

DALLAS 1

1511 Bryan Street
RIverside 7-8805

DENVER 3

655 Broadway
Suite 910
ALpine 5-1403

DETROIT 37

12950 West Eight Mile Road
JOrdan 6-8789

HUNTSVILLE, ALA.

Holiday Office Center
Memorial Parkway, South
539-8471

KANSAS CITY 11, MO.

3430 Broadway
VAlentine 1-8681

LOS ANGELES

291 S. La Cienega Blvd.
Beverly Hills, California
OLeander 5-9610

NEW YORK 17

205 East 42nd Street
Room 1205
ORegon 9-6990

SAN FRANCISCO

1330 Broadway
Suite 1121
Oakland 12, California
GLencourt 2-3664

TULSA 14

1754 Utica Square
RIverside 3-6485

WASHINGTON 6, D. C.

1000 Connecticut Avenue, N.W.
STerling 3-0311

CANADA

Computing Devices of Canada
P. O. Box 508
Ottawa 4, Ontario, Canada
TAlbot 8-2711

OTHER COUNTRIES

Bendix International Division
205 E. 42nd Street
New York 17, New York
MUrray Hill 3-1100

Bendix Computer Division

LOS ANGELES 45, CALIFORNIA

THE Bendix CORPORATION

BSP-12601-1161