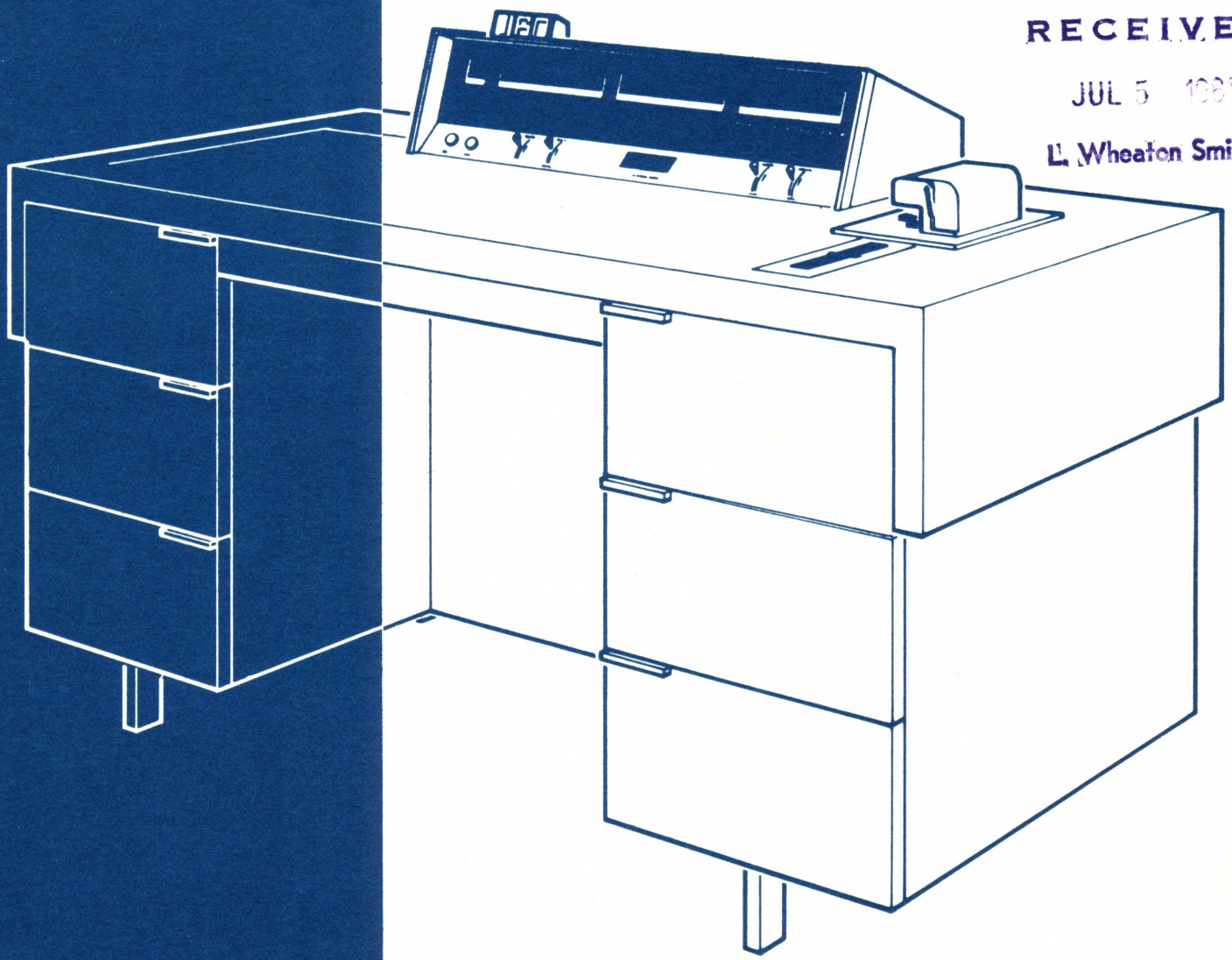
160

Systems Programs for the 1 6 0 Compute



control data | corporation
COMPUTER DIVISION
501 Park Avenue
Minneapolis

SYSTEMS

PROGRAMS

FOR THE

160

COMPUTER

Publication No. 084b

June 1961

CONTROL DATA CORPORATION

SYSTEMS PROGRAMS

FOR THE 160 COMPUTER

FOREWORD

The 160 Systems programs described in this volume have been prepared by the programming staff of Control Data Corporation. This manual was prepared to aid the experienced programmer by collecting the existing material on the 160 computer in a standardized form for easy reference.

Section 3 describes standards and procedures utilized in organizing programming for the 160.

Programs (Section 4) are discussed in full, and detailed operating instructions furnished. The machine language listing of these programs were not included because tapes are available either at the installation or by request from Control Data Corporation. Flow charts have also been omitted since the programs are fully operational. Should a programmer wish to obtain flow charts they will be mailed upon application from Control Data Corporation.

The manual is not to be construed as a replacement for the Programming Manual nor was it meant as a study guide. As new routines are generated and approved they will be distributed to holders of this manual.

Corrections and additions to this manual may become necessary and the programming staff of Control Data Corporation welcomes any suggestions.

CONTROL DATA CORPORATION

160 COMPUTER

Figure 1. Model 160 Computer

CONTROL DATA CORPORATION MODEL 160

The MODEL 160, figure 1, is a desk-sized, general purpose digital computer that is completely transistorized; the memory is high-speed ferrite core storage. Significant advantages in the design of the Model 160 are:

Average instruction execution time (including memory access time)	15 microseconds
4096 12-bit words of storage	6.4 microseconds cycle time
All transistorized 5 megacycle circuitry	identical to 1604 circuits
Complete line of input-output equipment	
Compatibility with 1604 computer in the Satellite Computer Configuration	
High Reliability	
Low Power Consumption	750 watts
Versatile Addressing	direct, indirect and relative
64 Instructions	
Programming packages available	
Projection Display	concise presentation of computer status and register contents

FUNCTIONAL CHARACTERISTICS

The CONTROL DATA 160 is a stored program, high-speed, digital computer. The instruction logic is single address with one instruction per computer word.

The computer memory consists of 4096 12-bit words of magnetic core storage. The complete memory cycle is 6.4 microseconds. Instructions and operands are available for use by the computer control unit 3.2 microseconds after the initiation of a memory reference.

The CONTROL DATA 160 instruction repertoire provides 64 instructions which are classified into 20 functional groups. Instruction execution times range from 6.4 microseconds to 25.6 microseconds.

Information transfer and arithmetic operations are done in the parallel mode. When information is transferred within the computer, all bits of the 12-bit word in a computer register are transmitted simultaneously.

INPUT-OUTPUT CHARACTERISTICS

Standard input-output equipment consists of a photoelectric paper tape reader that operates at 350 characters per second; and a high-speed paper tape punch that operates at 110 characters per second. The computer console provides a display of the principal registers and the key lever switches for manual control of the computer. The contents of the displayed registers can be altered manually to facilitate program checkout and to allow decisions made at the console to affect the course of action taken by a program.

Binary or binary coded information (BCD) is transmitted into and out of the computer at a maximum transfer rate of 78, 000 words per second. Input and output transmission is done in the parallel mode with up to 12 binary bits of information being transmitted simultaneously. Up to five external control units may be attached to the Model 160. Paper tape input-output is not considered external equipment. Separate external function logic is employed to achieve a high degree of flexibility in controlling data transmission to and from peripheral equipment. Control transmission paths are independent of those used for the transmission of data.

APPLICATIONS

The 160 can perform many functions for which special purpose devices are normally required.

a) punched card to magnetic tape conversion
b) magnetic tape to punched card conversion
c) paper tape to punched card or magnetic tape conversion
d) magnetic tape to line printer output
e) magnetic tape duplication and comparison
f) typewriter input to any of the above

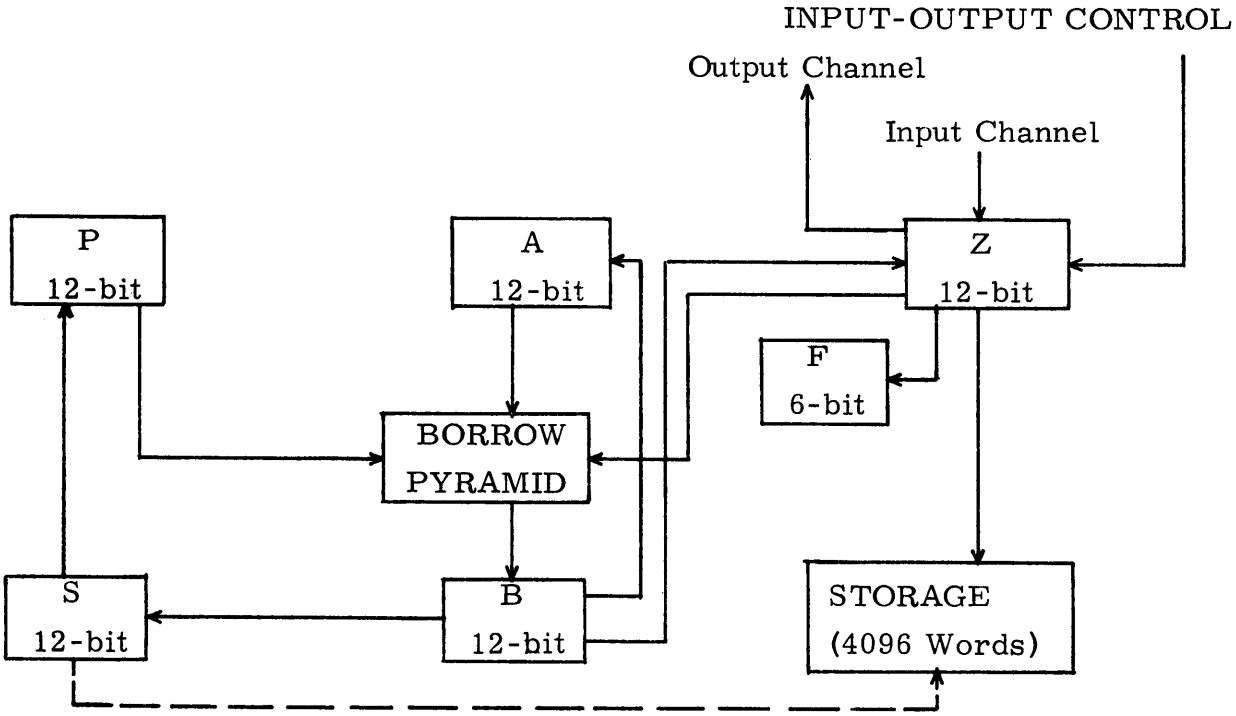
In addition to the normal peripheral operations listed above the 160 can be programmed to do the following:

a) character validity checking
b) record sequence checking
c) binary to BCD and BCD to binary conversion
d) editing and format control
e) merge-sort
f) code conversion

A floating point arithmetic subroutine package enables the solution of engineering and statistical analysis problems with great precision.

The Control Data 160 can be connected as a satellite to the large scale Control Data 1604 computer by means of the 1607 magnetic tape system. This allows data transmission to take place between the two computers at a rate of 78, 000 12-bit words per second. Economy is effected by the fact that one tape system is used jointly by the two computers.

160 COMPUTER, SIMPLIFIED LOGICAL BLOCK DIAGRAM

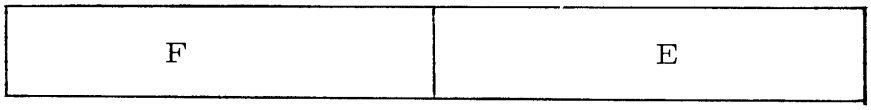


Register

A - Accumulator	holds the result of all arithmetic and logical operations.
B - Auxiliary Arithmetic Register	transient register used in conjunction with the borrow pyramid in additions and subtractions.
F - Function Code Register	holds the function code of the instruction currently being executed.
P - Program Address Register	holds the address of the instruction currently being executed.
S - Memory Address Register	holds the address of the memory location being referenced.
Z - Communication Register	transmits information between the computer and its memory and between the computer and its associated input-output equipment.

INSTRUCTION FORMAT

When a computer register is used to hold an instruction, the contents of that register are interpreted as shown:



F is the 6-bit function code. (bits 06 through 11)

The function code determines which of the 64 possible instructions will be performed.

E is the execution address. (bits 00 through 05)

The normal use of E is to determine the location of an operand to be used in executing an instruction. E is always treated as a positive quantity.

Each of the 4096 memory locations has a unique address. Depending on the function code used, E will be interpreted in one of the following ways to refer to any of the memory locations.

No address mode (n).

E is used as the lower 6-bits of a 12-bit operand. The upper 6-bits are zeros. No storage reference is made. For the shift A instruction, E specifies a particular type of shift operation.

Direct address mode (d).

E is interpreted as the address of one of the first 64 words in core storage and the contents of that memory location become the operand.

Indirect address mode (i).

E is used to address one of the first 64 memory locations. The 12-bit contents of that location are then used as the operand address.

Relative address forward (f).

E is added to the contents of P (P is the address of the current instruction) to obtain the operand address. The operand thus addressed cannot be further than 63 locations forward of the current instruction. P is unchanged.

Relative address backward (b).

E is subtracted from the contents of the P register to obtain the operand address. The operand cannot be further than 63 locations immediately preceding the current instruction. P is unchanged.

ARITHMETIC

The arithmetic performed by the Control Data 160 is called one's complement arithmetic.

When a computer register or core memory location contains a word which is to be interpreted as an arithmetic operand, or result, the following rules apply.

All positive numbers have a "0" in the most significant position, and all negative numbers have a "1" in that position. The leftmost position is thus the sign digit and can be handled as any other digit in the word. The concept of a "0" for positive and a "1" for negative in the most significant position (12th bit) is used.

The complement representation of any binary number is derived from the normal one by interchanging "1's" and "0's". The most direct way to determine the magnitude of a negative number is to form the one's complement of that number.

Examples:

Sign ——— 000 000 010 110 = $+26_8$ or $+22_{10}$

111 111 101 001 = -26_8 or -22_{10}

Normally positive zero (all zeros) is used in computation; however, negative zero (all ones) may be used if needed. An exception being negative zero $\neq$ zero for a zero-non-zero jump.

INSTRUCTION GROUPS

The 64 instructions may be classified according to the functions listed below.

Legend:	( )	indicates contents of the register named.
	M	designates an operand address.
	→	indicates location of result of operation.
	$(M)^{\prime}$	designates the logical, bit-by-bit, product operation.

ADD:	(A) + (M) → A
SUBTRACT:	(A) - (M) → A
LOAD:	(M) → A
STORE:	(A) → M
LOAD COMPLEMENT:	$(M)^{\prime}$ → A
LOGICAL PRODUCT:	Form in A the logical product of original contents of A and operand.
LOGIC SUM:	Form in A the logic sum of original contents of A and operand.
INPUT:	Transfer a word or successive units of information from an external device to computer memory.
OUTPUT:	Transfer successive or E portion of instruction computer memory to the external equipment.
CONDITIONAL JUMP:	Jump depending on the status of A register.
UNCONDITIONAL JUMP:	Jump regardless of the status of A register.
SHIFT:	E = 02 shift (A) left one position E = 10 shift (A) left three positions E = 12 mult. (A) by octal 12 all shifts are end-around (circular)
REPLACE SHIFT:	Shift (M) left one place in A and replace (M) with (A).
REPLACE ADD:	(A) + (M) → A, M
REPLACE ADD ONE:	(M) + 1 → A, M
HALT:	Stop computer operation.

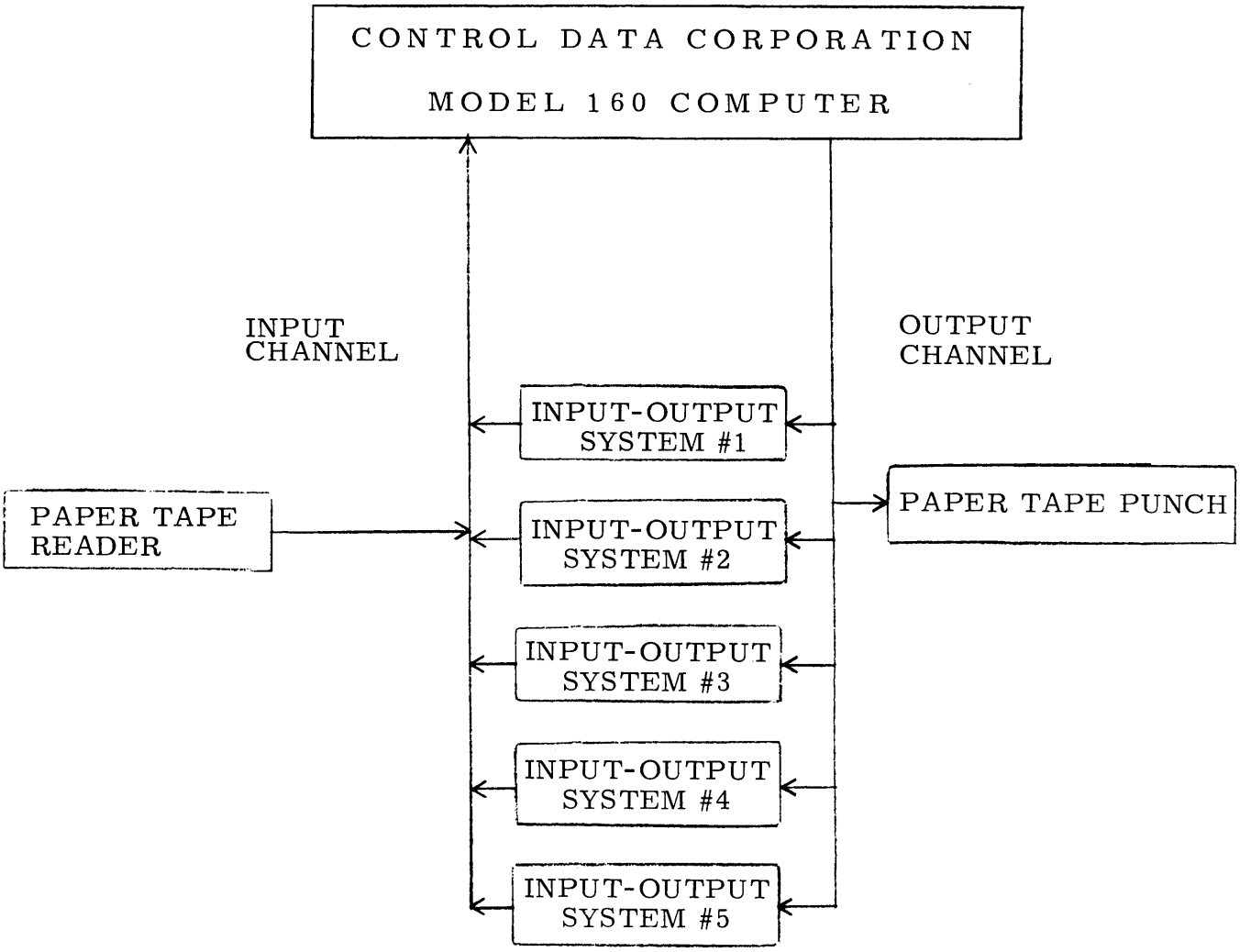
PROGRAMMING EXAMPLE: DUPLICATE PAPER TAPE

Memory Location	Content	Comment
0000	7507	External Function: select paper tape reader input
0001	7210	Input Instruction: read in two frames
0002	0072	Terminating Address + 1
0003	7505	External Function: select paper tape punch output
0004	7305	Output Instruction: punch out two frames
0005	0072	Terminating Address + 1
0006	6506	(A) Non-Zero Jump: go back to the instruction at location 0000
0007	4102	Function Code to Select Reader Input (used by instruction at 0000)
0010	4104	Function Code to Select Punch Output (used by instruction at 0003)
0011	0070	Starting Address for Input and Output Area (used by instructions at 0001 and 0004)

PROGRAMMING AIDS

160 Assembly Program (OSAP):	A two pass symbolic assembly program and assembly correction program.
Service Routine Library:	A set of general purpose programs for input, output, and translation of data and programs.
Library of Subroutines:	A set of arithmetic routines and elementary function routines.
Interpretive Arithmetic Packages:	The following routines employ an extremely fast interpretation program to perform arithmetic operations on extended precision operands.
Binary:	22 bit fractional arithmetic.
Binary Floating Point: (FPP-33)	33 bit fraction, 10 bit exponents.
Decimal Arithmetic: (BCK)	3N digit decimal operands.
Decimal Floating Point: (CALINT)	3 digit exponent and 9 digit fraction.

INPUT-OUTPUT SYSTEMS, FUNCTIONAL BLOCK DIAGRAM



As many as five of the following input-output systems may be connected to the Control Data 160.

Magnetic Tape System

One to four 163 tape handlers may be associated with the same magnetic tape system. Character transmission rate is 30 kc.

or

One to four 164 tape handlers may be attached to each magnetic tape system. Character transmission rate is 15 kc.

In both cases there is complete compatibility with IBM magnetic tape units.

Typewriter System

A Soroban modified IBM electric typewriter is operated on-line. The computer accepts input at normal typing speeds and prints output data at a rate of 10 characters per second.

1606 High Speed Line Printer System

This output system operates at speeds up to 1000 lines per minute in an alpha-numeric mode.

1609 Punched Card System

Punched card input and output is provided by an IBM 521, or 523 card read-punch unit.

1610 Adaptor

Punched card input is provided by an IBM 088. Punched card output is through an IBM 523. For on line printing the 407 is used.

1607 Magnetic Tape System

In a satellite system, the 1607 magnetic tape system provides communication between a 1604 computer and a 160 computer.

Real Time Clock

A continuously running clock provides programmed real time information.

Communication Line Buffer

Allows direct hook up of Teletype input lines to the 160. Other forms of communication can be handled in similar manner.

THE SATELLITE COMPUTER SYSTEM

INTRODUCTION

General purpose computers have emerged as the most consistently economical and dependable devices in the industry. The reasons behind this emergence are numerous and strongly support new ways of exploiting these computers.

Control Data's two general purpose computers, the small-scale 160 and the large-scale 1604, offer a unique new approach in the Satellite Computer System. It will be as difficult to describe all the potential uses of this system as it is to describe the different uses of the 1604 alone. Some uses should become obvious as the concept is described below.

BASIC SATELLITE COMPONENTS

Figure 1 shows the basic satellite system of the 160 and 1604 computers and the 1607 magnetic tape subsystem. In this combination, the two computers can share the magnetic tape unit properties. The two computers could be connected to each other directly, but with the loss of certain advantages which will become evident.

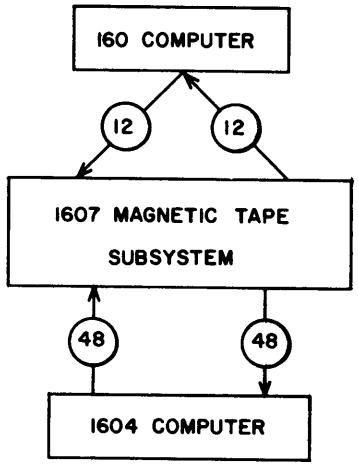


Figure 1. Basic Satellite System

As shown in Figure 1, the 160 communicates on a 12-bit two-way channel, and the 1604 communicates on two 48-bit channels. Dealing first with the 1604, these two 48-bit channels provide the means of writing on tape and simultaneously reading from tape. Within the 1607 there are two complete channel controls, one for the write channel and one for the read channel. These controls are largely independent of each other as shown in Figure 2.

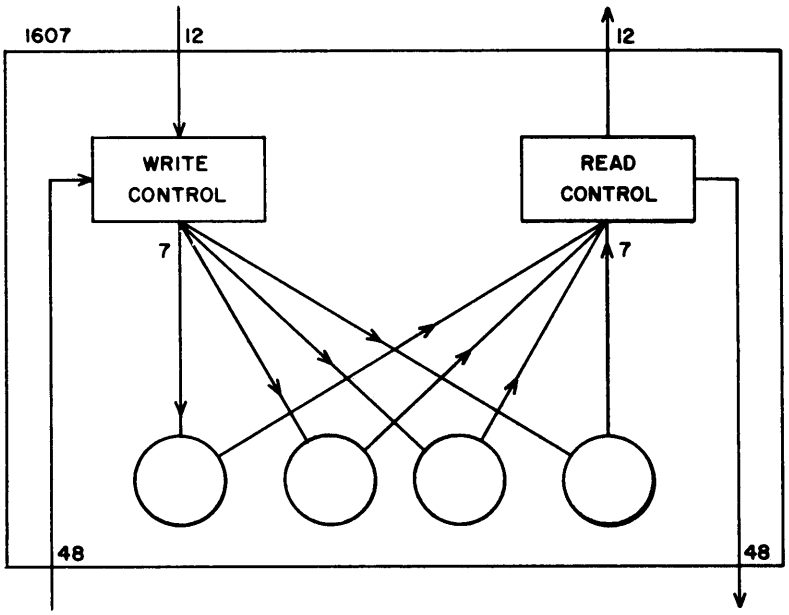


Figure 2. 1607 Controls

The four magnetic tape handlers are shown as circles with connection to the write and read controls. Data is recorded in six bits plus parity, commonly referred to as character-serial. Each 48-bit word is made up of eight characters (six bits each) with no parity. To record one 48-bit word therefore, the word is disassembled as shown in Figure 3, the upper six bits first, etc. A parity bit is added at this point.

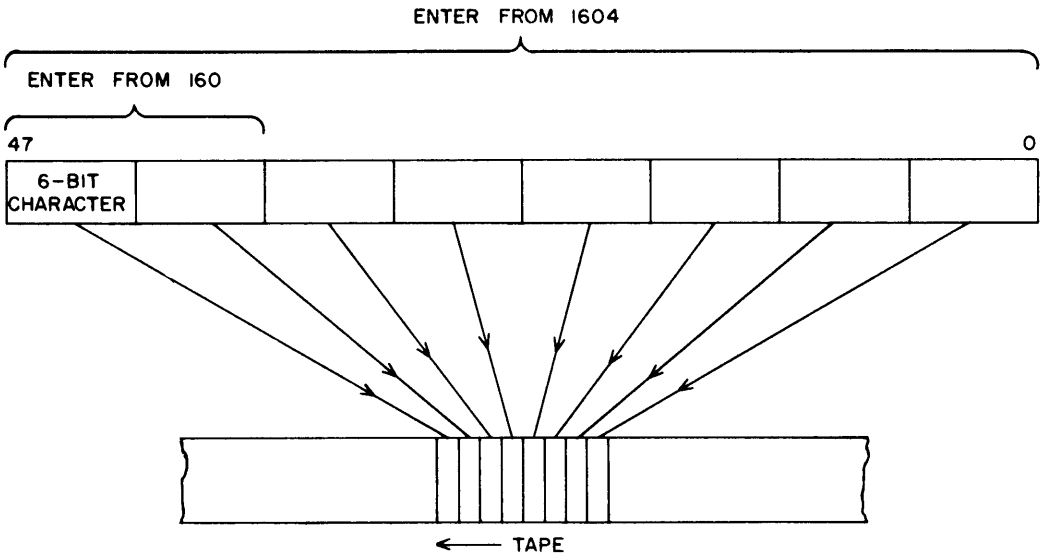


Figure 3. Disassembly of Words Into Characters

To read tape, the characters are read, checked for parity, and assembled into the 48-bit word, first character to the upper six bits, etc. When reading or writing on the tape, one 48-bit word is transferred between the 1604 and 1607 for every eight characters on the tape. The rate of transfer is dependent on the tape speed and the recording density. These are 150 inches per second and 200 per inch respectively. Therefore, the characters pass the head at 30,000 per second and 48-bit words transfer at 3750 per second.

The above system of channel controls in the 1607 are easily adapted to handle 12-bit words from the 160 computer. The 12 bits are positioned in the upper two character positions of the 48-bit register (see Figure 3). In this case, the upper six bits are recorded first, then the lower, followed by another word transfer from the 160. Again characters pass the head at 30,000 per second but 12-bit words transfer at 15,000 per second.

Briefly summarizing the above, the 1607 tape subsystem allows either the 160 or 1604 computers access to the tape handlers. The nature of the read and write controls allows each to be used with a great deal of independence. Therefore, the read channel may be used by the 160 while the write channel is used by the 1604, and vice versa. The read or write control may be used by only one computer at a time.

An added feature of the 1607 not shown in figure 2 is a 6-bit path from the write channel control to the read channel control. This path bypasses tape completely and allows for a direct transfer of data between the computers. When using this path, the write control is assigned to one computer, and the read control is assigned to the other. Twelve-bit words from the 160 are disassembled into characters in the write control, passed to the read control, assembled into 48-bit words and transferred to the 1604. Similarly 48-bit words from the 1604 are disassembled into characters in the write control, passed to the read control, assembled into 12-bit words and transferred to the 160. Since this path does not use magnetic tape, the rate of transfer is dependent only on the read and write controls and the transfer rates of the two computers. These rates are listed below:

1604 Word Rate (8 characters)	5,000/sec. minimum
	50,000/sec. maximum
160 Word Rate (2 characters)	78,000/sec.

This requires the read and write controls to operate at 156,000 characters per second maximum and 40,000 characters minimum.

The following operations are available:

... Transfer from 1604 to 160

... Transfer from 160 to 1604

... Read tape x to 1604 while write tape y from 1604

... Read tape x to 1604 while write tape y from 160

... Read tape x to 160 while write tape y from 1604

... Read tape x to 1604

... Read tape x to 160

... Write tape y from 1604

... Write tape y from 160

SATELLITE PROGRAMMING

Independent Operation

A method of operation is to assign the 1607 tape subsystem to either of the two computers. With this method the effect is to organize two independent computer systems. Programs can run in both systems without fear of interplay between the two. Although it may appear to be a trivial interconnection of equipments, the convenience of placing the 1607 in either system is important.

With an independent operation, the control and transfer of data to and from magnetic tape is accomplished exactly as in a non-satellite system. The following external functions apply:

Select	1604 Code	160 Code	
Select read tape n, binary	320n1	50n1	*
Select read tape n, coded	320n2	50n2	*
Read selected read tape, binary	32001	5001	
Read selected read tape, coded	32002	5002	
Interrupt when selected read tape ready	32004	-	
Rewind selected read tape	32005	5005	
Backspace selected read tape	32006	5006	
Rewind selected read tape, interlock	32007	5007	
Select write tape n, binary	420n1	60n1	*
Select write tape n, coded	420n2	60n2	*
Write selected write tape, binary	42001	6001	
Write selected write tape, coded	42002	6002	
Write end-of-file mark	42003	6003	
Interrupt when selected write tape ready	42004	-	
Rewind selected write tape	42005	6005	
Backspace selected write tape	42006	6006	
Rewind selected write tape, interlock	42007	6007	
Enter status	-	6053	

* Available only when 1607 manually assigned to the 160.

The External Sense instruction of the 1604 obtains specific status information. The 160 computer causes all status indicators to enter as a single 12-bit word, which is examined by the 160 program. The correspondence between the 1604 Sense Codes and the bit position of each indicator in the 160 input word is as follows:

Sense	1604 Sense Code	160 Word
Ready to read	32000-1	X2XX
Read parity error	32002-3	XX4X
Read length error	32004-5	-
End-of-file mark	32006-7	XX1X
Ready to write	42000-1	X1XX
Write reply parity error	42002-3	XX2X
Write reply length error	42004-5	-
End of tape marker	42006-7	XXX4

Operation of the magnetic tape subsystem from either computer entails use of the above SELECT and SENSE codes. Issuance of a SELECT causes that selection to be made and subsequent motion, if any, to begin. Word transfers through the 1604 buffer system or the 160 input-output system, are made at a rate determined in the 1607 tape subsystem. The determination of unit readiness, error, etc. can be made at any time through use of the SENSE codes. Any selection made before the unit is ready is apt to be rejected.

On-Line Operation

A more powerful interconnection of the basic Satellite System allows the assignment of either computer to the tape subsystem under program control. An added feature of this programmed interconnection allows direct word transfers between the memories of the two computers without any tape motion. Programming for this interconnection is necessarily more complicated.

Several additional SELECT and SENSE codes are provided specifically to aid in the on-line sharing of the 1607.

Select	1604 Code	160 Code
Select Read control for 160	32501	5051
Select read control for 1604	32502	5052
Select Write control for 160	42501	6051
Select Write control for 1604	42502	6052
Select Direct 1604 to 160	42503	-
Select Direct 160 to 1604	32503	-
Release Direct selections	42500	-
Select 160 action request	42504	-
Select Interrupt	-	-
Release Interrupt	32505	-
Release 160 action request	-	6050

Sense	1604 Code	160 Code
Read control available	32500-1	4XXX
Write control available	42500-1	2XXX
160 action request	-	XXX2
Direct 160 to 1604	-	1XXX
Direct 1604 to 160	-	X4XX
160 interrupt	32504-5	-

Typical operation of an on-line Satellite System assigns the 1604 as the master control. When a 160 desires a share of the 1607 or a direct transfer, it does so by interrupting the 1604. A typical sequence then follows:

. . . 160 interrupts 1604

. . . 1604 recognizes interrupt and selects a direct transfer from 160 to 1604

. . . 160 recognizes direct selection

. . . The request is transferred from 160 to 1604

. . . Any return communication is set up and transferred

. . . The 1604 releases a read or write control to the 160

When the 160 is finished, it releases the read or write control back to the 1604. Further complication may be introduced if the 1604 allows only a set time period for the 160 to use the control. In that event, the 1604 is able to withdraw the control. The action taken in the 1607 is to complete the current record (or motion).

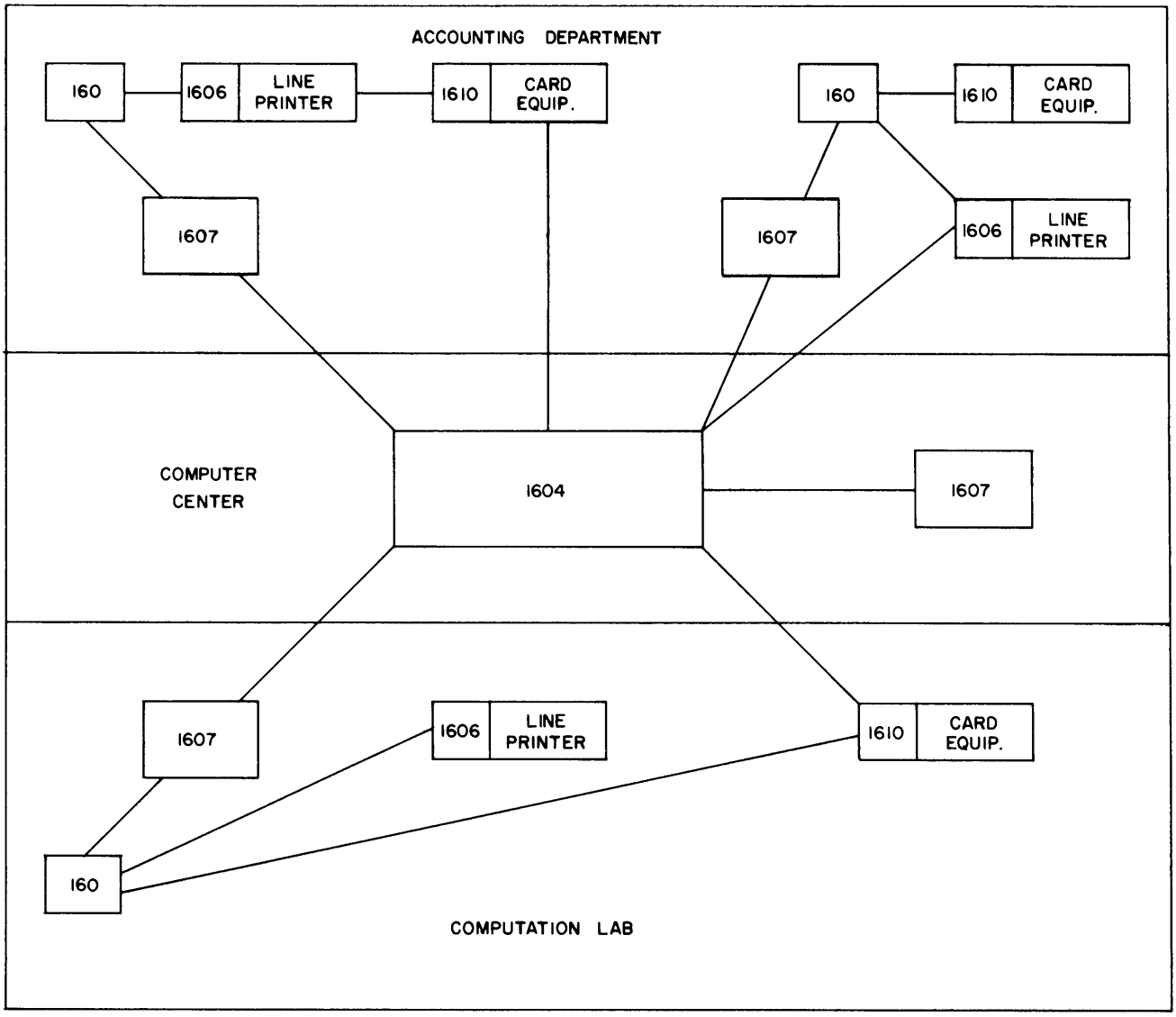


Figure 4. Typical Satellite System

SUMMARY

A summary of main features of the Satellite System:

- ... Large-scale 1604 computer
- ... Up to six small-scale 160 computers
- ... Computers can share magnetic tape units
- ... Direct communication allows access to any part of the system from any other part
- ... Problem preparation on 160
- ... Programmed on-line or off-line 160
- ... Programmed on-line or off-line peripheral units
- ... Use of 160's as external buffer
- ... Handling of tape not necessary between on-line and off-line operations
- ... Accessibility to 160 of full computing capability of 1604
- ... Sharing of peripheral units limits costly duplication
- ... Transfers between 160 and 1604 may be direct or via magnetic tape
- ... Magnetic tape recording and read back at 30,000 characters per second from either computer
- ... Up to 158,000 characters per second direct connection between 160 and 1604

160 STANDARDS

CONTENTS

1. TERMS AND ABBREVIATIONS

1.1 Minimum 160 Computer -- The term "160 Computer" denotes a Control Data 160 have 4,096 words of core storage, a photoelectric paper tape reader, and a high-speed paper tape punch.

1.2 Standard 160 Computer -- The term "Standard 160 Computer" denotes a Control Data 160 having 4,096 words of core storage, a photoelectric paper tape reader, a high-speed paper tape punch, an input/output typewriter and a magnetic tape handler.

2. CHARACTER CODES

2.1 160 Forty-eight Character Codes -- The codes on Table 2.1 shall be used where compatibility with off-line card-to-tape, tape-to-card, and tape-to-line-printer equipment is desired. It should be noted that there is no automatic alteration in the 160 from a BCD code on magnetic tape to the corresponding BIN code, and vice-versa.

If alteration must be performed, for compatibility with tapes produced by other data processing systems or to alter information from a BCD tape to the more-easily-sorted BIN codes, the computer program shall perform the alteration, which consists of complementing the 2^5 bit if the 2^4 bit equals "1" (from left to right, the bits are denoted as 2^5, 2^4, 2^3, 2^2, 2^1, and 2^0).

2.2 160 Sixty-four Character Codes for Anelex Printer -- The set of codes given in Table 2.2 are used in output programs for the Anelex printer.

2.3 Teletype Codes -- For programs producing or processing five-bit Teletype codes, the codes listed in Table 2.3 shall be used. When stored in the computer memory, the codes shall be treated as six-bit codes, by prefixing the five-bit Teletype code by a binary "0".

2.4 Console Typewriter Codes -- For console typewriter alarms, etc., the standard console typewriter codes listed in Table 2.4 shall be used.

2.5 Flexowriter Codes -- For programs employing seven-level paper tape, prepared or listed off-line by means of a Flexowriter, the six-bit Flexowriter codes listed in Table 2.5 shall be used.

TABLE 2.1

48-CHARACTER CODES

CHAR	CARD	BCD	BIN	CHAR	CARD	BCD	BIN	CHAR	CARD	BCD	BIN	CHAR	CARD	BCD	BIN
0	0	12	00	+	12	60	20	-	11	40	40			20	60
1	1	01	01	A	12 1	61	21	J	11 1	41	41	/	0 1	21	61
2	2	02	02	B	12 2	62	22	K	11 2	42	42	S	0 2	22	62
3	3	03	03	C	12 3	63	23	L	11 3	43	43	T	0 3	23	63
4	4	04	04	D	12 4	64	24	M	11 4	44	44	U	0 4	24	64
5	5	05	05	E	12 5	65	25	N	11 5	45	45	V	0 5	25	65
6	6	06	06	F	12 6	66	26	O	11 6	46	46	W	0 6	26	66
7	7	07	07	G	12 7	67	27	P	11 7	47	47	X	0 7	27	67
8	8	10	10	H	12 8	70	30	Q	11 8	50	50	Y	0 8	30	70
9	9	11	11	I	12 9	71	31	R	11 9	51	51	Z	0 9	31	71
=	8,3	13	13	.	12 8,3	73	33	$	11 8,3	53	53	,	0 8,3	33	73
-	8,4	14	14	)	12 8,4	74	34	*	11 8,4	54	54	(	0 8,4	34	74

TABLE 2.2

ANELEX PRINTER CODES

CHAR	CODE	CHAR	CODE	CHAR	CODE	CHAR	CODE
Blank	20	F	66	V	25	$\leq$	15
0	12	G	67	W	26	'	16
1	01	H	70	X	27	$\sqsubset$	17
2	02	I	71	Y	30	$\sqsupset$	32
3	03	J	41	Z	31	$\rightarrow$	35
4	04	K	42	.	73	$\equiv$	36
5	05	L	43	-	40	$\cup$, $\wedge$	37
6	06	M	44	+	60	% or $\vee$	52
7	07	N	45	=	13	\$ or $\neg$	53
8	10	O	46	(	34	$\uparrow$	55
9	11	P	47	)	74	$\downarrow$	56
A	61	Q	50	/	21	>	57
B	62	R	51	*	54	<	72
C	63	S	22	,	33	$\geq$	75
D	64	T	23	:	00	?	76
E	65	U	24	$\neq$	14	;	77

Notes: 1. Codes within heavy lines same as Table 2-1.

2. In last column, codes $\cup$ % \$ appear if business application, $\wedge$ $\vee$ $\neg$ for scientific application.

TABLE 2.3

TELETYPE CODES

LC[1]	FC[2]	CODE[3]	LC[1]	FC[2]	CODE[3]
A	-	30	Q	1	35
B	?	23	R	4	12
C	:	16	S	bell	20
D	$	22	T	5	01
E	3	20	U	7	34
F	!	26	V	;	17
G	&	13	W	2	31
H	#	05	X	/	27
I	8	14	Y	6	25
J	'	32	Z	"	21
K	(	36	space	space	04
L	)	11	CR	CR	02
M	.	07	FIG	FIG	33
N	,	06	LET	LET	37
O	9	03	LF	LF	10
P	o	15			

Notes: 1. "LC" - Letters Case

2. "FC" - Figures Case

3. Octal equivalent of five-bit code given

4. CR - Carriage Return

5. LET - Shift to letters case

6. FIG. - Shift to figures case

7. LF - Line feed

TABLE 2.4

CONSOLE TYPEWRITER CODES

L. C.[1]	U. C.[2]	CODE[3]	L. C.[1]	U. C.[2]	CODE[3]
a	A	30	0	)	56
b	B	23	1	*	74
c	C	16	2	@	70
d	D	22	3	#	64
e	E	20	4	$	62
f	F	26	5	%	66
g	G	13	6	¢	72
h	H	05	7	&	60
i	I	14	8	½	33
j	J	32	9	(	37
k	K	36	-	_	52
l	L	11	/	?	44
m	M	07	,	"	54
n	N	06	+	o	46
o	O	03	.	.	42
p	P	15	;	:	50
q	Q	35	CR	CR	45
r	R	12	UC	UC	47
s	S	24	LC	LC	57
t	T	01	BS	BS	61
u	U	34	=	≠	02
v	V	17	TAB	TAB	51
w	W	31	SPACE	SPACE	04
x	X	27			
y	Y	25			
z	Z	21			

Notes:

1. L.C. = Lower case
2. U.C. = Upper case
3. Code = Octal equivalent of six-bit code
4. CR = Carriage return
5. LC = Shift to lower case
6. TAB = Move to next tabulator stop
7. UC = Shift to upper case
8. BS = Back-space

TABLE 2.5

FLEXOWRITER CODES

UC	LC	CODE	UC	LC	CODE
A	a	30	Y	y	25
B	b	23	Z	z	21
C	c	16	o	0	56
D	d	22	1	1	74
E	e	20	2	2	70
F	f	26	3	3	64
G	g	13	4	4	62
H	h	05	5	5	66
I	i	14	6	6	72
J	j	32	7	7	60
K	k	36	8	8	33
L	l	11	9	9	37
M	m	07	-	-	52
N	n	06	'	/	44
O	o	03	(	)	54
P	p	15	+	,	46
Q	q	35	=	.	42
R	r	12	:	;	50
S	s	24	CR		45
			Upper Case (UC)		47
T	t	01	Lower Case (LC)		57
			Back Space (BS)		61
U	u	34	Color Shift (CS)		02
			Tabulate (TAB)		51
V	v	17	Stop		43
			Space		04
W	w	31	Tape Feed		00
			Delete		77
X	x	27			

Notes: 1. Leader = Blank Tape, Delete = Deleted Character,
Stop = Stop Flexowriter reader,
2. 10, 40, 41, 53, 55, 63, 65, 67, 71, 73, 75, and 76 - illegal

3. INPUT/OUTPUT FORMATS

3.1 Card Formats -- The standard card, for programming purposes, shall be the 80-column, 12-row card in which information is represented by a pattern of rectangular holes. Alphanumeric data (letters, numerals, and special symbols) shall be entered or punched on standard 80-column cards in the card codes described in Section 2.

3.2 Magnetic Tape Formats -- The standard magnetic tape shall be one-half inch plastic tape, carrying data on six levels and either an odd or an even parity (check) bit on a seventh level. Recording shall be in the form of variable-length records. The load point (beginning) and end of the tape shall be marked by reflective coatings.

A special mark, consisting of an even-parity 17 code, shall be used to denote "end-of-file".

No restrictions shall be placed on record length or format other than a record shall be more than one character long; however, when optional tape or card input or output is desired, the information written onto the tape, when examined frame-by-frame, should duplicate the column-by-column (or half-column-by-half-column) information on the corresponding card format.

3.3 Paper Tape Formats -- The standard paper tape shall be non-oiled, 0.875" wide seven-level perforator tape.

3.3.1 Program Load Paper Tape

The program load format requires that the first two frames following the blank tape leader contain the insert address of the block of data. The first frame of the insert address must also contain a seventh level hole. The two frames following the insert address must be blank. The block of bi-octal data is then recorded on the following frames with the first frame of each word containing a seventh level hole. At the end of the data block, up to ten blank frames of tape are allowed before the next insert address without coming to a program stop.

3.3.2 Binary Paper Tape -- Binary paper tape, sometimes called "bi-octal". Bi-octal paper tape can be read directly into the computer. Place initial address into the P register, set Load-Clear switch to LOAD and press the Run switch. Every other frame contains a seventh level hole. The absence of the seventh level hole terminates the read. P is advanced by one for each new word read. P is advanced by one for each new word read, with the exception of the first.

3.3.3 Octal Paper Tape -- The octal paper tape format, often called the "Flex-Load" format, is produced by or is reproducible on a Flexowriter.

4. PROGRAMMING STANDARDS

4.1 Memory Allocations

4.1.1 Temporary Storage -- Computer locations 0070 through 0077 (octal) shall be used for temporary storage. Storage locations 0-77 are unique in their use for direct and indirect addressing because they can be referred to by an instruction any place in storage. These locations should be reserved for use as indirect addresses and as counters which are used by more than one routine. Certain locations should be reserved for subroutine return addresses.

To maintain compatibility with Control Data Corporation programming aids the following conventions will be observed:

1) Locations 70 to 77 will be used as transient locations within a subroutine. The subroutines will have to preset these locations before using them.
2) Location 0 to 1 will be used as the entry to a program where the instructions are:

Locations	Contents
0000	7001
0001	Address of Program

3) Locations 2 through 7 will be used for subroutine exits. Location 7 is the exit for the highest-order subroutine, 6 the next lowest, and so forth.

4) Locations 10 to 67 may be used as permanent storage by any program.

4.1.2 Service Routines -- Service routines will be distributed in relocatable binary or symbolic assembly language.

4.1.3 Resident Service Routine -- Computer locations 7400-7776. Locations 7400 to 7776 are used by service routines. Any programmer wishing to use these services should not use these locations in his program.

4.2 Subroutine Entry and Exit

4.2.1 Subroutines shall be entered with A containing the address of the first parameter. Additional parameters will be in successive locations.

Return shall be to the location following the last parameter.

4.3 Alarms -- Any alarm routine shall first store the contents of all registers, next perform the alarm operations, and finally restore the registers.

5.1 Program Abstracts -- Abstracts shall contain the following:

160 identification (as specified in Section 5.5)
Purpose (Brief)
Space Required

5.2 Program Descriptions (Detailed)

5.3.1 The published material for each routine shall constitute a distinct package, separate from material for all other routines; this shall be done to facilitate revisions and republication of the material for one routine without the necessity for republishing others.

5.3.2 The published material for each routine shall consist of the following:

A. IDENTIFICATION
Title
Identification (as specified in Section 5.5)
Category
Programmer
Date

B. PURPOSE -- A BRIEF DESCRIPTION

C. USAGE

1. Calling Sequence, or Operational Procedure
2. Arguments, or Parameters
3. Space Required (decimal and octal)
4. Temporary Storage Requirements (decimal and octal)
5. Alarms, or Print-Outs
6. Error Returns, or Error Codes (left in accumulator or elsewhere)
7. Error Stops
8. Input and Output Tape Mountings
9. Input and Output Formats
10. Timing
11. Accuracy
12. Cautions to User (e.g., do not step through routine, etc.)
13. Equipment Configuration
14. References

D. METHOD OR ALGORITHM -- A BRIEF DESCRIPTION OF THE MATHEMATICS

E. FLOW CHART

If any of the above items are not applicable, the item will not appear. The numbering system will be maintained.

5.2.3 The master copy of the published material for a routine shall be furnished to Control Data Corporation in a form that they specify.

5.2.4 Each routine shall also be stored and distributed in relocatable binary or symbolic assembly language.

5.3 Identification

Each program shall be identified on the program description and on the abstract by a designator consisting of the following parts:

1. Classification Code
2. Sequence number
3. 160 reference number

5.3.1 Classification Code

The classification code shall consist of letters which identify the class to which the routine belongs according to the following list:

1. L - Library routine
2. RS - Resident service routine
3. S - Service routine
4. T - Test routine

5.3.2 Sequence Numbers

Three digits. The sequence number is assigned according to the order in which it was submitted.

5.3.3 Reference number

The reference number consists of the number 160 with a decimal point followed by 3 digits which are assigned in order of origination of the routine. The reference number is an index number which can be used in ordering or refering to any of the routines.

A. IDENTIFICATION

160.010

TITLE: Tape Leader Preparation

IDENTIFICATION: S 001 (Replaced by S 024)

CATEGORY: Service Routines

PROGRAMMER: J. A. Pederson

DATE: August 1960

B. PURPOSE

Used to prepare tape leaders which identify program tapes. The characters 0 to 9, as S,T,P,R ,-,., can be punched on a tape leader.

C. USAGE

1. Operational Procedure

a. Load program starting at location zero.

b. Master Clear, turn on the punch and start. The program will punch two frames of leader.

c. Place in A the desired character to be punched from the following table.

d. Push Run switch. The program will punch the character and stop.

e. After preparing the leader on the punch, run out about two inches and then the desired program may be copied.

If the program is internal the resident service routine may be used to punch out the information from core storage. By clearing and starting at location 0050, the information on the photoelectric paper tape reader may be repunched on the high speed punch.

Character Code in A Register:

0	0
1	1
2	2

Character Code in A Register (Cont.)

3	3
4	4
5	5
6	6
7	7
8	10
9	11
	12
S	13
T	14
P	15
-	16
R	17

3. Space required $311_8 = 201_{10}$ locations

4. Temporary Storage Requirements - 7 locations

10. Timing - Output approximately 60 frames per second

14. Equipment Configuration - Minimum 160 computer

A. IDENTIFICATION 160.011

TITLE: Duplicate Paper Tape

IDENTIFICATION: S 003

CATEGORY: Service Routines

PROGRAMMER: J. A. Pederson

DATE: August 1960

B. PURPOSE

This routine duplicates information on paper tape from the photoelectric reader to the high speed punch.

C. USAGE

1. Operational Procedure - The routine floats and may be loaded anywhere in memory.

   a. Load, set P = any arbitrary address.

```
7507   select reader
7210   read 2 frames
0072
7505   select punch
7305   punch 2 frames
0072
6506   go back
4102   reader code
4104   punch code
0070   start of I/O
```

   b. Master Clear

   c. Set P = starting address selected

   d. Turn on reader and punch

   e. Place tape to be duplicated in the reader

f. Press Run switch

g. To stop duplication return switch to center. The duplication process stops automatically after the tape in the reader has been read.

3. Space required - $12_8 = 10_{10}$ locations

13. Equipment Configuration - Minimum 160 computer.

A. IDENTIFICATION 160.012

TITLE: Paper Tape Duplicator

IDENTIFICATION: S 013 (Replaced by S 023)

CATEGORY: Service Routines

PROGRAMMER: L. Kuller

DATE: August 1960

B. PURPOSE

The program is used to generate and verify copies of an original paper tape.

C. USAGE

1. Operational Procedure

1. Load the tape containing S 013 beginning at location zero.
2. Load Mode Stop

P=0072

A=0073

The program functions in three modes.

Load Tape - Makes an image of the tape in the memory of the computer.

Verify Tape - Compares the original tape, or any of the copies, with the image of the original tape stored in memory.

Punch Tape - Punches duplicate paper tapes from the image stored in the computer memory.

a. Load Tape

1. Master Clear
2. Insert tape to be duplicated into reader
3. Press Run switch.
4. Program Stop

P=0034

A=0000

Z=7707

Program Stop occurs after the computer has read twenty blank frames of tape. To continue on the same tape push the Run switch after returning it to the center position. Restarting may be done as many times as necessary unless an overflow occurs.

5. Overflow condition - More than 4037 frames of punched paper tape have been entered.

   P=0023

   A=0000

   Z=7750

   After an overflow stop the program readies itself to read a new tape into the memory if the Run-Step switch is returned to the center position and then to the RUN position.

b. Verify Tape

1. Set P=0001
2. Turn on reader and insert tape to be verified.
3. Press Run switch
4. Discrepancy stop

   P=0064

   A=the image for that frame in memory

   Z=0007

   The program will continue to check the remainder of the tape if the Run switch is returned to the center position and then to the RUN position.

5. Program Stop

   P=0054

A=0000

Z=7720

The Punch Tape mode is selected after the program stop by returning the Run-Step switch to the center position and then to the RUN position.

c. Punch Tape

1. Start Punch Motor
2. Set P to 0002
3. Press Run switch
4. Computer stop

P=0060

A=last address in the field

Z=7710

The Verify Tape mode is selected after the program stop by returning the Run-Step switch to the center position and then to the RUN position

3. Space Required - $72_8 = 58_{10}$ locations

13. Equipment Configuration - Minimum 160 Computer.

A. IDENTIFICATION 160.013

TITLE: Resident Service Library (SLOOP)

IDENTIFICATION: RS 016 (Replaced by RS 022)

CATEGORY: Service Routines

PROGRAMMERS: L. Kuller

DATE: July 1960

B. PURPOSE

These routines enable the operator to read instructions into the 160 memory from punched paper tape, dump the contents of the memory on punched paper tape or on the on-line typewriter, and verify the accuracy of the punched paper tape. The resident service library of the 160 computer is composed of the following service routines:

Program Load	Bi-Octal Punch
Program Punch	Bi-Octal Verify
Program Verify	Flex Dump

C. USAGE

1. Operational Procedure

Load the tape containing SLOOP starting at location 7400 with the Load-Clear switch in the LOAD position. The P Register will contain 7776 when loading is completed.

a. Program Load - Program Load is used to enter blocks of data in program load format into the memory of the 160 computer from punched paper tape.

1. Set the P Register to 7400
2. Turn on the reader and insert tape
3. Press the Run switch
4. Program Stop

A=0000

P=7767

Z=7707

5. Format Error

P=7755

A=0000

Z=0007

After a format error tape must be removed from the reader because the program will not accept the remainder of the tape.

The Program Load format requires that the first two frames following the blank tape leader contain the insert address for the block of data. The first frame of the insert address must also contain a seventh level hole. The two frames following the insert address must be blank. The block of bi-octal data is recorded on the following frames with the first frame of each word containing a seventh level hole. At the end of each data block, up to ten blank frames are allowed before the next insert address without coming to a program stop.

b. Program Punch - Program Punch is used to punch out portions of the 160 memory on paper tape in a form suitable for reloading by the service routine Program Load.

1. Start Punch Motor
2. Set P Register to 7401
3. Place the initial address of the region to be punched in A Register.
4. Press Run switch

5. Computer will stop with

   P=7607

   A=0000

   Z=7701

6. Place the address of the last word to be punched in the A Register.

7. Press Run switch; program will punch blocks of 64 words where the block division is at addresses divisible by 64, i.e., octal XX00. A blank leader and trailer will also be punched.

   Program Stop - Operation completed

   P=7604

   A=0000

   Z=7707

c. Program Verify - Program Verify is used to compare the data on punched paper tape in the Program Load format with the current contents of the 160 memory.

1. Set the P Register to 7402
2. Place tape in the reader which must be turned on.
3. Press Run switch.
4. Discrepancy - Computer stops. A contains the address of the discrepancy. Return Run switch to center position and press again to RUN. Program will check the remainder of the tape.
5. Format Error - Computer stops. Remove the tape from the reader because the program will not accept the remainder of the tape.
6. At the end of a data block, up to ten blank frames are allowed before the next insert address without coming to a program stop.

Discrepancy

P=7731

A=address of discrepancy

Z=0001

Format Error

P=7755

A=0000

Z=0007

Program Stop

P=7767

A=0000

Z=7707

d. Bi-Octal Punch - Used to punch out portions of the 160 memory on paper tape in bi-octal format.

1. Start Punch Motor
2. Set P Register to 7403
3. Set A Register to initial address of the region to be punched
4. Press Run switch
5. Computer will stop with

   P=7417

   A=0000

   Z=7701
6. Set A Register to last address to be punched
7. Press Run switch again
8. Program Stop

   P=7414

   A=0000

   Z=7707

The program will punch a blank leader followed by the specified region of memory and a blank trailer.

e. Bi-Octal Verify - Used to compare bi-octal data on punched paper tape with the current contents of the 160 memory.

1. Set P=7404
2. Set A=location of first data word on the paper tape
3. Turn on reader and insert paper tape
4. Press Run switch. Tape will be checked against current contents of the computer's memory.
5. Discrepancy - Computer stops.

   P=7731

   A=address of the discrepancy

   Z=0001

   To continue return switch to neutral and then RUN.

6. Format Error

   P=7755

   A=0000

   Z=0007

   Remove tape because program will not accept the remainder

7. Program Stop

   P=7767

   A=0000

   Z=7707

   Occurs when the eleventh blank frame on the trailer is encountered.

f. Flex Dump - Produces a punched paper tape of a program suitable for off-line listing on a Flexowriter, or to print the listing directly on the on-line typewriter.

1. Set P=7405
2. Set A=initial address of region to be listed
3. Press Run switch
4. The computer will stop with

   P=7462

   A=0000

   Z=7701
5. Set A=last address to be listed
6. Press Run switch again
7. Program Stop

   P=7466

   A=0000

   Z=7702
8. Set program option in A Register

   A=0000 Punch listings on paper tape for off line printing

   A=0001 Punch listings on paper tape with tab code and stop code following each listing.
9. Turn on proper output device
10. Press Run switch again

Flex Dump will produce the specified listing allowing up to 48 listings on each page, with a double space following each listing with an address divisible by eight. Each listing shows the address where the information was stored (4 digits) followed by the information (4 digits). A stop code is punched at the end of each page when a punch option is chosen. When the on-line typewriter option is selected, computer operation stops at the end of each page to allow the operator to change paper. When

this occurs P=7545; A=0002; Z=7703. When a punch option is chosen a blank leader and a trailer are punched in addition to the listings.

3. Space Required - $376_8 = 254_{10}$ locations
4. Temporary Storage Requirements - Locations 0070 through 0077
5. Alarms or Print Outs - Discrepancy or format error. See discussion of the routines.
7. Error Stops - Machine stops on discrepancy or format error. For octal display on the console see discussion of routines.
8. Input/Output Tape Mountings - Paper Tape Punch under program control, P/T Reader under load mode or program control. See discussion of routines
9. Input/Output Format - Output controlled by program. Input bi-octal format or program load format.
10. Timing - Output approximately 60 frames per second. Input approximately 350 frames per second.
12. Caution to user - Locations 7400 to 7776 are used by the service routines which make up SLOOP. Do not use these locations in the program. The contents of location 7777 cannot be loaded, verified, punched or listed by any of the service routines in SLOOP unless it is the first location of the selected region.

    Temporary storage location contents may be altered if a service routine in SLOOP is used.
14. Equipment Configuration - Minimum 160 Computer.

A. IDENTIFICATION 160.014

TITLE: Flexo

IDENTIFICATION: RS 017 (Replaced by RS 022)

CATEGORY: Service Routines

PROGRAMMERS: L. Kuller

DATE: August 1960

B. PURPOSE

1. Read instructions into the memory from punched paper tape prepared on a Flexowriter.
2. Dump the contents of memory on punched paper tape for off-line listing in a Flexowriter.
3. Dump the contents of memory directly on the on-line typewriter.
4. Verify the accuracy of punched Flexowriter tape.
5. Dump the contents of memory on punched paper tape in bi-octal format.

The following Service Routines are included in Flexo:

Flex Load

Flex Dump

Flex Verify

Bi-Octal Punch

C. USAGE

1. Operational Procedure -

Load the tape containing Flexo starting at location 7400 with the Load-Clear switch in LOAD position. The P and A Register will contain 7776 when loading is completed.

a. Flex Load - This program reads 160 tapes prepared on a Flexowriter or by the Flex Dump program and stores them in the locations as specified by the tape.

1. Turn on reader and insert flex tape.
2. Set P=7400
3. Press Run switch
4. Program Stop

   P=7711

   A=7711 or 0000

   Z=7707

b. Flex Dump - Used to produce punched paper tape listings of a program suitable for off-line printing on a Flexowriter, or to print the listing directly on the on-line typewriter.

1. Set P=7405
2. Set A=initial address of the region to be listed
3. Press Run switch
4. Computer will stop with

   P=7466

   A=0000

   Z=7702

5. Set the program option in A as follows:

   A=0000 Punch listings on paper tape for off line printing

   A=0000 Punch listings on paper tape with tab code and stop code following each listing.

   A=0002 Print listings on the on-line typewriter

6. Turn on the proper output device
7. Press Run switch

   If the typewriter is selected the computer will stop at the end of each page to permit change of paper.

   P=7545

A=0002

Z=7703

To continue the program return the Run switch to center position and then to RUN position.

c. Flex Verify - The program is designed to check the 160 tapes prepared on a Flexowriter or by the Flex Dump program against the current contents of the 160 storage locations specified by the tape.

1. Turn on reader
2. Insert tape on the blank leader
3. Set P=7401
4. Press Run switch
5. Discrepancy - Computer stops

   P=7732

   A=address where discrepancy was found

   Z=0001
6. Program Stop

   P=7711

   A=0000 or 7711

   Z=7707

d. Bi-Octal Punch - Used to punch out portions of the 160 memory on paper tape in a form suitable for re-loading with the Load-Clear switch in the LOAD position.

1. Start Punch motor
2. Set P=7403
3. Set A=initial address of the region to be punched
4. Press Run switch

5. Computer stops

   P=7417

   A=0000

   Z=7701

6. Set A=last address of last word to be punched

7. Press Run switch after returning it to center postion

8. Program Stop

   P=7414

   A=0000

   Z=7707

3. Space Required - $302_8 = 194_{10}$ locations

4. Temporary Storage Requirements - Locations 0070 through 0077

7. Error Stops - See Flex Verify for Discrepancy stop

13. Caution to User - The service routines which make up SLOOP cannot be in the 160 memory simultaneously with Flexo. Flexo uses locations 7400 through 7776 and they should not be used by any program.
    The contents of location 7777 cannot be punched by Bi-Octal Punch or listed by Flex Dump unless it is first location of a selected region. Location 7777 can not be loaded or verified by Flex Load and Flex Verify unless it is referenced in eight digit format. If location 7777 is referenced in the four digit format the data will be stored at location 0000.

14. Equipment Configuration - Minimum 160 Computer. If Flex Dump with the typewriter option is chosen an electric typewriter must be provided.

15. References - See RS 022 which replaces RS 017.

A. IDENTIFICATION

TITLE: Flex Load and Flex Verify

IDENTIFICATION: S 018

CATEGORY: Service Routines

PROGRAMMER: L. Kuller

DATE: September 1960

B. PURPOSE

Flex Load is designed to read 160 program tapes prepared on a Flexowriter or by the Flex Dump program and store them in memory at addresses specified by the tape.

Flex Verify is designed to check 160 program tapes prepared on a Flexowriter or by the Flex Dump program against the current contents of the 160 storage locations as specified by the tape. The program indicates any discrepancies that are found.

C. USAGE

1. Operational Procedure

a. Turn on reader

b. Insert tape someplace on blank leader

c. Flex Load set P=0400

Flex Verify set P=0401

d. Press Run switch

e. Discrepancy stop (Flex Verify)

P=0434

A=address of the discrepancy

Z=0001

f. Program Stop

P=0414

Z=7707

3. Space Required - $177_8 = 127_{10}$ locations

4. Temporary Storage Required - Locations 0072 through 0077

12. Caution to User - Location 7777 can not be loaded or verified by these programs unless it is referenced in the eight digit format. If 7777 is referenced in the four digit format the data will be stored at location 0000.

13. Equipment Configuration - Minimum 160 Computer

15. References - See RS 022 for a discussion of tape preparation and format.

A. IDENTIFICATION

160.021

TITLE: Flex Tape to Magnetic Tape Converter

IDENT NUMBER: S 019

CATEGORY: Conversion Routine

PROGRAMMER: L. Kuller

DATE: August, 1960

B. PURPOSE

Produce a copy of a Flexowriter tape on magnetic tape in a form suitable for listing.

C. USAGE

1. Operational Procedure

a. Load program tape S 019

1. Turn on reader, insert tape and set P=0000.

2. Set Load switch and press Run.

3. First stop

P=0004

A=1111

Z=0000

4. Set P=1000, press Run switch

5. Program Stop

P=1177

A=0000

Z=0000

b. Convert Flexowriter tape to magnetic tape

1. Turn on reader and insert Flexowriter tape.

2. Set CODED parity selection on magnetic tape unit.

3. Set P=1000

4. Press Run switch

5. Program Stop (16 consecutive blank frames read)

   P=1021

   A= 0000

   Z=7707

6. Option-write end-of-file mark, clear computer and press Run switch.

7. To convert more Flexowriter tape, return Run switch to center and then to RUN.

3. Space required - $204_8 = 132_{10}$ locations

7. Error - parity, routine attempts to write output block on tape until the error disappears.

13. Equipment Configuration - Minimum 160 computer with magnetic tape units.

D. METHOD

Punched paper tape characters are translated into equivalent line printer codes. If there is no equivalence, the punched paper tape character is ignored. When a CR is read, blank codes to fill the 120 character line printer line are inverted on the magnetic tape. The output from the 160 is copies on magnetic tape with parity checking.

A. IDENTIFICATION 160.015

TITLE: Floating Bi-Octal Punch and Floating Bi-Octal Verify

IDENTIFICATION: S 020

CATEGORY: Service Routines

PROGRAMMERS: L. Kuller

DATE: September 1960

B. PURPOSE

Floating Bi-Octal Punch is used to punch out portions of the 160 memory in bi-octal format. Floating Bi-Octal Verify compares bi-octal data on punched paper tape with the current contents of the 160 memory.

C. USAGE

1. Operational Procedure - Load the tape beginning at any location except 7766 through 0077 with the Load-Clear switch in LOAD position

   a. Floating Bi-Octal Punch

      1. Start Punch motor
      2. Set P=initial address of the floating bi-octal routine
      3. Set A=initial address of the region to be punched
      4. Press Run switch
      5. Computer stops

         A=0000

         Z=7701

      6. Set A=last address to be punched
      7. Press Run switch after returning to center position
      8. Program Stop

         A=0000

         Z=7707

   b. Floating Bi-Octal Verify

1. Set P to the third location of the region containing the floating bi-octal routines
2. Set A=first data word location on the tape
3. Turn on reader and insert tape
4. Press Run switch
5. Discrepancy Stop

   A=address at which discrepancy was found

   Z=0001
6. Format error on the tape

   Z=0007

   The tape must be removed because the computer will not accept the remainder.
7. Program Stop

   A=0000

   Z=7707

3. Space Required - $106_8 = 70_{10}$ locations
4. Temporary Storage Requirements - locations 0074 through 0077

12. Caution to User - The contents of location 7777 can not be verified or punched unless it is the first location of the selected region. The tape containing the bi-octal routines can not be loaded beginning at locations 7776 through 0077 since some part of the floating bi-octal will overlap temporary storage regions.

13. Equipment Configuration - Minimum 160 Computer

A. IDENTIFICATION 160.017

TITLE: Convert Binary Coded Decimal to Binary

IDENTIFICATION: L 021

CATEGORY: Information Processing

PROGRAMMERS: J. Pederson

DATE: August 1960

B. PURPOSE

This subroutine will convert a binary coded decimal number of up to six digits to the equivalent binary number in 22 bit arithmetic format. The binary coded decimal number is stored one digit per word with the digit as the low order four bits of the word.

C. USAGE

1. Operational Procedure

The address of the high order digits of the decimal number is specified by the contents of storage location 0010. The number of digits in the decimal number is specified by the contents of storage location 0011. The resulting binary number will be stored in location 0012 and 0013 with the high order portion of the word in 0012.

The 22 bit arithmetic format uses the high order bit of the low order word as a buffer to catch overflows, thus this bit must be a zero and is not considered as an information bit. The low order bit of the high order word will be considered as the 2^{11} bit.

On completion, the contents of location 10 will point one location beyond the low order digit of the decimal number. Contents of 0011 will be unchanged. The routine uses the high order four bits of location 0013 to catch the overflow from the multiply process. This overflow is added to the low order four bits of 0012. On completion of the routines,

the program reassembles words 0012 and 0013 to the 22 bit arithmetic format. The assumption in the conversion routine is that the numbers are positive. Any sign indication will have to be added later.

2. Argument or Parameters - None

3. Space Required - 37 locations

4. Temporary Storage Requirements

   0010 - Contains address of high order digit of decimal number

   0011 - Contains count of number of digits in decimal number

   0012 - High order binary result

   0013 - Low order binary result

   0070 - Counter used in routine

   0077 - Mask (0377)

10. Timing - The routine takes 230.4 + 198.4 N microseconds where N is the number of decimal digits to be converted. The routine will take approximately 1.2 milliseconds to convert a five decimal digit number. The time to convert binary coded decimal information to the corresponding binary information is given in the table below. These times are derived on the assumption that the binary coded decimal information is stored one digit per word with the high order digit of the number given first.

    The break in the time sequence between 3 and 4 digits, 6 and 7 digits, 9 and 10 digits is based on changing from a single to double to triple to quadruple precision binary representation.

CONVERSION TIME

Number of decimal digits	Conversion time in milliseconds
2	0.11
3	0.18

CONVERSION TIME (Cont.)

Number of decimal digits	Conversion time in milliseconds
4	1.03
5	1.23
6	1.43
7	2.6
8	2.9
9	3.4
10	5.0

11. Accuracy - The routine is good for up to six decimal digit numbers.

13. Equipment Configuration - Minimum 160 Computer.

A. IDENTIFICATION 160.018

TITLE: Resident Service Library

IDENTIFICATION: RS 022 (Replaces RS 017)

CATEGORY: Service Routines

PROGRAMMERS: L. Kuller

DATE: December 1960

B. PURPOSE

Selecting the proper routine enables instructions to be loaded into the memory from punched paper tape or the on-line electric typewriter; or the contents of the memory to be dumped on punched paper tape or the on-line typewriter. The accuracy of the punched paper tape may be verified and the check sum of a block of memory may be determined. The contents of memory exclusive of RS 022 may be cleared.

The following service routines comprise the resident service library:

Check Sum	Flex Tape Punch	Clear Memory
Bi-Octal Punch	Flex Tape Verify	
Bi-Octal Verify	Type Load	
Flex Tape Load	Type Dump	

C. USAGE

1. Operational Procedure - Load the tape of RS 022 as follows:

    1. Set P=7400
    2. Set Load-Clear switch in LOAD position
    3. Program Stop

        P=7776

        A=7776
    4. If Check Sum is desired: Select Check Sum Program and sum from location 7400 through 7776; If RS 022 has been correctly loaded a check sum of 0160 will be obtained.

a. Check Sum - Used to determine if the data or instructions are correctly stored in a specified region of memory. A check sum is performed by summing all the words within a region.

1. Set P=7400
2. Set A=initial address of the region
3. Press Run switch
4. Computer stops with P=7610, A=0000, Z=7701
5. Set A=last address of the region
6. Press Run switch again after returning it to center position
7. Program Stop

   P=7604

   A=Check sum of the region

   Z=7707
8. Compare A with known check sum
9. To rerun program clear A and proceed from Step 2 above.

b. Bi-Octal Punch - Used to punch out portions of the 160 memory in bi-octal format.

1. Start Punch motor
2. Set P=7401
3. Set A=initial address of the region
4. Press Run switch
5. Computer stops with P=7624; A=0000; Z=7701
6. Set A=last address to be punched
7. Press Run switch after returning it to center position
8. Program Stop

   P=7621

   A=0000

   Z=7707

9. To repeat, go back to Step 3 above.

c. Bi-Octal Verify - Bi-Octal Verify is used to verify the accuracy of a tape prepared by Bi-Octal Punch.

1. Set P=7402
2. Set A=location of first data word on the tape
3. Turn on reader and insert tape anywhere on the blank leader
4. Press Run switch. Computer checks the data on the tape against the current content of the memory. The format is also verified.
5. Computer stops on finding a discrepancy or format error:

   P=7441

   A=address where error occured

   Z=0001

   To continue checking the tape press Run switch after returning it to center position.
6. Program Stop

   P=7432

   A=0000

   Z=7707
7. To repeat the program insert new tape into reader, set initial address and press Run switch after returning it to center position.

d. Flex Tape Load - Used to read 160 program tapes prepared on a Flexowriter, by the Flex Tape Dump program or an OSAP listing and store in the storage locations specified by the tape.

1. Turn on reader
2. Insert tape someplace on the blank leader
3. Set P=7403

4. Press Run switch. Program will store and check the information following the first carriage return character on the tape.

5. Computer stops on finding a discrepancy:

   P=7515

   A=address at which the discrepancy occured

   Z=0001

6. Program Stop:

   P=7470

   A=0000

   Z=7707

7. Repeat the program by putting a new tape in the reader. Push Run switch after returning it to the center position.

Characters 0 through 7, carriage return, tab, and period are recognized by the program. The illegal code 75 will cause computation to stop as explained in the paragraph on limitations (see No. 13 - Caution to user). The illegal code 76, which can be punched by OSAP, causes the program to ignore all characters on that line. All other characters are ignored by the program. The carriage return defines the beginning of a line.

Tape may be prepared in an eight digit format giving the four digits of the address followed by the four digits of the instruction or constant. A space or tab may or may not be inserted between the address and the data in this format. The program always takes the last four characters and stores them at the address specified by the first four characters.

Tape may also be prepared in a four digit format in which a carriage return followed by a tab followed by four characters of data make up the line. The program will take the four characters and store them at the next consecutive memory location following that at which information was last stored.

If an insufficient number of digits in either format is present on a line, the data on that line will be discarded. A period appearing after a carriage return or a tab is interpreted as an end of file code and will cause the program to stop.

Example of format accepted by Flex Tape Load, Flex Tape Verify and Type Load

The following format will be accepted.

1463	30 51	Comments on the program
	41 63	
	5403	No periods are allowed except as an end of file mark
1502	6552	
	43 05	
4506	0112	
5602		
	0273	
	1783	
		The period causes the program to stop.

The Flex Tape Load program would process the above tape and cause the following information to be stored in memory:

Location	Contents
1463	3051
1464	4163
1465	5403
1502	6552
1503	4305
4506	0112
5602	0273

The last entry is missing because the program ignored the 8 and rejected the data because only three recognizable characters appeared on the line.

e. Flex Tape Punch - Used to produce punched paper tape listings of a program suitable for off-line printing on a Flexowriter.

1. Turn on Punch
2. Set P=7404
3. Set A=initial address
4. Press Run switch
5. Computer stops with

   P=7677

   A=0000

   Z=7701
6. Set A=address of last word to be listed
7. Press Run switch after returning it to center position

Flex Tape Punch will produce the specified listing allowing up to 48 listings on each page, with a double space preceding each listing with an address divisible by eight. Each listing shows the address where the information was stored (4 digits) followed by a tab, followed by the information (4 digits). A stop code is punched at the end of each page. A blank leader and trailer are punched in addition to the listings.

8. Program Stop

   P=7674

   A=0000

   Z=7707
9. To repeat insert initial address in A and press Run switch after returning it to center position.

f. Flex Tape Verify - Used to verify the accuracy of a tape prepared

by Flex Tape Punch.

1. Turn on reader
2. Insert Flexowriter tape into reader someplace on the blank leader.
3. Set P=7405
4. Press Run switch. On encountering a carriage return the program will begin assembling information according to the format rules given under Flex Tape Load. It will check this information against the current contents of the memory.
5. Discrepancy - Computer stops

   P=7515

   A=address of discrepancy

   Z=0001

   Press Run switch after returning it to center position.

   Program will continue to check the rest of the tape.
6. Program Stop

   P=7470

   A=0000

   Z=7707
7. To repeat the program insert a new tape in the reader and press the Run switch after returning it to center position.

g. Type Load - Used to store data received from the on-line electric typewriter.

1. Turn on the electric typewriter
2. Put Operation Mode switch into CLEAR position
3. Press Operation Mode switch into COMPUTER position. The ready light should be on and the Input Request light off.

4. Check to ensure Input Disconnect switch is in center position.

5. Set P=7406

6. Press Run switch. The status indicator will immediately show an input (IN) condition. The input request light and the ready light on the typewriter cabinet will also be on.

7. Type in instructions and data using the format described in the Flex Tape Load program.

8. Discrepancy - Computer stops

   P=7515

   A=address of the discrepancy

   Z=0001

   To continue the program press the Run switch after returining it to the center postion.

9. End of File code typed. Computer stops

   P=7470

   A=0000

   Z=7707

10. To repeat the program press the Run switch after returning it to the center position.

h. Type Dump - Produces listings of programs and data directly on the on-line electric typewriter.

1. Turn on the typewriter

2. Put Operation Mode switch in the CLEAR position

3. Put Operation Mode switch in the COMPUTER position

4. Check to ensure Input Disconnect switch is in center position

5. Set P=7407

6. Set A=initial address of the region to be listed

7. Press Run switch

8. Computer stops

   P=7677

   A=0000

   Z=7701

9. Set A=address of the last word to be listed.

10. Press Run switch after returning it to center position. Type Dump will produce the specified listing allowing up to 48 listings on each page, with a double space preceding each listing with an address divisible by eight. Each listing shows the address where the information was stored (4 digits) followed by a tab, followed by the information (4 digits). Computer stops to allow the operator to change paper.

11. End of Page

    P=7743

    A=0000

    Z=7702

    Continue by pressing the Run switch after returning it to center position.

12. Program Stop - Typewriter shows listing is completed by an end of file code (carriage return followed by a period).

    P=7674

    A=0000

    Z=7707

13. To repeat insert the initial address in A and press the Run switch after returning it to center position.

i. Clear Memory - Clears locations 0000 through 7377. The region

occupied by RS 022 and 7777 are not cleared.

1. Set P=7410
2. Press Run switch
3. Program Stop (0000 through 7377 clear)

   P= 7417

   A=0000

   Z=7707
4. Computer will start the Flex Tape Load program automatically and read a Flexowriter program tape inserted in the reader if the Run switch is pressed after returning it to center position.

3. Space required - $377_8 = 255_{10}$ locations
4. Temporary Storage Requirements - Locations 0070 through 0077
10. Timing - Output approximately 60 frames per second

    Input approximately 350 frames per second
12. Caution to user - Locations 7400 through 7776 are used by RS 022. Location 7777 cannot be loaded, verified, summed, punched or listed by any of the service routines in RS 022 unless it is the first location of the selected region. Due to storage limitations in RS 022, when the illegal Flexowriter code 75 is received in the Flex Load, Flex Verify of Type Load programs, the computation stops with the computer in an OUTPUT status. If the Run-Step switch is returned to the center position, P and A will both contain 7545 and Z=7465. The Clear switch must be pressed to clear this condition.

14. Equipment

Configuration - Standard 160 Computer.

A. IDENTIFICATION

TITLE: Punched Paper Tape Duplicator

IDENTIFICATION: S 023 (Replaces S 013)

CATEGORY: Service Routines

PROGRAMMERS: J. Pederson

DATE: December 1960

B. PURPOSE

This program produces multiple copies of a given seven level punched paper tape.

C. USAGE

1. Operation Procedure

a. Enter Tape

1. Turn on reader, insert S 023 tape and load starting P = 0000
2. Place tape to be duplicated on the reader and enter with P=0000. Tape will be read in until succeeding blank frames indicate the end of the tape.
3. Program Stop

P=0214

A=0000

Z=7701

4. If more information remains to be duplicated press switch after returning it to center position.
5. Error Stop - More than 6395 frames have been entered.

P=0206

A=7777

b. Verify

1. Place tape on reader and run with P=0001
2. Program Stop

P=0034

Z=7707

3. Error Stop

P=0361

A=0000

Z=7705

c. Punch

1. Turn on Punch

2. Set P=0002 A=number of copies desired (in octal)

3. Press Run switch

4. Program Stop

P=0320

Z=7707

The program will punch an 18 inch leader between copies.

d. To Verify multiple copies

1. Place tape in the reader

2. Set P=0001; A=number of copies to be verified

3. Error Stop

P=0361

Z=7705

Tear out bad copy. Master Clear. Place tape on the reader and set P=0030. Press Run switch. Alternative method: Set P=0001, A=remainder of the tape to be verified and press Run switch. Program Stop with

P=0034

Z=7707

The program will copy up to 6395 frames of seven level tape and produce the number of copies specified by the A Register. It also produces an 18 inch leader between copies. Use S 025 for tapes greater than 6395 frames.

3. Storage Requirements: Program 0000 - 0377

Tape Image 00400 - 7775

13. Equipment Configuration - Minimum 160 Computer

A. IDENTIFICATION

TITLE: Tape Leader Preparation

IDENTIFICATION: S 024 (replaces S-001)

CATEGORY: Service Routines

PROGRAMMERS: R. Olson

DATE: December 1960

B. PURPOSE:

Used to prepare paper tape leaders for identifying program tapes. The numerical characters 0 through 9, the letters of the alphabet, as well as ⟸ ,), (, can be punched on a tape leader.

C. USAGE:

1. - Operational Procedures - Program S 024 can be loaded anywhere in memory except locations 0016 through 0100. The P register will contain the initial address + 221 when loading is completed.

1. Master Clear
2. Turn on Punch
3. Enter initial address of program in both the P and A registers.
4. Press Run switch - Two frames of blank leader will be punched.
5. Set A to code for the desired character to be punched from the character code table.
6. Press Run switch - Desired character will be punched RETURN to step (5) to continue punching additional characters.

In order to space words enter 0050 into the A register in step (5) above.

The programmer must allow sufficient blank leader between his program and the program heading. Most of the resident service routines used to dump the contents of memory or reproduce paper tape are designed to prepare sufficient blank leader for spacing.

CHARACTER CODES:

Character	Code	Character	Code
0	0	K	24
1	1	L	25
2	2	M	26
3	3	N	27
4	4	O	30
5	5	P	31
6	6	Q	32
7	7	R	33
8	10	S	34
9	11	T	35
A	12	U	36
B	13	V	37
C	14	W	40
D	15	X	41
E	16	Y	42
F	17	Z	43
G	20	⇐	44
H	21	-	45
I	22	(	46
J	23	)	47
		space	50

2. Arguments or Parameters - None

3. Space Required - $172_8 = 122_{10}$ locations

4. Temporary Storage Requirements - Locations 0070 through 0076

10. Timing - Output approximately 60 frames per second

12. Caution to user - Make certain that at least two inches of blank tape separate heading and program.

13. Equipment configurations - Minimum 160 computer

14. References - See S-001 Tape Leader Preparation

A. IDENTIFICATION

TITLE: Punched Paper Tape Duplicator (for tapes of more than 6395 frames)

IDENTIFICATION: S 025

CATEGORY: Service Routines

PROGRAMMER: R. M. Olson

DATE: February, 1961

B. PURPOSE

This routine is designed to reproduce paper tapes of more than 6395 frames by placing the paper tape image on magnetic tape.

C. USAGE

1. Operational Procedure

a. Load Program Tape S 025

1) Place S 025 tape in the reader and set P = 0050.

2) Set load switch and press run.

3) Program stop

P = 0406

A = 2161

Z = 0000

b. Enter Tape to be Duplicated

1) Place tape to be duplicated in the reader.

2) Load a reel of magnetic tape onto tape unit. Check the following items:

a) Magnetic tape unit 1 selected.

b) BINARY parity selected.

c) The tape is run forward past load-point, or is set to load point and the CLEAR button has been pushed.

d) The magnetic tape unit has a green ready light on.

The magnetic tape units need not be touched during the rest of the operations.

3) Set P = 0050

4) Press Run switch

5) Temporary program stop

P = 0167

A = 0000

Z = 7700

At this stop there are two options:

a) If more tape is to be duplicated; press the Run switch after returning it to the center position.

b) If at the end of the tape: Master Clear, set P = 0166, and press the Run switch. This will transfer the last partial block of data onto magnetic tape.

6) Final program stop.

P = 0204

A = 2161

A = 7700

7) Error Halt (parity error)

P = 0207

A = 2161

Z = 7750

c. Verify Master Tape and Copies

1) Place tape in reader.

2) Set P = 0052

3) Press the Run switch

4) Proper verify stop

P = 0320

A = 2161

Z = 7700

5) Improper verfiy stop

P = 0316

A = 2161

Z = 7700

6) Error Halt (parity error)

P = 0306

A = 2161

Z = 7752

d. Punch New Copies

1) Turn on punch

2) Set P = 0054

3) Press the Run switch

4) Program stop

P = 0400

A = 2161

Z = 7700

5) Error Halt (parity error)

P = 0376

A = 2161

Z = 7754

3. Space Required - $324_8 = 212_{10}$ locations.

4. Temporary Storage Requirements - locations 0070 through 0076.

12. Caution to user - Make certain that the magnetic tape unit is set to BINARY mode and the Tape Unit 1 is selected.

13. Equipment configurations - Standard 160 computer.

A. IDENTIFICATION

TITLE: Flexowriter Tape to ANelex

IDENTIFICATION: S 026 (Flexlex)

CATEGORY: Service Routines

PROGRAMMER: Bud Vitoff

DATE: February, 1961

B. PURPOSE

1. Flexlex was designed primarily for printing OSAP listing tapes. It can be used efficiently whenever the ANelex is available for use immediately after assembly.
2. The routine can be used for printing any paper tape prepared by or for a Flexowriter.

C. USAGE

1. Operational Procedure
   a. Load program into 0000 through 0650.
   b. Turn on reader. Insert tape to be printed anywhere on its blank leader.
   c. Master clear and start. A page eject is executed before printing starts.
   d. When ten consecutive blank frames are detected, a page eject is executed and printing stops. Starting at this point enters a closed "page eject and stop" loop.
   e. A master clear restarts the program.
   f. A closed "page eject and stop" loop can be entered by setting P = 0001.

7. Programmed Stops

   Z = 7700: Page eject loop entered at 0001

   Z = 7702: Page eject loop entered after ten consecutive blank frames.

Z = 0004: Program check sum failure. Reload program.

9. Output format

a. Lines per page: 56

b. Flexowriter code interpretation (7th level is ignored):

1. carriage return and tabulator codes cause appropriate printing action, with tab stops after every 12 columns (i.e. in columns 13, 25, 37, etc.)*

2. delete codes and illegal codes are ignored.

3. all other non-printing codes are represented by the indicated substitutes:

Code	Meaning	Substitute
02	color shift	'
43	stop	*
47	upper case	:
57	lower case	?
61	backspace	(

c. An automatic carriage return is executed after 120 columns have been "set" for a line of print.

10. Timing

Printing speed varies with the amount to be printed on each line, but OSAP listings run about 325 lines per minute.

11. Equipment configuration

Basic 160 computer and ANelex.

*
Tab stops must be equally spaced across the page; however, the routine may be easily changed to provide one of the following options:

1. tab interval fixed at a value other than 12, or
2. tab interval fixed, with provision for changing it at the beginning of each run (by manually changing the contents of register A at a programmed stop).

The tab interval is set at the beginning of a run by the instructions in locations 0107 and 0110:

Location	Current	Change 1	Change 2
0107	LDN 14		LDN xx
0110	LDN 14	LDNxx	HLT 01

Load the program, and make the change in core. Master clear and start. Error stop 0004 will display the calculated new check sum in register A.

Permanent change:

Enter the new check sum into location 0650 and punch a new bi-octal tape.

Temporary change:

Start. Program stop 7704 will display the programmed check sum in register A. (Of course, the check sum difference should be accounted for by your changes.) Start to enter the program.

A. IDENTIFICATION

TITLE: 160 ANelex Dump

IDENT NUMBER: S 029

CATEGORY: Composite Output

PROGRAMMER: T. A. Ammerman

DATE: April, 1961

B. PURPOSE

List 120 character BCD tapes on the ANelex printer, 1000 lines per minute. The tapes may be prepared either on or off line. The 160 Computer provides control.

C. USAGE

1. Operational Procedure:

   a. Load program tape

      1) Turn on reader and insert S 029 tape
      2) Master Clear
      3) Set Load-Clear switch to LOAD
      4) Run

   b. Load magnetic tape

      1) Place tape to be listed on tape unit.
      2) Select unit number 1
      3) Select CODED mode
      4) Press WRITE LOCKOUT button

   c. Select 160 - ANelex

      1) Set 160 - 1604 ANelex switch to the 160 position.

   d. Set desired parameters in A

A = 0000	Eject enough paper to clear printer
A = 0001	Single space
A = 0002	Double space
A = 0003	Spacing under program control
A > 0003	Illegal code (however will result in single spaced listing

e. Set P = 0100

f. Run

g. Stop
A = 0020

This denotes end of file. If more records are to be printed, reset parameters and Run. A stop with A ≠ 0020 indicates an error (see Error Stops).

3. Space required: $6217_8 = 3215_{10}$ locations

7. Error Stops:

a. Parity error - record will be read three times before the computer stops. To continue listing press Run, bad record will be printed and routine will continue

b. ANelex drops out of READY - depress Ready button on the ANelex and printing will continue.

9. Output: Print out of the BCD tape at 1000 lines per minute.

12. Restrictions:
In order to obtain the maximum speed of 1000 lpm from the ANelex it is necessary to restrict the number of characters to the first 47 on the print wheels. Those used are shown in table 1. Table 2 lists the characters which will appear as blanks.

Table 1. Usable Characters

0	A	K	U	(
1	B	L	V	)
2	C	M	W	/
3	D	N	X	*
4	E	O	Y	,
5	F	P	Z	:
6	G	Q	.	~
7	H	R	-	
8	I	S	+	
9	J	T	=	

Table 2. Non-recognized Characters

≠	'	≡	↑	≥
≤	[	∧	↓	?
%	]	∨	>	;
$	→	¬	<	

13. Machine Configuration:
ANelex printer, 160 computer, Amper FR 3/4 00 tape handler.

A. IDENTIFICATION

TITLE:	Single Precision Fractional Square Root
IDENTIFICATION:	S 042
CATEGORY:	Demonstration Routine
PROGRAMMER:	Sanford Elkin
DATE:	February, 1961

B. PURPOSE

This is a demonstration routine. It will find the square root of a proper fraction with a maximum error of 2^{-11}.

C. USAGE

1. Operational Procedure: Load the bioctal tape starting at cell 0, and clear. Place the number N in the A register, and start. $X = \sqrt{N}$ will appear in the A register. Place a new N in A and repeat.

3. Space Used: Octal cells 100-206, plus locations 0, 1, 7, and 55-67.

10. Timing: Each iteration about 2.25 milliseconds, maximum of 25 milliseconds.

11. Accuracy: Maximum error is 2^{-11}.

D. MATHEMATICAL METHOD

Newton-Raphson iteration with $X_{i+1} = 1/2\ (X_i + N/X_i)$ and $X_0 = 1$, stopping when $\Delta X \leq 2^{-11}$. The Single Precision Divide Subroutine is used.

A. IDENTIFICATION

TITLE:	Single Precision Divide Subroutine
IDENTIFICATION:	S 043
CATEGORY:	Library Routine
PROGRAMMER:	Sanford Elkin
DATE:	February, 1961

B. PURPOSE

This subroutine will divide a positive 23-bit fraction by a positive 11-bit fraction, giving a rounded 11-bit fractional quotient.

C. USAGE:

1. Operational Procedure: The dividend (a and b) must be placed in locations 60 and 61 respectively, and the divisor (x) in 62. The contents of a must be less than the contents of x, and both must be positive. b contains 12 low order magnitude bits, which may be all zeroes. The routine is entered at the symbolic address DVDSBR, and the return address must be in cell 7. The quotient (y) will be in cell 63.

3. Space Required: $27_{10} = 33_{8}$ locations, plus locations 7 and 60-63.

10. Timing: 2.0 milliseconds

11. Accuracy: 11 bits, with answer being rounded.

12. Cautions to User: The dividend is destroyed by the subroutine. If the true quotient equals or exceeds $1 - 2^{-12}$, the octal value 4000 will be in cell 63.

D. MATHEMATICAL METHOD:

Repeated subtractions are used.

A. IDENTIFICATION

TITLE:	9-Bit Quick Multiply
IDENTIFICATION	S 044
CATEGORY:	Library Routine
PROGRAMMER:	Sanford Elkin
DATE:	February, 1961

B. PURPOSE

This subroutine will multiply two signed 11-bit numbers together in about 600 microseconds, giving a signed answer accurate to approximately 10 bits.

C. USAGE

1. Operational Procedure: The two numbers and their product are interpreted as signed fractions with magnitude less than unity. The multiplicand must be in cell 10 and the multiplier in cell 11. Cells 70-72 are used for temporary storage, and the product will be placed in the A register. The routine is entered at the symbolic address MPY9B, and cell 2 must contain the return address.

3. Space Required: 181_{10} or 265_{8} locations, plus locations 10, 11, and 70-72.

10. Timing: Average - 595 microseconds, maximum - 660 microseconds.

11. Accuracy: Average - 1×2^{-11}, worst case - 3×2^{-11}.

D. MATHEMATICAL METHOD

2^{-10} is added to the absolute value of each factor, and the two low-order bits of each are truncated. If either resulting number is 0 or 1, 0 or the other factor becomes the answer. Each of the three octal digits of the resultant multiplier is examined, and the partial product of each with the resultant multiplicand is obtained. The bits in each partial product which are less significant than 2^{-12} are truncated, and the sum of the partial products is truncated to 2^{-11}. The sign of the product is then obtained, and the answer is placed in the A register.

A. IDENTIFICATION

TITLE: Integer Divide
IDENTIFICATION: S 045
CATEGORY: Library Routine
PROGRAMMER: Sanford Elkin
DATE: March, 1961

B. PURPOSE

This subroutine will divide a positive 23 bit integer by a positive 11 bit integer, giving a 12 bit quotient with an 11 bit remainder.

C. USAGE

1. Operational Procedure: The 23 bit integer must be placed in a (70) and b (71), the divisor in x (72), and the return address in exit (7). x must be greater than a (a being the more significant half). The routine must be entered at DVDINT. The quotient will be in y (73) and the remainder in a (70).

3. Space Required: $22_{10} = 26_{8}$ locations.

4. Temporary Storage: Octal locations 7 and 70-74.

10. Timing: 2.0 milliseconds.

D. MATHEMATICAL METHOD

Repeated subtractions are used.

A. IDENTIFICATION

TITLE:	Paper Tape Edit Program
IDENTIFICATION:	S 046
CATEGORY:	Service Routine
PROGRAMMER:	Harold C. Schnackel
DATE:	February 8, 1961

B. PURPOSE

This program permits changes to be made to symbolic paper tapes prepared for assembly via FLAP, OAR, or OSAP. Changes may be in the form of replacements, insertions, or deletions of complete lines relative to the tape to be corrected.

C. USAGE

1. Operational procedure
   a.) Machine load bioctal program tape at location 0000.
   b.) Position correction tape in reader, master clear, and run.
   c.) When correction tape has stopped, position the tape to be corrected in the reader, and run. The original tape will be read in and the corrected tape will be punched out.
   d.) Normal stops (octal)
      1.) Location 0611: correction tape has been read in correctly. Run from here to read tape to be corrected.
      2.) Location 1026: END psuedo-op on the original tape has been encountered and the tape has been edited correctly and completely.
      3.) Location 1116: WAI psuedo-op on the original tape has been encountered. Continuation is left to the discretion of the user.
2. Arguments, or Parameters: none
3. Space required
   a.) Program plus transient storage occupies locations 0000 through 1122.
   b.) Input from the correction tape occupies locations 1123 through 7776 as needed.
4. Temporary storage requirements: see C3
5. Alarms, or Print-outs: none
6. Error Returns: none
7. Error Stops (octal)
   a.) Location 0444: The additive field of the current correction code caused reference to be made to a line which should have been referenced by the next location symbol on the original tape. Continuation will cause the current correction code to be bypassed and the next to be processed.
   b.) Location 0524: Computer capacity for holding the corrections has been exceeded. Corrections must be reorganized into two or more passes.

c.) Location 0707: A non-digit character with the exception of blank, space, and delete code, is present in the additive field of the correction code line. A CR is the only legal termination of this field. Correction tape must be corrected.

d.) Location 0764: Computer storage was completely searched without finding a correction code. This is due to a machine error.

e.) Location 1011: An illegal edit code (outside the set d,i,r,z) has been encountered. User must correct the correction tape and restart.

f.) Location 1027: The tape to be corrected has been processed. However, all corrections were NOT processed. Somewhere on the correction tape a correction code has referenced a non-existent location symbol. Depressing RUN here will cause the first unprocessed correction code to be punched out and a stop to occur at 1112.

8. Input and Output Tape Mountings: none

9. Input and Output Formats

FLAP, OSAP, and OAR formats may be used on the tape to be corrected. The format of a correction specification line on the correction tape is as follows:

a.) Tab

b.) A letter of the set d,i,r,z where

d = delete
i = insert
r = replace
z = end of correction tape

c.) Tab

d.) Location symbol on original tape

e.) Tab

f.) Pure decimal digit address relative to the location symbol. The absence of a location symbol means that this numerical quantity is an absolute line count.

g.) Carriage Return

The following example illustrates formats of input and output tapes.

Input tape to be corrected:

```
abcd     ldf   qxr
         adn   22
         sbf   qxr      1
         sti   cntr
         ldf   02
         jfi   02
               next
               subr
next     aod   cntr
         hlt   00
         end
```

Corrections to be applied:

```
          r     abcd     1
          adn   33
          sti   efg
          d     abcd     2
          r     abcd     6
                return
          i     abcd     8
                4756
          r     next
return    aod   cntr
          nzb   abcd
          z
```

Corrected tape:

```
abcd      ldf   qxr
          adn   33
          sti   efg
          sti   cntr
          ldf   02
          jfi   02
                return
                subr
                4756
return    aod   cntr
          nzb   abcd
          hlt   00
          end
```

10. Timing: Limited by paper tape reader and paper tape punch.
11. Accuracy: not applicable.
12. Cautions to user:
    a.) Tape to be corrected must begin with a carriage return.
    b.) Tape containing the corrections must begin with a carriage return.
    c.) Any character in the additive field of the correction specification line that is not of the set (null, space, digit, delete code) will cause an error stop.
    d.) The additive field of the correction specification line may be terminated only by a carriage return.
    e.) The correction specification line identification, i.e., the symbol and additive fields, is always relative to the original program listing.
    f.) There must be a delete code for every line to be deleted.

g.) There may be any number (within the capacity of the Edit Program) of consecutive insertions or replacements following a single insertion or replacement code, respectively.

h.) A replacement of line n with m new lines is equivalent to deleting line n and inserting the m new lines at n + 1.

i.) An insertion results in a line or lines inserted AHEAD of the line specified in the correction code.

j.) The order of corrections (correction line identification) must be according to the original list.

k.) The symbolic identification of a line of the original listing must be made either by the location symbol of the line itself or by relative reference to the last symbol preceding the line.

l.) The same line in the original listing may not be referenced by more than one correction code.

13. Equipment configuration: minimum

14. References: none

A. IDENTIFICATION

TITLE:	Prime Factor Extractor
IDENTIFICATION:	S 047
CATEGORY:	Demonstration Routine
PROGRAMMER:	Sanford Elkin
DATE:	February, 1961

B. PURPOSE

This is a demonstration routine which will find the prime factors of any number up to 4095_{10} ($=7777_8$).

C. USAGE

1. Operational Procedure:
   a) Load the bioctal tape starting at location zero, and clear.
   b) Place the number in the A register
   c) Run. A prime factor will appear in A.
   d) Run again. The quotient will appear in A. If the quotient is one, the prime factors have been extracted and the routine may be begun again with b). Otherwise repeat c) and d).

3. Space Required: $112_{10} = 160_8$ locations (including the divide subroutine), plus locations 0, 1, 6, 7, and 31-64.

D. METHOD

The number is divided by primes from 2 to 61_{10} ($=75_8$), and the remainder tested for zero.

A. IDENTIFICATION

TITLE:	BI-OCTAL DUMP 2
IDENTIFICATION:	S 049
CATEGORY:	Resident Service Routine
PROGRAMMER:	H. C. Schnackel modified by C. M. Atchison
DATE:	April 4, 1961

B. PURPOSE

This program will sequencially dump in BI-OCTAL (machine-load format), the information stored in core memory beginning with the starting address, and ending at the terminating address minus one. The output tape can be loaded by use of the "Load" reader mode on the 160 computer.

C. USAGE

1. Operational Procedure
   a. Load BI-OCTAL Program Tape at 7732_8.
   b. Set P register at 7732_8.
   c. Set A register equal to first word location to be dumped.
   d. Run.
   e. Set A register to last word location to be dumped plus one.
   f. Punch on.
   g. Run.

3. Space Required
   a. Decimal--37 locations high core
   b. Octal----45 locations high core

4. Temporary Storage Requirements
   Uses and restores location 0.

12. Cautions to User
   a. The program is not relocatable.
   b. The program does not punch leader or trailer in the output tape.

13. Equipment Configuration
   a. Basic 4k 160 with paper tape input and output.

A. IDENTIFICATION

TITLE:	TRACK
IDENTIFICATION:	S 050
CATEGORY:	Service Routines
PROGRAMMER:	R. Beale
DATE:	March, 1961

B. PURPOSE

Trace a program, providing a flex-coded paper tape as output. Only the beginning and ending addresses of a consecutive instruction string appear as output, thus the object program is executed at higher speed than is possible using a full trace.

C. USAGE

1. Operational Procedure
   a. Clear memory
   b. Machine load the bioctal tape starting at 7000--correct loading will end with P = 7577.
   c. Machine load the object program without altering locations 7000-7600_8. Position input data tape in paper tape reader if required.
   d. Set P = 7000 A = starting address of program to be traced. Run. Halt 7701 will immediately occur--P = 7002. Without otherwise altering console, set A = normal contents of A at start of object program. Turn on punch, run.

3. Space required
   7000 through 7534
   $535_8 = 349_{10}$

10. Timing
    Depending on the nature of the object program, its instructions are executed at 10-500 per second. The average is close to 75 per second.

12. Cautions to User
    Because each instruction must be interpreted before execution, the timing relationships within a program are altered--it is not possible to trace most card to tape programs for this reason. The track program attempts to faithfully execute a sequence of instructions regardless of its correctness. If proper selection of peripheral equipment does not precede the activation of the equipment, the computer hangs on a "sel" error indication. Punching by the track program does not alter the object program external function selections or senses. Each instruction is executed from upper core, rather than from its normal position in memory; therefore programmed error stops or halts in the object program show P = 7146.

A. IDENTIFICATION

TITLE:	Binary to 4-bit Decimal Conversion
IDENTIFICATION:	S 051
CATEGORY:	Library Routine
PROGRAMMER:	Sanford Elkin
DATE:	March 1961

B. PURPOSE

This subroutine will convert a 24 bit binary integer into a decimal integer with each digit in successive cells.

C. USAGE

1. Operational Procedure: The 24 bit integer must be placed in locations BINDEC + 100_8 and BINDEC + 101_8, and the return address in cell 6. The 8-digit answer will be put in cells 70_8-77_8, with the units digit in 77_8.

3. Space Required: $67_{10} = 103_8$ locations.

4. Temporary Storage: Octal locations 6 and 70-77.

10. Timing = Approximately 4.3 milliseconds per decimal digit.

D. MATHEMATICAL METHOD

The binary integer is divided by 12_8. The remainder is stored in the appropriate location and the quotient used as a new binary integer.

A. IDENTIFICATION

TITLE:	Paper Tape Verify
IDENTIFICATION:	S 052
CATEGORY:	Service Routines
PROGRAMMER:	R. Beale
DATE:	January, 1961

B. PURPOSE

The program is used to verify copies of an original paper tape.

C. USAGE

1. Operation procedure:
   a. Master Clear
   b. Machine load Paper Tape Verify program at zero
   c. Master Clear
   d. Insert original tape into reader
   e. Run
   f. On HLT 01, P = 0107; insert second tape into reader and run without altering console
   g. On HLT 02, P = 0130: the last tape is equivalent to original--to verify another tape, place it in reader and run without altering console--upon successful verification step g may be repeated.

3. Space Required: $135_8 = 93_{10}$

7. Error Stops: On ERR 01, P = 0132: the last tape is not equivalent to the original--to continue verifying tapes, place new tape in reader and run without altering console. This will execute Step 1g.

10. Timing: 350 frames/second

13. Equipment Configuration: Minimum System

D. METHOD OR ALGORITHM

A series of sequence-sensitive check sums is formed for each tape and tested for equivalence. There is no limit on the lengths of tapes to be verified.

E. FLOW CHART

Not applicable.

A. IDENTIFICATION

TITLE:	ALNUP
IDENTIFICATION:	D 053
CATEGORY:	Resident Display Routine
PROGRAMMER:	C. M. Atchison
DATE:	April 7, 1961

B. PURPOSE

This routine will punch character messages in paper tape that are legible to an unskilled observer. These characters are formed in a 5 by 7 matrix on an output tape. The standard 160 Flexowriter coded paper tape is used as input.

C. USAGE

1. Operational Procedure:
   a. Prepare Flexowriter (Standard CDC 160 code) input tape.
   b. Load BI-OCTAL program tape at location zero.
   c. Turn punch on.
   d. Place Flex input tape in reader.
   e. Run from zero.
   f. Normal stop: P = 140, Z = HLT 77.

3. Space Required
   a. 574_8 locations.
   b. 380_{10} locations.

4. Temporary Storage Requirements
   a. Variable according to length of input file starting at location 600_8.

5. Cautions to User
   a. The program does not punch leader or trailer on the output paper tape.

13. Equipment Configuration
   a. Standard 160 coded Flexowriter for preparation of input tape.
   b. 160 computer with paper tape input and output.

A. IDENTIFICATION

TITLE: PARBIT
IDENTIFICATION: S 054
CATEGORY: Subroutine
PROGRAMMER: R. A. Zemlin
DATE: 3 May 1961

B. PURPOSE

For applications in which the 160 is to punch paper tape for subsequent re-reading, it is desirable practice to use the seventh level as a parity check on the remaining six levels, preferably using odd parity. PARBIT is a routine for calculating even or odd parity bits for 6-bit characters, and may conveniently be used to form 7-bit checked characters for output or to check 7-bit input characters for proper parity.

C. USAGE

1. Return address should be stored at location "exit". Transfer to locations "even" or "odd" for forming even or odd parity.

2. Enter with 6-bit character in A5-A0. Contents of the remaining positions of A are ignored. On return A5 contains the parity bit, and the remaining positions of A contain zeros.

3,4. Space required is 13 (decimal) or 15 (octal) cells, of which one is a temporary cell.

10. Execution time is 115.2 μs. (odd) or 108.8 μs. (even).

PARBIT

```
odd     lsn   40    entrance for odd parity
even    stf   x     entrance for even parity
        sha   02
        lsf   x
        stf   x
        lpn   12
        sha   12
        sha   02
        lsf   x
        lpn   40
        jfi   1
exit    bss   1     storage for return address
x       bss   1     temporary storage
```

A. IDENTIFICATION

TITLE: One Sixty Assembly Program
IDENTIFICATION: OSAP
CATEGORY: Assembly Program
PROGRAMMERS: R. Hyer
DATE: September 1960

B. PURPOSE

The OSAP Program accepts 160 Computer instructions expressed in symbolic form, assigns absolute locations to the symbolic addresses, performs the necessary work of relative addressing to provide an absolute machine code for input to the 160 Computer.

Data prepared on punched paper tape or punched cards will be accepted by the OSAP assembly program, or the input may be converted to magnetic tape and then this medium used for input.

The final output will be punched paper tape suitable for input to the 160 Computer using one of the program controlled loading routines.

OSAP is available in various versions for 160 Computers with different peripheral equipment. The version described here assumes a minimum 160 Computer using only paper tape input and output.

C. USAGE

1. Theory of Operation - OSAP operates as a two pass assembly program with an optional correction pass and an optional final conversion pass.

   During the first pass OSAP reads the symbolic tape and produces a symbol table which specifies the location of symbols found in the location portion of the program and also an undefined symbol

table.

The undefined symbol table lists all symbols occurring in the Address and Additive field which did not occur in the location field. At the end of the first pass OSAP will punch out the contents of the undefined symbol table. On the second pass OSAP reads in the symbolic tape and produces a listing tape which includes the information from the original symbolic tape and the final absolute assignment of the program. The listing tape then can be used to produce an optional binary tape for input to the 160 Computer. Optionally, the contents of the symbol table may be punched out. Two location counters are used. Absolute locations are assigned to symbolic instructions and quantities according to the current value of a location counter. The location counter is set to an initial value by an origin type pseudo instruction and is then advanced by one for each computer word created by the assembly program. Each time a symbol occurs in the location field of the coding form, the corresponding value of the location counter is assigned as the absolute location of that symbolic location.

The two counters used are the program location counter (PRG) and the constant location counter (CON). The program location counter normally set to 0100 at the start of an assembly run but may be set by a pseudo instruction to any value from 0000 to 7776. (In order not to interfere with the CON locations from 0000 to 0077 PRG should be set to 0100 and no lower). When in use, the program location counter is advanced by one for each line of coding assembled. In other words, PRG creates the contents of the P register for the machine language program being assembled. The constant location counter is used to assign constants and special information in the range of 0000 to 0077 and the constant location counter is limited to this range. An Error Stop will occur if the

CON exceeds this range.

The symbols TEM 0, TEM 1, ..., TEM 7 are permanently assigned to give corresponding location values of 70 to 77. Other symbols may be assigned to these locations in the course of a program.

2. Operation Codes

The mnemonic operation codes accepted by OSAP expressed as 3 characters - normally the first two characters define the instruction and the third character defines the address mode, unique instructions are given a three character operation code.

The codes are:

LP	Logical Product	(02, 10, 11, 12, 13)
LS	Logic Sum	(03, 14, 15, 16, 17)
LD	Load	(04, 20, 21, 22, 23)
LC	Load Complement	(05, 24, 25, 26, 27)
AD	Add	(06, 30, 31, 32, 33)
SB	Subtract	(07, 34, 35, 36, 37)
ST	Store	(40, 41, 42, 43)
SR	Shift Replace	(44, 45, 46, 47)
RA	Replace Add	(50, 51, 52, 53)
AO	Replace Add One	(54, 55, 56, 57)
ZJ	Zero Jump	(60, 64)
NZ	Non-Zero Jump	(61, 65)
PJ	Positive Jump	(62, 6)
NJ	Negative Jump	(63, 67)

The address mode control characters are:

N	No Address
D	Direct Address
I	Indirect Address
F	Forward Relative
B	Backward Relative
R	Relative

If the address mode is R, OSAP will select the correct direction.

The unique operations are:

SHA Shift A (01)
HLT Halt (77)
JPI Indirect Jump (70)
JFI Forward Indirect (71)
INP Input (72)
OUT Output (73)
OTN Output (74)
EXF External Function (75)
INA Input to A (76)

3. Control (PSEUDO) Instructions

Control instructions are included in OSAP to control the advancing and setting of the location counters, to provide convenient methods of controlling the operation of OSAP and to introduce information in a form different from the normal format. Control instructions are written in the same format as the standard 160 instructions and any special meaning is explained under each instruction.

a. ORG (Origin) The instruction ORG will cause the current location counter to be assigned the numeric value as specified by the sum of the Address and the Additive field. Normally the quantity in the Address and Additive field will be a number, however, it is legal to use a symbol in either or both fields, provided the symbol has been assigned a numeric value by the time the ORG instruction occurs during the first pass. If an undefined symbol is given in either field, the OSAP will stop and indicate the fact by the stop coding.

b. CON (Constant Location Counter) Normally the CON instruction is given with no information in the Address or Additive field, OSAP will use the constant location counter as the current location counter which will continue from its previous value.

If a value is given in Address or Additive field, the numeric value will be calculated as in origin and this numeric value will be used as the current value of the constant location counter.

c. PRG (Program Location Counter) This instruction causes the program location counter to be used as the current location counter in a manner similar to CON as given above.

Comments on ORG, CON and PRG

These three instructions do not cause the location counter to advance on assembling. Thus the address specified applies to the next instruction to be assembled. If a symbol is given in the location field, this symbol will be assigned the numeric value assigned to the next location.

d. BLR (Block Reserve) The BLR instruction will advance the current location counter by the amount specified in the Address plus Additive field; in addition if a symbol is given in the Location field, that symbol will be assigned to the first numeric address in the block. The Code BSS will also cause the same action.

e. WAI (Wait Input) WAI will cause OSAP to stop and allow for insertion of a new tape for input. On pressing the run switch, assembly will continue.

If on a Wait the operator desires to execute the END function, the program will be started at a different location after master clearing.

f. END (End of Data for Assembly) The occurrence of an end code will cause OSAP to prepare for the next pass. On repositioning paper tape, the operator operates the Run switch

to continue with the next pass.

g. EQU (Equivalence Statement) The EQU instruction assigns the numeric value of Address plus Additive to the symbol given in Location. Address and Additive may be symbolic provided they are defined prior to their occurrence on the first pass. Equivalence will not cause the location counter to step.

h. REM (Remarks Statement) All that follows the OP code on a REM statement will be taken as remarks and will be ignored by OSAP. A REM instruction will not cause the location counter to advance.

i. DEC (Decimal Number) This instruction will cause the digits given in Address field to be converted from decimal to binary and stored.

j. BCD, FLX, TTY These three instructions will cause the information given, starting in the comment field, to be converted to BCD (for printer listing), Flexowriter code (for typewriter listing) or Teletype code, and store two characters per word in successive locations. The maximum number of characters under this option is 56. The end of the characters to be coded in this form is indicated by occurrence of slash followed by a period (/.).

The coded information then can be used by a short subroutine to provide output from the 160. The subroutine will use the occurrence of the pair /. to indicate the end of data. If an odd number of characters are given, an additional space will be included to provide complete words.

k. BCR, FLR, TTR These three instructions will cause information to be converted as above and stored one character per

word in the right six bits of the word. Conversion will be stopped by the occurrence of the character pair / . and these two characters will not be included in the converted data. This form of information may be obtained as output with single output instruction, but require twice as much storage as information prepared by the BCD, FLX and TTY instructions.

4. Preparation of Input

a. Card Input

The card format is as follows:

Columns	2-7	10-12	15-20	23-28	31-80
	Location	OP	Address	Additive	Comment

Numbers and symbols may be punched left justified and OSAP will right justify the numbers. Also blank columns in a field will be ignored as for example the symbol A P will be treated as AP.

Column 1 is reserved for identification and the character in that column will be used to indicate differing card formats.

b. Punched Paper Tape

For punched paper tape input, the tab function indicates the end of a field and a carriage return (CR) indicates the end of the last field of a line. The format is then:

LOC TAB OP TAB ADDRESS TAB ADDITIVE COMMENT CR

A tab may be replaced by a carriage return at any position in the line. Spaces, code delete, and stop code are ignored in the assembly. A line of typing may be eliminated in the case of an error by typing a slash followed by a carriage return (/CR).

5. Rules of Operation

The following rules are followed by the assembly program:

a. A line containing no information in Location, OP, Address, and Additive fields will be ignored by the assembly, but will appear in the listing tape.

b. If no information appears in the OP field, the quantity formed by the Address plus the Additive will be stored as a full 12-bit number.

c. Information specified by the Additive field will be added to information specified in the Address Field modulus 2^{12} -1. If a minus sign (-) is the first character of the Additive field, the Additive is subtracted from Address.

d. All non-printing characters are ignored (except control characters).

e. All symbols are left justified and spaces are added to make a total of six characters.

f. All numbers are right justified and left most zeros are assumed in the conversion from octal or decimal to binary.

g. Symbols in the Address and Additive field are converted to numeric by table lookup using the symbol table. The two numbers are then combined according to rule c to produce a numeric value. The resulting 12-bit number is then reduced to an execution address.

h. If the numeric value is between 0000 and 0077, combine it with the operation code to form the instruction.

i. If the operation code indicates a relative address operation and the numeric value is greater than 0077, attempt to make

a relative address by subtracting the location of the instruction from the numeric value and obtain a number less than 0077 for forward relative. If this fails, subtract the numeric value from the location of the instruction to obtain a relative backward address. If this fails, assume the address zero and flag a possible error on the listing.

j. If no operation code is given, use the four digit numeric value as the word.

k. If the operation code does not indicate relative addressing, use the value 00 and flag a possible error in the listing.

6. Operating Instructions

a. Load OSAP Model Zero program starting at location zero using machine load. Turn on the punch and reader and place the tape to be assembled on the reader: start with P = 0000. The program will come to a stop with P = 1044; return the run switch to neutral and run again. The undefined symbols in your program (if any) will be punched and the program will stop with P - 1270, a = 0000, and Z - 7750. This last stop indicates the first pass of the assembly is completed.

b. Rewind input tape and place it on the reader. Return the Run switch to neutral (do not Step or Clear) and run again to perform the second pass. A stop with P - 2126 and Z = 7776 indicates the program is complete and the punched tape may be listed on a Flexowriter. If a sorted symbol list is desired, return the Run switch to RUN after the above stop and a symbol listing will be punched. Program stop at this time is P = 2243, A = 0000, Z = 0000.

CONTROL DATA CORPORATION

OSAP CODING FORM

PAGE NO. ____________

DATE ____________

PROGRAMMER ____________

LOCATION	OP	ADDRESS	ADDITIVE	COMMENTS

PROGRAM ____________ ROUTINE ____________ PAGE ____________

A. IDENTIFICATION **160.006**

TITLE: OSAP Correction Routine
IDENTIFICATION: OCR
CATEGORY: Assembly Routines
PROGRAMMERS: A. Perro
DATE: December 1960

B. PURPOSE

OCR is designed to correct an OSAP listing tape and to produce a new tape suitable for assembly under OSAP control. OCR is capable of changing an address, deleting an address (or addresses) three types of additions and one special corrective routine.

C. USAGE

1. Operational Procedure

   a. Load bi-octal tape containing OCR with P = 0000

   b. Place correction tape into reader and set P = 0002 (Flexo-writer Format)

   c. Press Run switch

   d. Program Stop

      P = 0754
      A = 0000
      Z = 7770

   e. Turn on punch. Insert assembly tape to be corrected in reader. Set P = 0000. Run.

   f. OCR will output corrected flex tape for new OSAP assembly.

   g. Load OSAP

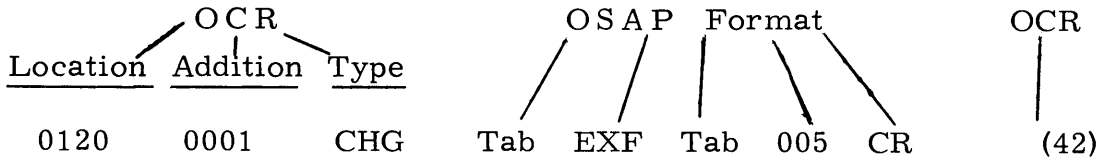
   h. Assemble new flex tape under OSAP control.

Correction Tape Preparation - The tape must be in Control Data Flexowriter coding in the following format:

Location	Addition	Type
4 Characters	4 Characters	3 Characters

These must be in sequence and immediately followed by the correction in OSAP input format and a period at the end.

EXAMPLE: Change location 0120 to 7505



All of this must be in sequence with no additional spaces. Each change or group must end with a period (Flexcode 42). The end of a correction tape must be a Flexowriter stop code (77).

Correction Entries:

Corrections to areas with existing locations:

ADD - Addition

Insertion of a new instruction or instructions between existing instructions. The new instruction will be inserted at the location specified and the existing instruction will be moved forward according to the number of insertions.

CHG - Change

Replace an instruction with another. This restricts the operator to one change per location on a line basis.

DEL - Delete

Delete an address or addresses beginning with the location specified.

New insertions where no locations are listed:

This format varies from the others being preceded by NLL.

Format: NLL Location Add. Type OSAP input format

EAL - Add just after a location

EBL - Instructions added before a group of addresses, but not at the beginning of a program.

EXAMPLE:		PRG	6000
		WAI	
6000	2114	LDI	P - - - etc.

Add instructions starting at 5770 to 6000 between WAI and 6000.

OCR format: Location contains the location nearest - 6000
Additions contain the number 0010
Type EBL - This will save the WAI, insert the octal 10 instructions and continue.

EAL - Address just after a location

EXAMPLE:	0025	1750	KILO	1750
			WAI	
			PRG	6000

Add instructions between 0025 and WAI

Use the same format as for EBL.

To correct a pseudo Op at the beginning of a program with no locations listed:

NPO - Followed by the OSAP input format and ending with a period.
This one entry will not have any location or type instruction.

```
EXAMPLE:                        CON         0010
                0010    0000            B
                0011    0000            C
                Change CON to CON 0050
                                CON         0050
                0050    0000            B
                0051    0000            C
```

USE OF THE FPP-33 SIMULATOR

WITH THE MODEL 160 COMPUTER

The FPP-33 is an easy to use programming language designed for solving scientific computing problems on the Model 160 Computer.

A limited repertoire of 1604 instructions is used. These are written in 1604 format, with an upper and lower instruction in each address. Each instruction in the Simulator consists of an operation code, an index designator, a break-point code, and a base execution address. All addresses are in octal notation. In section 7 there is a short table showing octal to decimal conversions.

The FPP-33 occupies a portion of the computer memory, leaving octal addresses 0020 to 1017 available to the programmer. This is the equivalent of 512 decimal 1604 addresses. (This represents 160 locations 0100 through 4074.) Each of these addresses may hold two instructions or one data word. Data words are entered as decimal floating point numbers.

A repertoire of 21 instructions is available in the FPP-33. These are standard 1604 numeric operation codes. Mnemonic codes are not available in the Simulator.

OPERATIONAL PROCEDURE

A. LOAD SIMULATOR TAPE

1. Clear Memory - Clears all of memory without exceptions

   1) Master Clear

   2) Set Enter-Sweep switch to Sweep

   3) Hold Clear button on Z Register down

   4) Press Run switch while holding clear button on Z Register

   5) Return Run switch to neutral position

   6) Master Clear

2. Load FPP-33 Tape - Uses all of memory except 0020 through 1017 (1604 Locations)

   1) Turn on Paper Tape Reader

   2) Insert tape on the blank leader

   3) Master Clear - This must follow any readjustment of the reader

   4) Set P=4100

   5) Set Load switch

   6) Press Run switch

   7) Computer stops with P=0060
      A=0024
      Z=0000

   8) If tape fails to read in correctly, repeat starting with step 2).

3a. Check Sum - Tests memory to determine if the FPP-33 or any other routine has been read in correctly.

The Check Sum routine, including a floating bi-octal dump, occupies 46 octal locations and may be loaded anywhere.

1) Turn on the reader and position tape
2) Master Clear
3) Set P= first 160 location to be used. Start at any location between 0100 and 4030
4) Set Load Clear switch to LOAD
5) Push Run switch

3b. Verify - Verifies the loading of a program tape through the check sum

1) Master Clear
2) Set P to the first 160 location into which the check sum routine was loaded.
3) Set A to the first 160 address of the area to be summed
4) Press Run switch
5) Computer will stop with Z=7700
6) Set A to the last 160 address of the area to be summed
7) Press Run Switch
8) Program Stop A= check sum

Z= 7730

To check the loading of the FPP-33 simulator tape set A=4100, then to 0060. The check sum 0035 will be displayed in A if the tape has been read correctly into memory.

B. LOAD A PROGRAM OF INSTRUCTIONS

1a. Loading Instructions from Flexowriter Tape

1) Place tape in reader

2) Turn on reader

3) Master Clear

4) Set P=7400

5) Set A=0000 (unless offsetloading is desired)

6) Press Run switch

7) Program Stop P=7462

A=0000

Z=7777

1b. Find and Correct Input Instruction - In loading Flexowriter tape if the last instruction is not followed by a carriage return and a semicolon, the instruction tape will not stop. If it goes completely out of the reader, the input instruction of the reader subroutine is destroyed.

If is then necessary to reload the 1604 Simulator or to find and correct the input instruction.

1) Reinsert the Flexowriter tape in the reader

2) Master Clear

3) Set P=7400

4) Press Run switch

5) Computer stops with P=address of the input instruction

6) Master Clear

7) Set P= address of the input instruction

8) Set Enter-Sweep Switch to Enter position

9) Set Z=7667

10) Press Step switch once

11) Return Enter switch to neutral

12) Master Clear

2. Entering Instructions Directly from the Typewriter

1) Master Clear

2) Set P=7402

3) Press Run switch

4) Type in locations and instructions in one of the acceptable formats for paper tape

5) Return Run switch to neutral position after entering the entire program

6) Master Clear

NOTE: In case of error: Carriage return and retype the location and instructions.

C. DATA LOAD

1. Loading Data (Including constants for use in a program) - Data must be entered in floating point format. The numbers are expressed as decimal fractions followed by an unbiased exponent.

   Example: 1604.0 is .1604

   1) Insert tape in reader

   2) Turn on reader

   3) Master Clear

   4) Set P=7404

   5) Press Run switch

   6) Program error-Address is out of range or exponent can not be converted, master clear, set P=7404 and press Run switch

   7) Program Stop - P=7761

      A=0000

      Z=7722

   NOTE: If the tape has run out of the reader because the last data word was not followed by a carriage return and a semi-colon proceed as in B1b above, however, set P-7404 in step 3.

2. Loading Data directly from the Typewriter

   1) Master Clear

   2) Set P=7406

3) Press Run switch

4) Type in locations and data in the format used for Flexowriter tape

5) Return Run switch to center position after all data have been entered

6) Master Clear

NOTE: If an error is made in typing a location, set Run switch to center, master clear, reset P to 7406, Run and start typing the location again.

If an error is made in a data word, type "x" and begin the data word again, provided the x is used before the exponent reading has been completed with tab or C.R. The x causes the routine to ignore the previous information in the data word. It has no effect on the address. The X may be followed by a carriage return and/or a tab. The input routine looks for a minus sign, a decimal point, or a final semi-colon, ignoring everything else.

E. PROGRAM START

1. Start operation of the program -

1) Load reader if program calls for tape input

2) Start punch if program calls for tape output

3) Master Clear

4) Set A=first 1604 instruction address of the program

5) Press Run switch

F. MANUAL DATA OUTPUT

1. Simulator A Register Dump - dumps a floating decimal number. via the typewriter

   1) Master Clear

   2) Set P=7410

   3) Press Run switch

   4) Program Stop

P=7766

A=0013

Z=7713

2. Instruction Dump - The contents of consecutive locations may be punched on paper or listed on the typewriter in octal notation.

   1) Master Clear

   2) Set P=7401 (produces octal tape)

      Set P=7403 (typewriter listing)

   3) Set A=first 1604 location to be dumped

   4) Press Run switch

   5) Computer stops

   6) Set A=last 1604 location to be dumped (for one instruction A= first location step 3)

   7) Press Run switch

Data may be dumped using the instruction dump routine to punch a tape. This produces a data tape in octal format. Such a tape may be read back into the 160 in the same way that instruction tapes are loaded, using the instruction load routine (P= 7400). If the A register contains zeros, the tape reads into the same data locations from which it was punched. The data may be offset loaded into different locations. by setting the desired increment in the A register.

3. Data Dump - The contents of consecutive locations may be punched on paper or listed on the typewriter in decimal notation.

   1) Master Clear

   2) Set P=7405 (data dump via tape output not presently available)
      Set P=7407 (typewriter listing)

   3) Set A=first FPP-33 data location

   4) Press Run switch

   5) Computer stops; Set A=last FPP-33 data location
      NOTE: To dump a single location, set A=the same address as step 3).

   6) Press Run switch

G. MANUAL INPUT-OUTPUT OPERATIONS

1. Instruction Load and Dump

Paper Tape: load, P=7400, run

dump, P=7401, A=first address, run

A-last address, run

Typewriter: load, P=7402, run

dump, P=7403, A=first address, run

2. Data Load and Dump

Paper Tape: load, P=7404, run

dump, P=7405, (not presently available)

Typewriter: load, P-7406, run

dump, P=7404, A=first address, run

For successive addresses move

switch to center and back to RUN.

3. A Register Dump P-7410, run

H. PROGRAMMED INPUT-OUTPUT SUBROUTINES (under program control)

These subroutines permit input and output under program control. In each case an exit is made to the next upper instruction of the program following execution of the subroutine. They are all entered by a selective return jump command (75 4 m). The various subroutine entry instructions are tabulated below:

1. Input Subroutine Entry

Paper Tape	75	40	1224
Typewriter	75	40	1226

2. Output Subroutine Entry

Paper Tape

data word followed by tab	75	40	1230
data word followed by C.R.	75	40	1232

Typewriter

data word followed by tab	75	40	1234
data word followed by C.R.	75	40	1236

I. STOPS

1. Normal Stops

Stops in machine operation occur at the end of manual load and dump operations, at breakpoint stops and error stops. For each of these stops certain specific digits will be displayed in the P, A and Z registers of the Control panel. In order to determine whether, for example, information has been correctly entered, the display panel should be checked to see whether a normal stop has occurred, or if an error produced the stop. The numbers displayed by the P, A and Z registers of the console following various stops are listed below:

Normal Stop	P	A	Z	
1604 Simulator Load	0060	0024	0000	*
Instruction Load	7462	0000	7777	
Data Load	7761	0000	7722	
Breakpoint Stop	6515	(address)	7700	
Selective Stop	7650	0 or 4	7707	
Instruction Dump	4111	0000	7777	
Data Dump	7766	(address + 1)	7707	
A Register Dump		0013	7713	

*This is the only stop for which the Z register is not illuminated with a green background display.

2. Breakpoint Stops

The breakpoint operation is useful for the initial check-out of a program; as soon as a program works, it is usual to set all the breakpoint designators to zero. This must be done before a FPP-33 program can be run on the 1604. Such a program will run about a thousand times faster on the 1604 than on the 160 using the FPP-33.

Using breakpoint designators, an octal breakpoint code number, 0001, 0002, or 0004 is manually entered into the 160 location 0005 as follows:

1) Master Clear

2) Set P register to 0005

3) Set Z to desired octal digit

4) Set Enter-Sweep switch to Enter

5) Set Run-Step switch to Step

6) Return the Enter-Sweep switch to center position

7) Master Clear

The FPP-33 takes the logical product of the digit in the breakpoint designator of an instruction and the number in 160 location 0005. If the result is zero, the program continues, if non-zero, the program stops with the location of the instruction displayed in the A register. If the breakpoint occurs on a lower instruction, 4000 plus the address will be displayed.

The following are some combinations that may be used to cause breakpoint stops:

Contents of location 0005	Breakpoint designator
0001	1, 3, 5, 7
0002	2, 3, 6, 7
0004	4, 5, 6, 7

To restart the program after a breakpoint stops:

1) Master Clear

2) Set desired address in A

3) Press Run switch

3. Program Error Stops

If an error has been made in writing the program or a data error occurs the computer will stop and the following will be displayed on the console.

P = 7766 A = 0013 Z - 7713

Typewriter Output

Move the run switch to center and back to Run and the error information will be typed out:

Paper Tape Output

1) Master Clear

2) Turn punch on

3) Set P = 7762

4) Run

When paper tape is used, the error code is preceded by a stop code to allow the operation in the Flexowriter prior to listing.

In both cases the format is:

1) an error code

2) contents of the designated index register of the instruction which caused the error stop

3) upper or lower instruction and its location

4) both instructions of the program step

An example of error output is:

```
c
b   0005
u   0042          6010    0020
                  7540    1157
```

This shows an illegal operation code, c, was used in the upper instruction at location 0042. The content of index register number 1, used in this instruction, is 5. Both the upper and lower instructions at location 0042 are outputted, the u indicates the illegal operation code 60 in the upper instruction.

The following error codes are used to show the cause of error stops:

c - illegal operation code

d - index designator fault

e - exponent fault, the exponent exceeds the permitted range of ± three decimal digits

i - fault caused by contents of index register

a) index register negative for search instruction

b) contents of index register greater than y for an index skip instruction.

r - address range fault, an address outside the permitted octal range 0020 to 1017

s - skip instruction placed in lower position in equality or threshold search, and in storage or index skip instructions.

a - an illegal argument has been presented to a subroutine

FPP-33 INSTRUCTIONS

A. INSTRUCTION REPERTOIRE

(1) Load A (12, b, m)+

(2) Load A, complement (13, b, m)+

(3) Store A (20, b, m)+

(4) A Jump (22, j, m)

(5) Floating Add (30, b, m)+

(6) Floating Subtract (31, b, m)+

(7) Floating Multiply (32, b, m)+

(8) Floating Divide (33, b, m)+

(9) Storage Skip (36, b, m)+ (upper instruction only)

(10) Enter Index (50, b, y)+

(11) Exit to 160 (51, o, m)

(12) Increase Index (51, b, y)++

(13) Load Index Upper (52, b, m_u)+

(14) Load Index Lower (53, b, m_l)+

(15) Index Skip (54, b, y)+ (upper only)

(16) Store Index Upper (56, b, m_u)

(17) Store Index Lower (57, b, m_l)

(18) Equality Search (64, b, m) (upper only)

(19) Threshold Search (65, b, m) (upper only)

(20) Selective Jump (75, j, m)+++

(21) Selective Stop (76, j, m)+++

\+ Indirect addressing is not provided, b+7 will produce an error stop.

++(51, 0, y) is the Exit to 160 instruction.

+++j= 0 or 4 only j + any other number produces an error stop.

B. INSTRUCTION EXPLANATION

A program using the FPP-33 has an upper and lower instruction in each location. Each instruction is made up of eight octal digits. The first two octal digits are the operation code, the next one is the index designator, b, this is followed by a one digit break-point code and a four digit execution address. The index designator, b, is used to specify which one of the six available index registers is to be used in performing the instruction. If no index register is to be used, the index designator is zero. The break-point code permits interruption of the program at any desired instruction. If no interruption is desired, the break-point code is zero.

In the enter index and increase index instructions, the last four octal digits specify an octal number, y, rather than an octal execution address, m. The A jump, selective jump, and selective stop instructions are not indexable; a jump designator, j, replaces the usual index designator b as the third octal digit of each jump instruction.

The index registers, numbered 1 through 6, are used as address modifiers or an skip or jump conditioners. When used as an address modifier, the content of the designated index register is added to the base execution address before the specific instruction

is interpreted by the computer. This permits ready programming of a loop to perform the same series of operations on a number of data words arranged in sequence. Index registers may be used to modify the base execution address of such instructions as load A; load A complement; store A; floating add, subtract, multiply, and divide; and storage skip. The result must be a valid address within the range 0020 to 1017 or a range fault stop will occur.

The second function of index designators, to control skip and jump operations, will be explained in the detailed description of these operations.

A program step is shown below:

0076	12	00	0165	(upper instruction)
	30	50	0273	(lower instruction)

The program step is in location 0076. The upper instruction is Load A(12) with the contents of address 0165 (no index register is used). The lower instruction is floating add (30) the contents of address 0273 + B where B is the contents of index register 5. The next instruction of the program will be the upper instruction in location 0077. Locations are in octal notation, so location 0077 will be followed by location 0100.

Detailed operation of the various available instructions is explained below:

1. Load A (12 b m)

This instruction clears the A register and replaces its contents with an operand whose location is specified by the sum of the base execution address, m, and the contents of the designated index register.

2. Load A, Complement (13 bm)

This instruction clears the A register and replaces its contents with the negative of an operand whose location is specified by the sum of the base execution address, m, and the contents of the designated index register.

3. Store A (20 bm)

This instruction stores the contents of the A register at the storage location specified by the sum of the base execution address, m, and the contents of the designated index register. The contents of the A register are not modified by this instruction.

4. A Jump (22 j m)

This instruction has eight sub-instructions which cause a change in the program sequence because of a specified condition of the A register. The index registers are not used for address modification in this instruction. The jump designator, j, in the instruction specified which sub-instruction is to be performed. In jump conditions 22 0, 22 1, 22 4, and 22 5, both negative and positive zero are treated as zero. In jump conditions 22 2, 22 3, 22 6, and 22 7, plus zero is treated as a positive number and minus zero is treated as a negative number.

Assuming the jump condition is satisfied, any jump instruction interrupts the normal program sequence and transfers program control to the location specified in the execution address, m. A return jump includes a provision for later return to the next upper instruction of the main program sequence. This is explained later in the instructions for writing subroutines.

The sub-instructions and the conditions required to cause a jump in the program sequence are as follows:

22 0 m - Jump if the A register content is zero

22 1 m - Jump if the A register content is not zero

22 2 m - Jump if the A register content is positive

22 3 m - Jump if the A register content is negative

22 4 m - Return jump if the A register content is zero

22 5 m - Return jump if the A register content is not zero

22 6 m - Return jump if the A register content is positive

22 7 m - Return jump if the A register content is negative

5. Floating Add (30 b m)

This instruction forms the algebraic sum of two floating-point quantities. An operand is read from the storage location specified by the sum of the base execution address and the contents of the specified index register. This operand is added to the previous contents of the A register. The result is normalized and rounded and left in the A register at the end of the sequence.

6. Floating Subtract (31 b m)

An operand in floating-point format is subtracted from the previous contents of the A register, also in floating-point format. The operand is read from the storage location specified by the sum of the base execution address and the contents of the specified index register. The result is normalized and rounded in the A register.

7. Floating Multiply (32 b m)

This instruction forms the product of a floating-point operand with the previous contents of the A register, also in floating-point format. The operand is read from the storage location specified by the sum of the base execution address and the contents of the specified index register. The result is rounded and normalized in the A register.

8. Floating Divide (33 b m)

This instruction forms the quotient of two quantities in floating-point format. The dividend must be loaded into the A register prior to the execution of this instruction. The divisor is read from the storage location specified by the sum of the base execution address and the contents of the specified index register. The quotient is rounded and normalized in the A register at the end of the operation.

9. Storage Skip (36 b m)

This instruction should always be an upper instruction in a program step. It causes the computer to sense the sign bit of the quantity in the storage location designated by the sum of the base execution address and the contents of the specified

index register. If the quantity is negative, an exit is performed. If the quantity is positive, a half exit is performed. None of the quantities in the operational registers are modified by this instruction. A half exit proceeds to the lower instruction of the program step, while an exit proceeds to the upper instruction of the next program step. Therefore a skip instruction should always be an upper instruction. The Simulator gives an error stop if this restriction is violated.

10. Enter Index (50 b y)

This instruction replaces the contents of the designated index register with the octal number y contained in the instruction itself. No storage reference is made in this instruction. If zero is used as the index designator, this instruction becomes the pass instruction.

There are six index registers available, numbered one through six. The operand y may be any number from 0000 through 7777 in octal notation. Note that the largest positive number that can be used in the Simulator system is 3777 (octal). Numbers 4000 to 7777 are treated as negative numbers in the 160. The following brief table may be helpful in using this instruction and the increase index instruction:

Octal	Decimal
0000	0000
0001	0001
.	.
.	.
.	.
3776	2046
3777	2047
4000	-2047
4001	-2046
.	.
.	.
.	.
7776	-0001
7777	-0000

To enter minus two into Index 3, the command would be; 50 30 7775. The result of indexing must yield a valid address within the address range 0020 to 1017. If the address lies outside this range, a range fault stop occurs for the instruction which uses the index register to generate the address-- and not on the 50 or 51 instruction which sets the number into the index register.

11. Exit from Simulator to 160 (51 0 m)

This command is used to enter a subroutine written in basic 160 language. Its use is explained in more detail in the section on subroutines.

12. Increase Index (51 b y)

This instruction adds the operand y to the contents of the designated index register. No storage reference is made in this instruction. See number 10 above.

13. Load Index (upper) (52 b m_u)

This instruction replaces the contents of the designated index register with the address from the upper instruction at the designated storage location.

14. Load Index (lower) (53 b m_l)

This instruction replaces the contents of the designated index register with the address from the lower instruction at the designated storage location.

15. Index Skip (54 b y)

This instruction compares the quantity in the designated index register with the operand, y. If the quantity, B, in the index register is less than y, the B is increased one count and half exit is performed. When the two quantities y and B become equal, the designated index register is cleared to zero and a full exit is performed. A half exit proceeds to the lower instruction of the program step, and a full exit proceeds to the upper instruction of the next program step. If the quantity in the index register is greater than the operand, y, there is an error stop. That is, the Simulator assumes the operand, y, is the maximum value the index register will be allowed to attain.

16. Store Index (upper) (56 b m_u)

This instruction stores the contents of the designated index register in the address portion of the upper instruction contained in the storage location specified by the base execution address. The remaining bits at the specified storage location are not modified in this operation. This instruction effectively inserts an address in the upper or first instruction of the specified storage location.

17. Store Index (lower) (57 b m_1)

This instruction stores the contents of the designated index register in the address portion of the lower instruction contained in the storage location specified by the base execution address. The remaining bits at the specified storage location are not modified in this operation. This instruction effectively inserts an address in the lower or second instruction at the specified storage location.

18. Equality Search (64 b m)

A list of operands is searched to find one that is equal to the content of the A register. The number of items in the list is specified by the content of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list, namely the one at address m + B - 1, where B is the contents of the designated index register. The content of the designated index register is reduced by one for each operand examined. The search continues until an operand is reached that is equal to the contents of the A register or until the contents of the designated index register are reduced to zero. If the search is terminated by finding an operand equal

to the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list is equal to the value in A, then a half exit is performed. In the equality comparison made here, plus zero and the minus zero are treated as equal.

19. Threshold Search (65 b m)

This instruction searches a list of operands to find the first one that is greater than the contents of the A register. The number of items in the list is specified by the contents of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list. The content of the designated index register is reduced by one for each operand examined. The search continues until an operand is reached that is greater than the contents of the A register or until the contents of the designated index register are reduced to zero. If the search is terminated by finding an operand greater than the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list is greater than the value in A, then a

half exit is performed. In the comparison made here plus zero is considered as greater than minus zero.

20. Selective Jump (75 j m)

This instruction has two sub-instructions in the Simulator system which cause a jump in program sequence. The index registers are not used for address modification in this instruction. The index designator in the instruction specified which of the two jumps is to be made.

75 0 m - Jump unconditionally

75 4 m - Return jump unconditionally

The use of any number other than 0 or 4 as the index designator will cause an error stop.

21. Selective Stop (76 j m)

This instruction has two sub-instructions which cause the program to stop. The index registers are not used for address modification in this instruction. The index code in the instruction specifies which of the two stops is to be made. A normal or a return jump to the base execution address occurs on restart.

76 0 m - Stop unconditionally (normal jump on restart)

76 4 m - Stop unconditionally (return jump on restart)

The use of any number other than 0 or 4 as the index designator will cause an error stop.

Note that in using a storage skip (36 b m) or index skip (54 b y) command, the instruction is followed by either an exit or half exit. An exit proceeds to the next program step, a half exit proceeds to the lower instruction of a program step. Therefore, these two commands should always be used as upper instructions. Similarly, the equality search (64 b m) and threshold search (65 b m) commands should be used only as upper instructions. The Simulator gives an error stop if this requirement is not satisfied.

The enter index (50 b y) instruction becomes a pass or "do nothing" instruction if 0 is used as the index designator. The instruction 5000 0000 performs a pass to the next instruction. It may be used, for example, to fill a lower instruction in order to place a skip or search command in an upper instruction.

The various commands to enter index, load index, increase index, and store index may be used to modify instructions during the operation of a program. This is useful in writing a program that requires the repeated execution of a small loop within a larger loop.

C. BREAKPOINT

Breakpoint provides a means of stopping the operation of a program at a desired point. The digit immediately following the index designator of an instruction is used for this purpose. If no breakpoint stop is desired, this digit is zero. The Simulator takes the logical product (the bit by bit product) of the digit in the breakpoint field of the instruction and the number in basic 160 location 0005. If the result is zero, the program continues, if non-zero, the program stops with the address of the instruction displayed in the A panel. If the instruction to be executed is a lower instruction, 4000 plus the address will be displayed.

For example: if a one, a three, or a seven is entered in 160 location, 0005 and the following program is being operated:

		Breakpoint designator	
0037	12	00	0166
	30	00	0167
0040	20	00	0172
	32	01	0172

The program will stop on the lower instruction in location 0040 and the octal number 4040 will be displayed in the A panel. After a breakpoint stop, the operator may examine a portion of

memory, or insert anything into memory. To restart, master clear, insert the desired address in A, and hit the Run switch to continue the program.

D. ARITHMETIC SUBROUTINES

Subroutines are relocatable. They may be loaded in any portion of the available memory, using the offset load routine to be described later.

A subroutine is entered using a selective return jump instruction (75 4 address). On completion of the subroutine, the next upper instruction of the program is performed. The selective return jump may be either a lower or an upper instruction. If it is an upper instruction, the lower half is not used and is filled in with zeros. On completion of the subroutine, the desired function is in the A register.

E. PROGRAM EXAMPLE

A Simple program example is given below:

To compute $(x_i - x)$ for 100 values of x_i.

Store x_i in locations 0100-0243 (octal).

Store constant x in 0247.

0476	50	10	0000	set index register 1 to zero
	50	00	0000	pass to next instruction
0477	12	10	0100	load A with x_i using index 1
	31	00	0247	floating subtract x
0500	30	10	0100	store x_i - x in 0100 plus i, using index 1
	50	00	0000	pass to next instruction
0501	54	10	0143	index skip
0502	(next instruction in the program.)			

This short program loop computes (x_i - **x**) for 100 values of xi, and stores (x_i - x) in the locations where the original x_i were stored. The loop is part of a program to compute a serial correlation with variable lag. Note the use of pass instructions, the selective jump goes to an upper instruction and the index skip must be used as an upper instruction. To avoid interrupting the sequence of commands, a pass instruction is used. This part of the serial correlation program occupies octal locations 0476-0501 of the 1604 Simulator.

F. DATA WORD FORMAT

Data and constants are entered as a decimal fraction followed by an unbiased exponent. Each is preceded by the appropriate sign. If no sign is given, the quantity is assumed to be positive,

For example,

		coefficient	exponent
1046.	Would be entered	-.1046	4
-10.46		-.1046	2

The format of an input or output word is:

± . coefficient (tab) ± exponent

followed by carriage return or tab.

The plus sign is not required. You may use a space or nothing preceding the decimal point. The coefficient may be any number of decimal digits up to and including nine digits, with the decimal point to the left of the most significant digit. The exponent carries its own sign and may be any number of decimal digits up to and including three digits.

ELABORATIONS ON OPERATIONAL PROCEDURE

A. TO LOAD INSTRUCTIONS (a program of instructions)

Instructions may be entered directly from the typewriter, or from paper tape prepared on the Flexowriter, or may be re-entered from paper tape prepared by the 160.

1. Flexowriter instruction tapes may be parpared in several formats. An acceptable format is:
    1) C.R. (carriage return)
    2) 1604 location (4 octal digits)
    3) Tab
    4) 1604 upper instruction (8 octal digits)
    5) C.R. and tab
    6) 1604 lower instruction (8 octal digits)

Step 2) through 7) are repeated as often as required to complete the program. The lower instruction area of the last instruction must be filled in. The last 1604 instruction is followed by a carriage return and a semicolon to indicate the end of the tape operation.

Other instruction formats are acceptable. The one most often used specifies only the first 1604 location (step 2), above) and repeats steps 4) - 7) as often as required to complete the program.

Alphabetic comments may follow the instructions; letters and spaces are ignored by the Simulator's instruction reader subroutine.

The following instruction format corresponds to Steps 2) - 7) preceded by a carriage return:

Location	Instruction	
0035	12 00 0166	Upper
	31 00 0170	Lower
0036	20 00 0170	Upper
	76 00 0035	Lower

Or step 5) may be a tab, step 7) a carriage return and tab, and steps 1 - 3 may be performed only once:

Location	Upper Instruction		Lower Instruction	
0035	1200	0166	3100	0170
	2000	0171	7600	0035

2. To load instructions from Flexowriter tape:

   1) Place tape in reader, wider side toward the console, and depress the reader arm.

   2) Turn reader on

   3) Master clear

4) Set P to 7400

5) Set A to zero unless offset loading is desired

6) Run

Offset loading is discussed in the section on subroutines. On a normal stop the following will be displayed on the console:

P 7642 A 0000 F 7777

If the last instruction is not followed by a carriage return and a semicolon, the instruction tape will not stop. If it goes completely out of the reader the input instruction of the reader subroutine is destroyed. It is then necessary eithe to reload The Simulator or to find and correct the input instruction:

1) Reinsert the tape in the reader

2) Master Clear

3) Set P to 7400 and Run

4) A stop will occur with P at the address of the input instruction.

5) Master Clear*

6) Set P to the address shown in step 4*

7) Set Enter-Sweep to Enter

8) Set Z to 7667

9) Step once with Run-Step switch

10) Return Enter-Sweep switch to center position

11) Master Clear

* Master Clear and reset P to have the 160 operating in the correct phase for re-entry.

3. To enter instructions directly from the typewriter:

   a. Master Clear

   b. Set P to 7402

   c. Set Run-Step switch to Run

   d. Type in locations and instructions in one of the acceptable formats for paper tape.

   e. When the entire program has been entered, set Run switch to center position and

   f. Master Clear

If an error is made in typing an instruction, carriage return and retype the location and instruction. Typing a wrong location will probably also require restoring the proper contents of that location.

B. TO LOAD DATA (including constants for use in a program)

Data must be entered in floating point form. The numbers are expressed as decimal fractions followed by an unbiased exponent.

For example: 1604.0 would be entered as .1604 4

1. Flexowriter data tapes are prepared in the following format:

1) C.R. (carriage return)

2) 1604 location (4 octal digits)

3) Tab

4) Sign of the coefficient (plus, space, minus or nothing).

5) Decimal point

6) The coefficient, any number of decimal digits up to nine digits. Digits after the first nine will be ignored. Missing digits will be assumed to be trailing zeros.

7) Tab

8) Sign of the exponent (plus, space, minus or nothing).

9) The exponent, any number of decimal digits up to three digits. Digits after the third one will be ignored.

10) C.R.

Steps 2) through 10) are repeated as often as necessary. After the last piece of data, the carriage return must be followed by a semicolon to indicate the end of the data tape. A location must be specified for each data word.

Both the coefficient and the exponent carry a sign. If no sign is used, the sign is assumed to be positive. For example, to enter the decimal numbers 1234.5, - 12.34 and .00012 into locations 0116 and the following format could be used:

location	coefficient	exponent
0116	.12345	4
0117	-.1234	2
0120	.12	-3

Leading zeros on the exponent, and trailing coefficient zeros need not be entered. A coefficient followed by a carriage return, or by two tabs, will indicate an exponent of zero. Spaces or non-numeric comments may be entered before, in, or after a data word. The letter x has a special function and should not be used in any comments made before the exponent field is terminated. Any kind of comment except a period or minus sign may follow the termination of the exponent reading.

To correct an error in preparing the tape, a delete code or an x may be used. An x following the data causes all previous information in the data word to be ignored, provided that the x is used before the exponent reading has been completed with a tab or carriage return. The x will not have any effect on the location field. An error in a location may be corrected only by using the delete code.

An entry -.0 followed by C.R. may be used as an end of file mark for program convenience in dealing with records of variable length. This "file mark" must be sensed by programming.

If instructions and data are punched on the same tape for convenience in reading, each group is terminated by a semicolon. A few inches of leader between them helps the operator to identify the end of the instructions and the beginning of the data.

Data tapes may be prepared without having the location specified. These tapes must be read into the desired locations under control of the user's program. How to enter data under program control will be discussed later.

2. To load prepared data tapes into tape specified locations:
    a. Insert data tape in reader, wide side nearer the console, and depress reader arm.
    b. Turn reader on
    c. Master Clear
    d. Set P to 7404
    e. Run

When the data tape has read in the following should be displayed on the console:

P = 7761 A = 0000 Z = 7722

A program error stop will occur if an address is out of range or if an exponent cannot be converted. To attempt reloading after an error stop: reposition the tape, master clear, set P to 7404 and run.

If the last data word is not followed by a carriage return and a semi-colon, the data tape will not stop. If it goes completely out of the reader, the input instruction of the Simulator's data reading subroutine is destroyed. In this event, either reload the Simulator or correct the input instruction as described in section B1b, but with P set to 7404 to enter data.

3. To load data directly from the typewriter:

   a. Master Clear

   b. Set P to 7406.

   c. Run

   d. Type in locations and data in the format used for Flexowriter tape.

   e. When data have been entered set Run switch to center position.

   f. Master Clear

If an error is made in typing a location, set Run switch to center, Master Clear, reset P to 7406, Run and start typing the location again.

If an error is made in a data word, type "x" and begin the data word again, provided the x is used before the exponent reading has been completed with tab or C.R. The x causes the routine to ignore the previous information in the data

word. It has no effect on the address. The "x" may be followed by a carriage return and/or a tab. The input routine looks for a minus sign, a decimal point, or a final semicolon, ignoring everything.

C. DATA INPUT UNDER PROGRAM CONTROL

Data input called for during a program may be from prepared paper tape or directly from the typewriter. Input and output during a program are obtained by using subroutines. In each case, the sub-routine is entered by a return jump. On completion of the sub-routine, an exit to the next upper instruction of the program is made. The return jump may be either a selective return jump (75 4 m) or an A jump (22 b m) as desired.

1. Input from paper tape, under program control. The word format for the data tapes differs from that described in section B1 in that the 1604 location need not be specified. Tapes that have been prepared with the location specified may be loaded under program control into any desired location because the subroutine recognizes a minus sign or period as the beginning of the data word. The format for data tapes to be loaded under program control is the same as that shown in Section B1a with steps b) and c) omitted: The tape is begun with a carriage return. Each data word

may then be entered in the following format:

sign. coefficient (tab) sign exponent (C.R.)

If the sign is negative, a minus sign is typed; if positive, the sign may be omitted. After the last piece of data, the carriage return is followed by a semicolon to mark the end of input.

The instruction used to enter the input subroutine is

7540 1224

This causes one data word to be read from the data tape into the A register of the Simulator. To read a block of date into sequential locations, a program is required.

As an example, a program is shown below. This program is written in the form of a single loop to store 150 data words in locations 0100 to 0326.

Location	Upper Instruction		Lower Instruction	
0023	(previous . . . .)		5050	0000
0024	7540	1224	0000	0000
0025	2050	0100	5000	0000
0026	5450	0225	7500	0024
0027	(next instruction..)			

The program example is based on the following reasoning:

The upper instruction in location 0023 may be any previous part of the program.

0023 lower - enter zero in index register 5

0024 upper - return jump to data input subroutine

lower = this instruction is not used

0025 upper - store data word in location (0100 + B) where B is the content of index register 5.

lower - pass to the next instruction

0026 upper - index skip. Compare the contents, B of index 5 with octal 0225 (the number of times the register should be incremented, which is the number of data locations minus one). If B is not equal to 0225, the instruction adds one to B and proceeds to the lower instruction. When B = 0225, index 5 is cleared to 0000 and the lower instruction is skipped.

lower - selective jump to location 0024. The next data word is read and stored in the next location in sequence.

0027 (next upper instruction) - When 226 octal locations (150 decimal) have been filled with data words,

the program proceeds to location 0027 for the next instruction of the program.

2. Data input from the typewriter, under program control. When the program calls for direct typed input, the letters "IN" will be displayed in the Status Display panel. The instruction to call for typewriter input is:

7540 1226

The data format is the same as that described for paper tape in Section C1. An x may be used to correct typing errors in the data word, if used before the carriage return. The input subroutine reads one data word from typed input. To read a block of data into sequential locations, a program loop is required. See Section C1 for an example of such a loop.

If input from the typewriter is called for more than once during the operation of a program, the operator needs to know which of the inputs is desired. This may be accomplished in several ways. One of these is for the program to contain instructions to position the typewriter so input would follow one or more carriage returns or one or more tabs. The position of the typewriter carriage would thus identify the input called for by the program. Another method is for the programmer to use a subroutine which would type a tabulating

number when data input is called for. The subroutine is entered by a return jump and the tabulating number typed would be the contents of the execution address plus B (where B is the content of the disignated index register). The instructions to enter the subroutine are 7540 (addr.) upper, return jump to first address of subroutine, 00b0 y lower, tabulating number (y + B) where y is a four digit octal number and B the contents of the designated index register, b.

D. DATA OUTPUT UNDER PROGRAM CONTROL

The results of computations may be punched on paper tape or typed out by the typewriter. In either case, output is handled by a subroutine in the Simulator. The subroutine is entered by a return jump. On completion of the output subroutine, an exit is made to the next upper instruction of the main program.

1. Output Punched on Paper Tape.

The output subroutine punches the word contained in the Simulator accumulator. During this operation, the contents of the accumulator are lost. Therefore, if an answer is to be used in later computations, it should be stored before the punch instruction is given. Two output formats are provided. To obtain a data word followed by a carriage return, the instruction is:

7540 1236

For data followed by a tab, use

7540 1234

SUBROUTINES

Subroutines are always entered by using a return jump. There are three instructions which have return jump capabilities, viz. the Selective Jump, the "A" Jump and the Selective Stop. Normal exits from subroutines are usually made through use of a normal jump. The following is an example of a subroutine to square the contents of the accumulator.

Program to enter "square" subroutine				Subroutine to square the contents of the accumulator			
	program		Square	SLJ	0	0
				STA	0	Below
Trash	SLJ	4	Square	FMU	0	Below
	0	0	0	SLJ	0	Square
	STA	0	Garbage Below	0	0	0
	SLS	0	Wait	0	0	0
Garbage	0	0	0			
	0	0	0			
Wait	program					

At "TRASH" a return jump is made to the subroutine i.e., "SQUARE". The return jump causes the address "TRASH + 1" to be inserted in the address field of the upper instruction of "SQUARE". The return jump then jumps to the lower instruction of "SQUARE". When the subroutine has finished squaring the number a normal jump is made to "SQUARE". The upper instruction of square now reads "SLJ O TRASH +1". Hence, the exit from the subroutine is made to the next upper instruction following "TRASH". Had the instruction "SLJ 4 SQUARE" been the lower instruction of "TRASH" the exit from the subroutine would still have been made to the upper instruction of "TRASH + 1".

OFFSET LOADING

For some programs especially subroutines, it is desirable to be able to relocate them.

To allow the programmer to do this with a minimum amount of difficulty, offset loading capabilities have been incorporated in the instruction load routine and the data load routine.

To illustrate the usage of offset loading consider the following program to transfer five numbers from one area in memory to another.

	Symbolic Coding				Absolute Coding		
Start	PAS	0	0	0020	50	0	00000
	ENI	1	0		50	1	00000
	LDA	1	A	0021	12	1	00030
	STA	1	B		20	1	00040
	ISK	1	M-1	0022	54	1	00004
	SLJ	0	Start + 1		75	0	00021
	SLS	0	Start	0023	76	0	00020
	PAS	0	0		50	0	00000
A		al		0030	.1	1	
A + 1		a2		0031	.34	2	
A + 2		a3		0032	.6	0	
A + 3		a4		0033	.85	9	
A + 4		a5		0034	.0	0	

To place the above program beginning in location fifty it would be necessary to type the program in the following manner:

	Absolute Coding		
0020	50	0	00000
	50	1	00000
0021	12	1	00030/
	20	1	00040/
0022	54	1	00004
	75	0	00021/

```
0023    76      0     00020/
        50      0     00000
```

To use the instruction load routine for paper tape, it is necessary to set P = 7400. To load the above instructions at location 0050 it is necessary to set P = 7400 and set A = 0030, i.e., the increment. The load routine adds the increment to the locations, i.e., 0020, 0021, etc. The load routine also adds the increment to those addresses in instructions which are followed by a slash.

Since the addresses in location 0051 have been modified, it is therefore necessary to load the data beginning at location 0060 rather than at location 0030.

To use the data load routine for paper tape, it is necessary to set P = 7721. The data tape is prepared as originally indicated. To load the data tape beginning at location 0060 rather than at location 0030, it is necessary to set P = 7721 and A = 0030, i.e., the increment. The data load routine than adds the increment to the locations, i.e., 0030, 0031, etc.

If the programmer desires to offset certain locations and not offset others, a minus is used to indicate those locations which are <u>not</u> to be offset. Consider the following example.

```
-0020     75     0     00021/
          50     0     00000
0021      12     0     00023/
          20     0     00024/
0022      75     0     00025
          50     0     00000
```

If, for example, an increment of 0020 is used, the first 1604 word will still be placed in location 0020. The following two 1604 words will be placed in locations 0041 and 0042.

The final property necessary for offset loading Simulator programs is the capability of modifying 160 addresses in Simulator programs containing both 1604 instructions and 160 instructions. 160 addresses which must be modified are followed by a period. Consider the following example.

```
0020      12     0     00300/
          51     0     00023/
0021      20     3     00102
          67     0     16002
0022      70     0     17101
          03     0     00000
0023      75     4     01066
          00     0     00000
```

0024	76	0	00020/
	00	0	00000

In the above example those addresses followed by a slash are offset by the increment. The 160 address, 0300, os offset by 4 times the increment. This is consistent with the fact that 4 160 words are needed to make up 1 1604 word.

BASIC 160 INSTRUCTIONS IN A FPP-33 PROGRAM

When it is desirable to write a subroutine or some other part of a program in basic 160 language, the command used to exit from the FPP-33 to basic 160 is:

510 b m

No index register is used. A breakpoint may be used if desired. The execution address is the address of the first instruction to be executed on normal re-entry to FPP-33. The instruction immediately following the "exit to 160" instruction is the first 160 instruction to be executed.

Normal re-entry to FPP-33 is effected by the 160 instruction 7001. The next FPP-33 instruction to be executed is the upper instruction of the location specified by the address of the last executed "exit to 160" command. An example of exit to 160 and re-enter FPP-33 follows:

Location	Instruction		
0021	5100	0023	exit to 160, 0023 is the FPP-33 address of the first instruction to be executed on re-entry.
	0504	----	first basic 160 instruction
0022	7001	0000	re-enter FPP-33
	0000	0000	This is not used. An upper instruction is the first to be executed on re-entry.
0023	(	)	next instruction of FPP-33 program

In using exit to 160 commands, remember that one Simulator location is equivalent to four basis 160 locations. A basic 160 instruction contains four octal digits.

C. TO OFFSET LOAD A PROGRAM THAT CONTAINS 160 INSTRUCTIONS

The program is offset by placing the desired increment in the A register as before. This modifies the storage locations of the instructions.

The following codes are used in preparing a routine that can be offset loaded:

> A slash following an address: the load routine adds the increment which was entered in the A register to the address.

- A minus sign preceding a location indicates that this location is not to be offset.

. An address followed by a period is offset by four times the specified increment. This allows 160 insertions in a 1604 program to be offset loaded by the same load procedure.

; A semicolon is used to indicate the end of the routine.

The program steps in the following example may be offset loaded:

0020	1200	0300/	1604 instructions
	5100	0023/	exit to 160, set normal return for 0023/
0021	2030	0102	160 instructions
	6701	6002	7001 is a normal return, 7101 is a special return to 0300.
0022	7001	7101	
	0300	0000	(which is the 1604 address 0060/)
0023	7540	1066	
	0000	0000	
0024	7600	0020/	1604 instructions
	0000	0000	

D. STANDARD SUBROUTINE ENTRY AND EXIT

The practice that a subroutine is entered by a return jump instruction has been adopted. Three instructions in the Simulator have return

jump possibilities, namely the A jump, the selective jump and the selective stop. The usual exit from a subroutine is by means of a pair of normal jumps, one at the end of the subroutine and the other in the upper instruction of the first location. The return jump may be located in either an upper or a lower instruction, but the first instruction of the main program to be executed after performing a standard subroutine is always the upper instruction of the next 1604 word after the one which contained the return jump.

Example: return jump to subroutine.

Location	Instruction	
0076	1210 0026	load A with the contents of address (0026 + B)
	7540 0125	return jump to location 0125. This is an exit to a subroutine whose first instruction is in location 0125.
0077	2010 0526	next step in main program, store the function of x in address (0526 + B).
	The subroutine:	
0125	7500 0000	The return jump causes the next location in the main program (0076 + 1) to be inserted in the address field of the upper instruction of the first

subroutine location. This instruction then becomes 7500 0077, a jump to location 0077.

( ) The lower instruction of location 0125 is the first instruction of the subroutine.

The last instruction of the subroutine is a normal jump to the entry location 0125 (7500 0125). The upper instruction in this location now reads jump to location 0077, an exit to the next upper instruction of the main program.

When control is returned to the main program, the desired function is usually in the Simulator accumulator A and may be stored or operated on in the next sequence of instructions of the main program.

E. MATHEMATICAL SUBROUTINES FOR USE ON 33-BIT FLOATING POINT PACKAGE (FPP-33)

The subroutines are written in a combination of FPP-33 instructions and basic 160 instructions. All subroutines are relocatable in the available storage locations by use of the offset load routine of FPP-33. All of the subroutines are on punched paper tape in Flexowriter code (which is used for the offset load routine) and if a listing of the routine is desired, it may be obtained by typing out the tape on the Flexowriter.

The subroutine available are:

ARCTANGENT X

SINE X

COSINE X

ARCCOSINE X

ARCSINE X

SERIES EXPANSION (BASIC)

EXPONENTIAL (2^X, e^X, 10^X)

SQUARE ROOT

LOG TO BASE 2

PLOT

All routines are written as having location 00000 as the starting point and must be offset loaded to the desired location in the FPP-33 program. Offset locations are indicated by a slash (/) as for example, 0034/ refers to a location which is 34B locations following the base location to which a routine is offset loaded.

Routines are entered by performing a return jump (7540 xxx) instruction to the location specified as the entrance of the subroutine. The return jump may be in the upper or lower instruction of the particular word. Return from the subroutine will, in all cases, be to the upper instruction of the next word of the main program.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine: Arctangent
IDENTIFICATION NUMBER: FPP 33-1
PROGRAMMER: Payne

B. PURPOSE

Given a number X. Find the arctangant of X using the Maclaurin series as given on page 137 in Hastings.

C. USAGE

1. Operational Procedure:

   If $-1 \leq X \leq 1$, use the series directly.

   If $X > 1$, then take Arctangent of $1/X$ and subtract this value from $\pi/2$ to get arctangent of X..

   If $X < -1$ then subtract Arctangent $1/X$ from $-\pi/2$ to get Arctangent of X.

2. Entry:

   A contains the number X in floating point

3. Exit:

   A contains the answer in r adians

   B6 is used, but is reset to its original value before exit.

4. Error Conditions:

   None

5. Subroutines Used:

   Basic

6. Remarks and Restrictions:

Uses 51_8 permanent and 0 erasable locations.

Basic is located at 0040/ (incorporated on an offset Flex tape)

A. IDENTIFICATION
TITLE: FPP-33 Subroutine: Sine Cosine
IDENTIFICATION NUMBER: FPP-33-2
PROGRAMMER: Payne

B. PURPOSE

Given X, compute the Sin X or Cos X (where X is in radians)

C. USAGE

1. Operational Procedure:

   The framework for the program is around the Sin X routine.

   Cos X uses the Sin X as a subroutine where Cos X = .

   Sin ($\pi/2$ + X)

2. Entry:

   A contains the angle in radians (X)

3. Exit:

   A contains the value of Sin X or Cos X.

   B6 is used, but reset to its original value before exit.

4. Error Conditions:

   None.

5. Subroutines Used:

   Basic is incorporated in Sin X. Cos X incorporated both

   Basic and Sin X.

6. Remarks and Restrictions:

   Sin X uses 44 permanent FPP-33 locations and 0 erasable.

   Cos x uses 4 permanent FPP-33 locations and 0 erasable.

This is an offset program on flex tape with Sin X at 0000/ and Cos X at 0045/. Basis is at 0033/. (entry addresses). Accuracy is within 1 or 2 in the ninth decimal place.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine: Arcsine, Arccosine
IDENTIFICATION NUMBER: FPP 33-3
PROGRAMMER: Payne

B. PURPOSE

Given a number X, this subroutine will find the Arccos of X or the Arcsin of X.

C. USAGE

1. Operational Procedure:

The formula for the program is found on page 163 in Hastings.

Arccos X = $\sqrt{1-x}\,\psi(X)$, where $\psi(X) = a_o + a_1 x + a_2 x^2 + \ldots + a_7 x^7$

Arcsin X is evaulated by using Arccos X as a subroutine

where Arcsin X = $\pi/2$ - Arccos X.

2. Entry:

A contains the number X.

3. Exit:

A contains Arcsin X or Arccos X.

B6 is used, but reset to its original value.

4. Error Conditions:

None.

5. Subroutines Used:

Arccos X incorporated SQRT and Basic.

Arcsin X incorporated SQRT, Basic, and Arccos X.

6. Remarks and Restrictions:

Arccos X uses 55 permanent locations, 0 erasable locations.

Arcsin X uses 5 permanent locations, 0 erasable locations.

This is an offset program on flex tape entry is a Arccos X 0000/ Arcsin X 0055/. Basic is used to evaluate the series. The SQRT subroutine is also incorporated with the following entry points: Basic 0022/, SQRT 0034/.

Accuracy is within 3 or 4 in the eighth decimal place.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine - Basic (series expansion)
IDENTIFICATION NUMBER: FPP 33-4
PROGRAMMER: Mansfield

B. PURPOSE

Compute the value of a given series $\sum_{i=o}^{n} C_i x^i$ of n terms.

(n = 1, 2 ... n)

C. USAGE

1. Operational Procedure:

A code word must have been previously loaded in A before going to Basic. The code word contains in octal NNNOXXXX OOOOTTTT for which the following interpretations are used.

NNN is the number of terms in the series not including C_0.

XXXX is the address of the floating point number X to be used by the series.

TTTT is the address of the first coefficient C_O in the table of coefficients.

0000 are trash and ignored by the subroutine.

2. Entry:

A contains the code word.

3. Exit:

A contains the answer

B6 is used, but reset to its original value before exit.

4. Error Conditions:

   None

5. Remarks and Restrictions:

   Uses 11_8 FPP-33 locations (permanent) and 0 erasable locations. This routine is on offset flexowriter tape.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine - Exponential
IDENTIFICATION NUMBER: FPP 33-5
PROGRAMMER:

B. PURPOSE

Calculate 2^X, e^X, or 10^X

C. USAGE

1. Operational Procedure:

This basic subroutine calculates 2^X, but e^X and 10^X may be calculated by performing one preliminary multiplication on X using a constant contained in the subroutine. The multiplication factors for e^X and 10^X are stored in the subroutine as shown:

e^X factor is stored at 00338 plus the starting address

10^X factor is stored at 00348 plus the starting address

As an example of the use of this subroutine, it is assumed that the subroutine has been offset loaded by 07008. The main program to enter the subroutine by a return jump is as follows for 2^X:

Loc.	Upper inst.	Lower inst.	Remarks
0400	1200 0100	7540 0700	Load x in accumulator from 0100 and Jump to the subroutine.

For e^x a possible entrance is

Loc.	Upper inst.		Lower inst.		Remarks
0377	1200	0100	3200	0733	Load x into acc. Mult. by factor in 0733 (in subroutine)
0400	7540	0700	0000	0000	Return jump to subroutine, lower instruction is skipped.
0401	2000	0200			On return from subroutine, store e^x in 0200, the lower instruction is up to you.

2. Entry:

   A contains the number x (premultiplied if e^x and 10^x are desired)

3. Remarks and Restrictions:

   Entry is by a return jump to the first address in the subroutine. 35^8 permanent locations are required. This routine is on an offset flex tape.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine-Square Root
IDENTIFICATION NUMBER: FPP 33-6
PROGRAMMER:

B. PURPOSE

Given a number X in the accumulator, find the square root of the number by the use of the Newton iteration method and leave the square root in the accumulator.

C. USAGE

1. Entry:

   A contains the number X in floating point

2. Exit:

   A contains the square root in floating point

3. Remarks:

   This routine used 21 permanent locations and is on offset flex tape.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine-Log to Base 2
IDENTIFICATION NUMBER: FPP 33-7
PROGRAMMER:

B. PURPOSE

Given a floating point number in A, calculate the log to the base 2 of this number.

C. USAGE

1. Entry:

   A contains the number X in floating point.

2. Exit:

   A contains the log to the base 2 of the value of X.

3. Remarks and Restrictions:

   Routine uses 478 permenent locations and is on an offset flex tape.

A. IDENTIFICATION
TITLE: FPP-33 Subroutine-Trig Routines
IDENTIFICATION: FPP 33-8
PROGRAMMER:

B. PURPOSE

This tape combines the following subroutines with the given entrances:

SIN X	0000/
COS X	0045/
ARCTAN X	0051/
BASIC	0033/

C. USAGE

See writeups of the original routines.

A. IDENTIFICATION

TITLE: FPP-33 Subroutine: Plot
IDENTIFICATION: FPP 33-9
CATEGORY: Mathematical Subroutine
PROGRAMMER: H. Theiste
DATE: March, 1961

B. PURPOSE

The purpose of this routine is to plot results on the on-line plotter in either of two ways:

1. Moving from previous plot point to present plot point in a straight line with pen down.
2. Moving from previous plot point to present plot point with pen up, plot a symbol to represent desired point.

As written, the routine uses output from routines written in 1604 simulator language. The method is applicable for plotting either fixed or floating point results.

C. USAGE

1. Operational Procedure -

Before any plotting can be done, the routine must be provided with the X and Y scale factors. The scale factor (F) is determined by dividing 100 by the number of units per inch on the plotted output. Thus, if the X scale were 1" = 4 units and the Y scale were 1" = .5 units, the scale factors are

$$F(X) = \frac{100}{4} = 25$$

$$F(Y) = \frac{100}{0.5} = 200$$

These are stored in locations (0073) and (0074) respectively. It is the task of the user to set the X and Y scale factors in these locations.

After the scale factors have been set up, the initial values of X_o and Y_o must be provided to the routine. This is accomplished by a return jump to location 0070/of the plot routine with the address of X_o in B6.

Two entries are provided for plotting results from the 1604 simulator routine, depending on whether the plot is to be a line plot or a point plot. In either case, X and any Y values must be stored in consecutive locations with X first.

If a line plot is desired, i.e., move between points with the pen down, a return jump is made to location 0000/with the address of the X value in B6. The pen will move from its present position to the point X, Y along a nearly straight line, and will exit.

If a point plot is desired, i.e., move between points with the pen up and plot a symbol at the desired point, a return jump is made to location 0110/with the address of X in B6. This routine uses B1, but restores the original contents.

To load the plot subroutine:

a. FPP-33 routine must be in the computer.

b. Turn on reader and insert PLOT tape anywhere on leader.

c. Set P = 7400

d. A = First 1604 location of subroutine

e. Press Run Switch

f. Line Plot
Set B6 to location of X-coord. Return jump to first location of Plot routine.

g. Point Plot
Set B6 to location of X-coord. Return jump to location 0110/of Plot routine.

h. Set X_o and Y_o
Set B6 to location of X_o-coord. Return jump to location 0070/of Plot routine.

3. Space Required -
   110_8 FPP-33 locations = 440_8 or 288_{10} 160 locations.

4. Temporary Storage -
   The point plot routine uses B1, but restores the original contents before exit. The contents of B6 remain unchanged by the routine. Therefore, if several points are to be plotted, the address of X will always be in B6. Also X and Y are left unchanged by the routine.

11. Accuracy -
    The plotted point will be accurate to the nearest 1/100th of an inch.

13. Equipment Configuration -
    Minimum 160 computer with California Computer Products Company Plotter.

D. METHOD

To plot a given point, the routine multiplies each coordinate (X_L or Y_L) by its scale factor and retains only the integer portion of the coefficient of the floating point number, discarding the fractional portion. It then subtracts the previous coordinates to obtain values of ΔX and ΔY, representing actual pen motion.

The routine then determines which value has the greater magnitude, ΔX or ΔY. This indicates along which axis most of the pen motion will occur. The larger quantity (in magnitude) is called R_L and the other L.

R_L is divided by L to obtain the ratio R between the two motions. The directions of pen motion for R_L and L are determined and two plotter outputs are set up P_1 and P_2. P_1 to move the pen one unit in the R_L and L directions and P_2 to move the pen only in the R_L direction. The three quantities R_L, L and R are made positive to serve as counters and a fourth quantity T is obtained by R-2.

L is decremented by one and, if it remains positive, a P_1 output is given to the plotter; whereupon T is decremented by one and, if it remains positive, R_L is decremented by one and a P_2 pulse is given to the plotter. If, however, T goes negative R is added to the negative quantity, R_L is decremented by one and process is repeated starting with decrementing L by one. If L goes negative, and if R_L is at least one, R_L is decremented and a P_2 output is given to the plotter until R_L goes negative, at which time the X and Y coordinate values are updated and an exit is made from the routine.

160 LOW SPEED EQUIPMENT

SELECT		STATUS RESPONSE	
4102	Select PT reader	0010	Typewriter not ready
4104	Select PT punch		
4210	Select typewriter output		
4220	Select keyboard input		
4240	Request typewriter response		

163 and 164 MAGNETIC TAPES

SELECT		STATUS RESPONSE	
111x	Write	0001	Coded mode
113x	Read forward	0002	Not ready
112x	Backspace	0004	Parity error in last operation
116x	Rewind	0010	Program error
115x	Rewind unload	0020	End-of-file mark read
114x	Request tape status	0040	End of tape or load point

x = 1, 2, 3 or 4 for tape units
For alternate MTS use 12 --
For assembly mode use 21 --
For alternate MTS assembly mode use 22 --

160 COMPUTER PROGRAMMING TRAINING PROBLEMS

The following series of problems is designed to introduce the programmer to the various features of the 160 Computer in an orderly sequence.

Problem 1: This problem is designed to show the addressing modes of the 160 Computer. Write the problem with the following specifications:

Add two numbers A and B together and store the results in location C.

a. - A is in location 0040, B is in location 0065, C is in location 0072. The program is stored somewhere between locations 1000 to 1100.

b. - A is in 1000, B is in 1002, C goes to 1005. Program is in the range of locations 1010 to 1100.

c. - A, B and C as in b. Program is in 0700 to 0777.

d. - A, B, C as in b. Program is in locations 4000 to 4100.

NOTE:
Problem a. can be written using direct addressing.
Problem b. can be written as backward relative addressing.
Problem c. can be written as forward relative addressing.
Problem d. can be written as indirect addressing.

Problem 2: This problem is designed to show the conditional jump ability of the 160 computer.

a. If A is equal to B, do C plus D

b. If A is larger than B, do C minus D

c. If A is less than B, do D minus C

NOTE: In all cases store the sum at E, stop if condition is not satisfied.

The following conditions apply: Location 0040 holds A
Location 0132 holds B
Location 1000 holds C
Location 1750 holds D
Location 0050 holds E

The program is written to start at location 0140 or higher.

Problem 3: This is a problem in controlling the execution of a subroutine and loop control. A tape with the program will be provided. Execute the routine which starts at location 1000. This routine ends with the instruction 7007 (i.e., the routine exists to the location specified in location 0007).

a. Execute the above routine 10 times (decimal). The program is written to start at location 2035.

b. Execute the routine 100 times (decimal). The control program starts at location 4070.

Problem 4: Given 100 (decimal) numbers in successive storage locations starting at location 0476. Add up this list of numbers and store the sum at location 0050. The program starts at location 4730.

Problem 5: Given 100 (decimal) pairs of numbers which are stored starting at location 1000 and 2000 respectively. Add the pairs and store the sums in a list starting at location 3000. (i. e. add the number in 2000 and store the sum at 3000 and do it for the next locations, etc.)

Problem 6: Read in information from punched paper tape until the code octal 45 is encountered, then read in the next ten frames of tape and store the information in locations 0150 and up.

Problem 7: (Problem 7 is designed to show use of paper tape input and output.)

a. Duplicate paper tape from the reader to the punch.

b. Duplicate paper tape with levels 6 to 1 only (no seventh level)

c. Generate a binary output pattern on punched paper tape. (the binary output pattern consists of all possible binary configurations of hole and no hole on paper tape.

Problem 8: (Sideways add) Count the number of ones contained in the accumulator on starting the routine. Display the binary count in the A register after stopping the machine.

Problem 9: (Table lookup) Take the number contained in the A register and see if it is in a table of numbers in locations 1000 to 1011. If the number in A is in the table, stop and display the address of the number in the table. If the number is not in the table, stop the machine with all zeros in A.

TABLE OF POWERS OF 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE

4000 (Octal)	2048 (Decimal)
to	to
4777 (Octal)	2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 (Octal)	2560 (Decimal)
to	to
5777 (Octal)	3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.00109	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

A. IDENTIFICATION

TITLE: Business Calculations Komplete

IDENTIFICATION: BCK

CATEGORY: Decimal Computation

PROGRAMMER: H. Theiste

DATE: March, 1961

B. PURPOSE

BCK refers to a technique of representing decimal numbers in the 160 Computer to combine the advantages of decimal arithmetic with the binary nature of the 160 Computer.

C. USAGE

Each word of data in BCK is the binary representation of three consecutive digits of a decimal number. By combining 2, 3, or 4 consecutive BCK data words we can obtain numbers of 6, 9, or 12 decimal digits with the low order BCK word in the lowest addressed location and higher order BCK words in higher addressed locations. Examples of the BCK representation of some decimal numbers are illustrated below (assume storage beginning at location 1000).

Decimal Number	BCK	
	Location	Contents
654	1000	1216
654,321	1000	0501
	1001	1216
12, 345, 678, 901	1000	1605
	1001	1246
	1002	0531
	1003	0014

The BCK range for one address in octal notation is 000 - 1747.

Negative numbers in BCK are expressed in tens' complement form. The ten's complement is obtained by subtracting the number from the next higher power of 10, e.g., 1, 000 for a 3-digit number, 1,000, 000 for a 6-digit number, etc. The three numbers in the previous example would be represented negatively as follows:

Decimal Number	BCK Location	BCK Contents
-654 (346)	1000	0532
-654, 321 (345, 679)	1000	1247
	1001	0531
-12, 345, 678, 901 (987, 654, 321, 099)	1000	0143
	1001	0501
	1002	1216
	1003	1733

This notation implies that the range of the most significant word, W_H, of a number in BCK is $-(500)_{10} < W_H < (500)_{10}$. That is, if the most significant word of a BCK number is $(500)_{10} = (0764)_8$ or greater, the BCK arithmetic routines will treat the number as a negative number.

1. Principle of Operation - BCK consists of a control routine and a number of arithmetic and logical subroutines. Through the use of the control routine, it is possible to link together a number of BCK subroutines without the necessity of generating a subroutine linkage in the main program each time a routine is used. If desired, basic 160 Computer coding may be used at any time.

   The BCK subroutines are designed to operate on up to six specification parameters which will be called P, A, B, C, D, E in the following discussion.

2. Control Routine - The control routine provides the means of tying BCK subroutines together and performs the function of getting the parameters from memory to a standard location for use by the subroutines.

The two entrances to the control routine are DO and INTERP. The DO entrance causes the specified BCK subroutine to be executed once with control to be returned to 160 coding on completion. The INTERP entrance causes the control routine to continue to interpret calling sequences until the subroutine BASIC is called for.

A calling sequence consists of the address of a subroutine followed by the parameters required by the subroutine.

Examples -

To perform a multiply operation and return to basic 160 Computer coding a DO entrance is used. It is necessary that on entrance to the control routine, the A register of the 160 contains the address of the location which specifies the address of the subroutine to be executed. The parameters for the subroutine are in successive locations after the location specifying the address of the subroutine. The following coding (expressed in Assembly Language) will do the job.

Location	Op	Address Additive
L1	LDF	02
L2	JPI	DO
L3		NEXT
NEXT		MPY
		0404
		FACTOR
		P2
		RES
L4	JFI	01
		LOOP

In the INTERP mode, the subroutine calling sequences are listed sequentially. Thus, if it is required to add the 6-digit numbers A and B, store the result at C, multiply by the 9-digit number D and return to basic 160 coding, the following sequence is used (expressed in Assembly Language).

Location	Op	Address	Additive
L1	LDF	02	
L2	JPI	INTERP	
L3		CODE	
CODE		ADD 6	
		A	
		B	
		C	
		MPY	
		0203	
		C	
		D	
		E	
		BASIC	
L4 (etc.)			

3. BCK Subroutines

(a) ADDITION

Calling Sequence

```
ADDX
A
B
C
```

Description - The X in the name of the subroutine may be 3, 6, 9 or 12 to specify operation on 3, 6, 9 or 12 decimal digit numbers expressed as 1, 2, 3 or 4 BCK words. Operation is to add the BCK number starting at location A to the BCK number starting at location B and store the sum starting at location C.

(b) SUBTRACTION

Calling Sequence

```
SUBX
A
B
C
```

Description - Subtract the BCK number starting at location B from the BCK number starting at location A and place the difference starting at location C.

The X in the name of the subroutine may be 3, 6, 9 or 12 as in the ADDX subroutine.

(c) MULTIPLICATION

Calling Sequence

```
MPY
P
A
B
C
```

Description - Multiply the BCK number starting at location A by the BCK number starting at location B and store the product starting at location C.

The parameter P is coded as a four-digit octal number where the digits are WXYZ with the following interpretation.

WX - indicates the number of BCK words required to represent the number at A. X must be at least 1 and no more than 4.

YX - indicates the number of BCK words required to express the number at B. Z must be at least 1 and no more than 4.

The product stored starting at C will occupy WX + YZ BCK words. The multiplication takes place as if the numbers were whole numbers. The location of any assumed decimal point in the product follows the normal rules of decimal arithmetic.

(d) DIVISION

Calling Sequence

```
DIVIDE
P
A
B
C
```

Description - Divide the BCK number starting at location A by the BCK number at location B. Store the quotient starting at location C.

The parameter P is coded as a four-digit octal number where the digits are WXYZ with the following interpretation.

WX - indicates the number of BCK words required to represent the number at A. WX must be at least 2 and no greater than 10 (decimal 8).

YZ - indicates the number of BCK words required to express the divisor at location B. YZ must be at least 1 and no more than 4.

The division follows the rules of decimal arithmetic and the quotient stored at C will consist of WX - YZ BCK words. The remainder from the division, if desired, may be found starting at location REM and consists of WX BCK words.

(e) COMPARISON

Calling Sequence

```
COMPAR
P
A
B
C
D
E
```

Description - The quantity starting at location A is compared with the quantity starting at location B. As a result of the comparison, a transfer of control will take place as follows:

If A is greater than B, the next instruction will be obtained from location C. If A is equal to B, the next instruction will be obtained from location D. If A is less than B, the next instruction will be obtained from location E.

The parameter P is coded as a four-digit octal number where the digits are WXYZ with the following interpretation.

XYZ - indicates in octal the number of words to be compared in A and B.

W=O - indicates that the numbers being compared are in BCK notation and that an algebraic comparison is to be made.

W=4 - indicates that the numbers being compared are in input/output form and have the high order digits first. This is a magnitude comparison only.

(f) DATA MOVING

Calling Sequence

```
MOVE
P
A
B
```

Description - BCK information will be moved from the area starting at location A to the area starting at location B.

The number of words to be moved and the operation are controlled by the parameter P, coded as a four-digit octal word. The octal digits are WXYZ with the following interpretation.

XYZ - indicates the number of words to be moved. XYZ is in octal notation.

W=0 - indicates words are selected from A, A + 1, A + 2, etc., and are stored at B, B + 1, B + 2, etc.

W=4 - indicates words are selected from A, A - 1, A - 2, etc., and are stored at B, B - 1, B - 2, etc.

(g) UNCONDITIONAL TRANSFER

Calling Sequence

```
GOTO
A
```

Description - The next instruction to be executed will be obtained from location A.

(h) RETURN TO BASIC COMPUTER CODING

Calling Sequence

```
BASIC
```

Description - The next instruction to be executed will be in basic 160 Computer code and is in the next location.

(i) CODE CONVERSION

Calling Sequence

```
RECODE
P
A
B
```

Description - Change the coding of information starting at location B by a table lookup and simple substitution using the table starting at A.

The operation of the process is controlled by the parameter P which is coded as four octal digits WXYZ. The interpretation of the parameter is as follows:

XYZ - is the count of the number of words to be converted expressed in octal.

W=0 - indicates that the six-bit information is stored one character per word in the lower six bits of a word.

W=4 - indicates that the information is stored in the form of two six-bit characters per word.

(j) DECIMAL TO BCK CONVERSION

Calling Sequence

```
BCDBCK
P
A
B
```

Description - Convert the decimal information starting with the higher order digit in location A and lower order digits in higher storage locations to BCK form and store in memory starting at location B.

Operation of the process is controlled by the parameter P which is coded as WXYZ. WXYZ is interpreted as an octal number which specifies the total number of decimal digits to be converted.

The number of BCK words generated as a result of the conversion will be A/3 or A/3 + 1, whichever is required to provide sufficient storage.

If there is not an even multiple of 3 decimal digits available for recoding, high order zeros will be assumed.

(k) BCK TO DECIMAL CONVERSION

Calling Sequence

```
BCKBCD
P
A
B
```

Description - Convert the BCK information starting at location A to decimal information and store the result starting at location B. B specifies the location of the high order digits of the converted result with lower order digits placed in successively higher storage locations.

Operation of the routine is controlled by the parameter P which is coded as WXYZ. WXYZ is an octal number which specifies the total number of BCK words to be converted.

The number of decimal digits generated as a result of the conversion will be A x 3.

4. Space required: $(1046)_8 = (550)_{10}$ locations. In addition, locations 0040-0067 are used by the routine. Each subroutine is an entity in itself so only as much of the package as is needed must be loaded into memory. Only locations 0057, 0066, and 0067 need be kept permanently.

9. Input: Three tapes are available for loading BCK into memory.

(a) Bioctal tape which includes RS-22. This tape is loaded beginning at 6332 and ends at 0067.

(b) Bi-octal tape of BCK only. Beginning location is 6731, ending location is 0067.

(c) OSAP symbolic tape for assembly. Must be preceded by an ORG instruction.

10. BCK Subroutines

Name of Subroutine	Starting Location 6332	6731	Execution Time
ADD 3	6366	6765	.27 ms
ADD 6	6365	6764	.48-.54 ms
ADD 9	6364	6763	.69-.79 ms
ADD 12	6363	6762	.90-1.03ms
SUB 3	6421	7020	.26 ms
SUB 6	6420	7017	.45-.54 ms
SUB 9	6417	7016	.65-.79 ms
SUB 12	6416	7015	.85-1.03ms
MPY	6740	7337	8-36 ms Variable
COMPAR	6622	7221	.32 ms
MOVE	6713	7312	Inc=6.4(12+21W)ms Dec=6.4(15+21W)ms
GOTO	6704	7303	44.8 us
BASIC	6710	7307	38.4 us
RECODE	6450	7047	Variable
BCDBCK	6521	7120	.41ms min.
BCKBCD	6561	7160	.40ms min.
DO	0066	0066	
INTERP	0067	0067	

A. IDENTIFICATION

TITLE: Calculational Interpretive Programming System

IDENTIFICATION: CALINT 3-61

CATEGORY: Floating Point Decimal Computation

PROGRAMMER: J. A. Pederson

DATE: March, 1961

B. PURPOSE

CALINT is an interpretive programming system for the 160 Computer. It is particularly suitable for calculations involving formula evaluation and parameter studies. CALINT performs arithmetic in floating point decimal format. Thus, in programming it is only necessary to specify the decimal point in any convenient form and the system will take care of positioning throughout operation.

Input of the program and data is either via the on-line typewriter or by punched paper tape prepared off-line on a Flexowriter. Output is to the typewriter or on paper tape for listing on a Flexowriter.

C. USAGE

1. Theory of Operation - CALINT instructions consist of the basic arithmetic operations, decision operations and single valued higher function such as sine, cosine, square root, etc.

   The result of most operations usually goes to the CALINT accumulator. Hence, instructions are considered to be in the form

   A = B opn C

   where A is the CALINT accumulator, B is the first operand, C the second operand and opn the operation to be performed.

2. Operation Codes - The instructions available in CALINT are listed below. Detailed descriptions of the individual instructions are given in a later paragraph.

Instruction	Code	General Form	Operation
add	+ (,)	A = B + C	B + C → A
subtract	-	A = B - C	B - C → A
multiply	×	A = B × C	B C → A
divide	/	A = B / C	B ÷ C → A
transfer positive	pg	pg C	if A ≥ 0, go to C
transfer negative	ng	ng C	if A < 0, go to C
transfer zero	zg	zg C	if A = 0, go to C
pass	ps	ps	do nothing
subroutine entry	se	B se C	C → A, enter subroutine at B
square root	rt	rt C	$\sqrt{C} \rightarrow A$
sine	si	si C	sin C → A
cosine	co	co C	cos C → A
tangent	tn	tn C	tan C → A
arctangent	at	at C	atan C → A
logarithm	lg	lg C	log C → A
exponential	e	e C	$e^C \rightarrow A$
subroutine return	sr	sr C	return to main program
input	in	in C	one number from input device → C
output, tab	ot	ot C	one number from C, followed by a tab, → output device
output, carriage return	oc	oc C	output one number from C, followed by CR, → output device

Instruction	Code	General Form	Operation
unconditional transfer	go	go C	unconditional transfer to C
data transfer	p	Bp C	B→C
clear C	z	z C	zero's→C
execute machine code	mc	mc C	next instruct in machine code
set index register	sn	sn C	set index register n to C
test index register	nt	nt C	index register n increased by 1 ≤ C execute next instruction; > C execute second following instruction.
halt	h	h	stop, type address of halt instruction

3. Storage Allocations - Three areas are provided in storage for CALINT. These are for instructions, constants and variables; and are referenced i, c and v, respectively. The storage reference followed by the decimal address within a particular area selects a specific location. Thus i110, c3 and v22 specify locations instruction 110, constant 3 and variable 22.

When used as operands B and C addresses usually refer to constants or variables and are flagged with c or v. If B and C do specify an instruction, they must be preceded by i.

An operand may be complemented by preceding its storage address with a minus sign. Thus -v2 causes the complement of the quantity at variable address 2 to be entered into the operation.

The accumulator address can be specified either by the notation "a", or by leaving the address blank. For example, squaring the accumulator quantity can be expressed either as a x a, or x . This is true for most instructions.

One index register is available in CALINT. The letter "n" in a storage reference specifies the use of the index quantity in the selection of the location of an operand. If the current index quantity is 5, nv12 will specify the variable address 12 + 5 or 17.

4. Coding Format - Normal input to CALINT operation is either from the on-line typewriter or from punched paper tape. The paper tape is prepared by Flexowriter, or it is the Flexcode output of the 160 from typewriter input.

The coding follows the general form

LOCN	B	OPN	C	COMMENTS
4	6	2	6	

The digits represent the maximum number of characters to be used for each code. Each code, except LOCN, is preceded by a tab. A carriage return sets up the next line of coding.

The LOCN is given in the form ixxx, where xxx is the instruction address in which the instruction is to be stored. It is sufficient to specify only the first and in succeeding instructions leave LOCN blank, except for a tab, in which case the next instruction location is selected. The tab sets up the B operand.

B is the first operand reference. Instructions not using B ignore this code.

The OPN code is a one or two character mnemonic code. See the list in section 2. The code must be immediately followed by a tab or carriage return. A space will cause an error in translation.

C is the second operand reference. Similar to B, it may take the form ixxx, vxxx, cxxx or nothing. The implication is that the location may be preceded by either or both - and n, if the operation requires these codes. Spaces within a code are ignored, thus C1 = C 1 = C001 = C01.

The comments are used to annotate the program and are ignored on input to the computer. If the comments are extended beyond one line of coding, pass instructions (ps) must be used to cause the computer to ignore the operation portion of the next line.

All instructions are written in the lower case mode of the typewriter or Flexowriter. Since + is upper case on some Flexowriters, the lower case for this key ,(comma) is the proper code for addition.

If an error in coding is made, the code pair, x carriage return, causes the line to be deleted. Two cautions are advised with this regard. First, if multiplication (x) is specified with a blank C address the x must be followed by a tab. Second the comments line must not end with an x code.

CALINT routines are terminated by coding a carriage return, semicolon and a carriage return.

5. Number Format - Numbers are written in the general form

sign value exponent

The sign may be +, - or blank (positive).

The value consists of significant numbers with any number of leading or trailing zeros. A decimal point must be some where in the number.

The exponent is the power of the base 10, the quantity when multiplied times the value yields the true value of the expression. The exponent is expressed as the letter e, with the sign of the exponent (+, -, or blank) and up to three decimal digits as the exponent value. If the number can be conveniently expressed without exponent, the exponent does not need to be written.

The following numbers are examples of proper expression for CALINT input 1., +1.00234e-896, -3.14, .0000000000034, 7840000000000. The number zero must be written 0.0.

Numbers in CALINT output appear in the form:

decimal point, value, exponent

An example of which is

.359674421 e 005

6. Detailed Operation Description

a. ADDITION

General Form: A = B + C

Examples:

vl	+	v2
	+	v2
v2	+	
	+	
nvl	+	v2
-nvl	+	v2
-nvl	+	-v2
vl	+	-v2
nvl	+	nv2
a	+	v2
v2	+	a
a	+	a

Description

Form in the accumulator the sum of the two specified operands. The result is a normalized floating point number. If either operand is the

accumulator, the initial contents of the accumulator, apply as either or both operands. A test is made to see if one of the operands is zero. In this case, the addition is suppressed to avoid the loss of significance in the floating point addition operation. If the result of the addition is a zero, normalization is suppressed and the result will contain zero with the exponent of the operand with the greater exponent.

b. SUBTRACTION

General Form: A = B - C

Examples: (see addition)

Description:

Form in the accumulator the result of the second operand subtracted from first operand. The remarks on addition apply to subtraction.

c. MULTIPLICATION

General Form: A = B x C

Examples: (see addition)

Description:

Form in the accumulator the algebraic product of the two floating point operands. The product is normalized and rounded to nine decimal digits.

d. DIVISION

General Form: A = B / C

Examples: (see addition)

Description:

Form the result from dividing the first floating point operand by the second floating point operand. This quotient is normalized, rounded to nine decimal digits and placed in the accumulator register. Note that the forms a / vl and vl / a are available, thus it is possible to divide the contents of storage by the contents of the accumulator.

e. TRANSFER ON POSITIVE

General Form: pg C

Example: pg i219

Description:

If the number contained in the accumulator is positive,

obtain the next instruction from the instruction location specified by the C address. If the number contained in the accumulator is negative, obtain the next instruction from the next sequential instruction location.

f. TRANSFER ON NEGATIVE

General Form: ng C

Example: ng i238

Description:

If the number contained in the accumulator is negative, obtain the next instruction from the instruction location specified by the C address. If the number contained in the accumulator is positive, obtain the next instruction from the next sequential instruction location.

g. TRANSFER ON ZERO

General Form: zg C

Example: zg i198

Description:

If the number contained in the accumulator is zero, obtain the next instruction from the instruction location specified by the C address. If the number contained in the accumulator does not equal to zero, obtain the next instruction from the next sequential instruction location.

h. PASS OR DO NOTHING

General Form: ps

Example: c12 ps v39

Description:

Execute the next sequential instruction. The B and C address of the instruction are ignored.

i. SUBROUTINE ENTRY

General Form: B se C

Example: i101 se v3

104 se

105 se -nv6

Description:

The operand specified by the C address (or the accumulator if no C address value is given) is placed in the accumulator and a transfer is made to the instruction at the instruction location specified by B. In making the transfer, the current specification of the next instruction is saved as well as the contents of the index register n. The current contents of index register are unchanged and may be used by the subroutine. The saved contents of n and the location of the next instruction will be restored by the subroutine return instruction. Thus, the subroutine may change the contents of n.

j. SQUARE ROOT

General Form: rt C

Examples:
```
rt  nv45
rt  -v32
rt
rt  -
```

Description:

Obtain the square root of the floating point number specified by the C operand. If a negative number is used as the operand, a halt will occur. Accuracy is to within 2 in the ninth significant digit.

k. SINE

General Form: si C

Description:

Obtain the sine of the number specified by the C operand. The value of C must be express in radians.

l. CONSINE

General Form: co C

Description:

Obtain the cosine of the number specified by the C operand. The value of C must be expressed in radians.

m. TANGENT

General Form: tn C

Description:
Obtain the tangent of the number specified by the C operand. The value of C must be express in radians.

n. ARCTANGENT
General Form: at C
Description:
Obtain the arctangent of the number specified by the C operand.

o. NATURAL LOGARITHM
General Form: lg C
Description:
Obtain the logarithm of the number specified in location C.

p. EXPONENTIAL
General Form: e C
Description:
Raise the value e (2.7 · · ·) to the power given by the value of the C operand.

q. SUBROUTINE RETURN
General Form: sr
Description:
Exit from a subroutine. The values of the next instructions and the index register saved by the subroutine entry are restored. The next instruction executed is immediately following the instruction which caused entry into the current subroutine.

r. INPUT
General Form: in C
Examples: in a
in
in nv39
in c21

Description:

Read one number in from the selected input device and store it in the specified C address. If no C address is given, input will be to the accumulator. The input number must be terminated by a carriage return.

s. OUTPUT FOLLOWED BY TAB

General Form: ot C

Examples:
```
ot   v46
ot   a
ot
ot   nv54
```

Description:

Output one number from the C location on the selected output device. The number is followed by a tab code so the next outputted information will fall in a column to the right of the current number.

t. OUTPUT FOLLOWED BY A CARRIAGE RETURN

General Form: oc C

Description:

Output one number from the location specified by C on the selected output device. The number is followed by a carriage return code, so it will be the rightmost number on the current line.

u. UNCONDITIONAL TRANSFER OR GO

General Form: go C

Example: go i89

Description:

Obtain the next instruction from the location specified by the C address.

v. DATA TRANSFER OR PUT

General Form: B p C

Examples:
```
-vl   p   vl
-a    p   v35
-     p
nv56  p   nv78
-nv34 p   nv65
v67   p   c63
```

Description:

Take the number specified by the B address and place in the location specified by the C address. If a minus sign is used in the B specification, the negative of the number from B will be placed in C. Both B and C may be the accumulator.

w. SET ZERO

General Form: z C

Examples: z v3

z nv56

z

Description:

Replace the previous contents of the C address with the number zero. The C address may be any of the storage locations of CALINT or the accumulator.

x. MACHINE CODE

General Form: mc C

Example: mc 68

Description:

The next instruction to be executed will be from the 160 computer location C and will be 160 machine code. This instruction is used to obtain operations which are not supplied in the CALINT instruction code. It is recommended that a listing of the current version of CALINT be used in deciding which locations are available as the C address. On performing a jump to RNI in machine code, the next instruction after the machine code exit instruction will be executed in CALINT. (The location of RNI in the 3-61 version of CALINT is 1220.)

y. SET INDEX

General Form: sn C

Example: sn 5

sn 0

Description:

Replace the contents of the index register with the numerical value C given in this instruction.

z. INDEX TEST AND INCREMENT

General Form: nt C

Example: nt 15

Description:

The index register is incremented by one and then the result is compared with the value C given in the nt instruction. If the current value of the index register is less than or equal to the value C, the next sequential instruction will be executed. If the value of the index register is greater than the value C, the second instruction following the nt instruction will be executed.

7. Operating Instructions (3-61 version) - CALINT 3-61 tapes include the RS 022 service routine and thus must be loaded starting at 7400.

The memory allocation is such that constants are available from C1 to C127, variables from V1 to V127, and instructions from i0 to i305. The instruction range i0 to i209 may be used without destroying the RS 022 service routine.

If this storage allocation is not satisfactory, it may be changed by changing the contents of locations 23, 24, and 25 which define the beginnnng of each of the mentioned storage areas.

Presently the area of storage from 4633 and up to 7776 is allocated for instructions, constants, and variables. A suggested assignment which will give 600 instructions, 50 constants and 50 variables is:

location	specifies	current	suggested
0023	instructions	6633	5500
0024	constants	4633	5150
0025	variables	5633	4640

It should be noted that each instruction requires two 160 locations while each constant or variable requires four 160 locations.

1. TO LOAD CALINT MASTER TAPE

a. Clear 160 memory by the following steps.

1) Place enter Sweep switch in SWEEP
2) Run.
3) Depress clear button on right side of Z register.
4) Return all switches to neutral position.
5) Depress Clear switch.

b. Load master program tape by the following steps.
1) Turn on reader, turn off punch, place CALINT master tape in reader with seventh level toward the operator.
2) Depress Load-Clear switch to CLEAR, then place it in LOAD position.
3) Set P = 7400.
4) Run. The paper tape will pass through the reader and stop with
P = 4400
A = 4400
Z = 0000

2. TO LOAD PROGRAMS AND CONSTANTS TO BE EXECUTED BY CALINT

a. To load a program prepared on Flexowriter, or by the duplicate mode.
1) Turn on reader, place program tape on reader with seventh level toward the operator.
2) Depress the Load-Clear switch to CLEAR.
3) Set P = 0001.
4) Run. Program and data will be converted to internal form and stored in the CALINT storage. The tape will stop on reaching a semicolon followed by a carriage return as specified in the program preparation section.

b. To load a program and constants from the on-line typewriter.
1) Turn on typewriter and place typewriter in computer control.
2) Depress the Load-Clear switch on the 160 to CLEAR.
3) Set P = 0002.
4) Run. Type in program and data according to program preparation specifications.

c. To load a program and constants from the on-line typewriter and obtain a punched paper tape copy of the program and constants.
1) Turn on typewriter and place typewriter in computer control.

2) Depress the Load-Clear switch on the 160 to CLEAR.
3) Turn on punch.
4) Set P = 0003.
5) Run. Type in program and data according to program preparation specifications. The end of typing must be specified by typing carriage return, semicolon, carriage return.
6) The punch will punch out a trailer to bring the program tape out of the punch well. This tape may be used subsequently to reload the program.

3. TO START A PROGRAM MANUALLY
   a. Depress the Load-Clear switch to CLEAR.
   b. Set A = 0xxx, where xxx is the three decimal digits of the first instruction of the program. Note that only the values 0 to 7 are available at the console and thus it is not possible to manually start at an address with an 8 or 9 as one of its digits.
   c. Run. The program then will start execution.

4. TO START A PROGRAM UNDER TYPEWRITER CONTROL
   a. Depress Load-Clear switch to CLEAR position.
   b. Turn on typewriter and place typewriter under computer control.
   c. Set P = 0002.
   d. Run.
   e. Type si xxx, where xxx is the decimal address of the first instruction to be executed. Type a carriage return.

5. TYPEWRITER CONTROL OPERATIONS
   a. Depress the Load-Clear switch to the CLEAR position.
   b. Turn on typewriter and place typewriter under computer control.
   c. Set P = 0002.
   d. Run.
   e. The input light will come on the typewriter control panel and the operator may type in data and request various services. The services available are:
      1) Start a program as in section 4.
      2) Type in instructions according to input format.

3) Type in constants and variables according to the input format.
4) Start a program and trace the operation of all instructions.
5) Examine the contents of any variable or constant location (PEEK).
6) If program stopped due to a halt instruction, resume operation.

f. The typewriter control functions may be executed any time a start has been made as given above, or any time that a halt instruction has been executed and the address of the halt is typed out. After the halt, the input light will come on indicating a request for more control operations.

The typewriter control operations are selected by typing the one or two characters on a line after the conditions mentioned above (f) occur. The characters with their meaning are as follows:

i INSTRUCTION. The character i specifies an instruction. It must be followed by the address where the instruction is to be stored, then a tab, and then the instruction according to the input format specifications.

v VARIABLE. The character v indicates what follows is a number to be stored in the variable storage area of the CALINT memory.

c CONSTANT. The character c indicates what follows is a number to be stored in the constant storage are of CALINT memory.

siXXX START - followed by a carriage return. Start operation of the program at address xxx.

tiXXX TRACE - followed by a carriage return. Start operation of the program at location XXXX and trace the operation of each instruction.

ri RESUME OPERATION - followed by a carriage return. After a HALT typeout, the computer resumes operation at the next sequential CALINT instruction.

pcXXX
pvXXX } PEEK - followed by a carriage return. Typeout the number in constant or variable location xxx.

6. TO STOP A PROGRAM

a. If there is no output, move the Run-Step switch to NEUTRAL and then restart, or examine contents under typewriter control. All routines in CALINT are self restoring and no damage will be done to the control program.

b. If output is taking place on the typewriter, move the typewriter control lever to NEUTRAL. The computer will complete the present output and stop with a "sel" status. Do step a.

c. If the program is requesting input from the typewriter, do step a.

7. TO TRACE A PROGRAM

a. Under typewriter control, initiate the trace by typing tixxx. This will start the program under Trace control.

b. On initiation of each instruction, the typewriter will type out the contents of the accumulator, or the results of the last instruction. It will then do two carriage returns and type out the address of the instruction about to be initiated as bpXXX.

c. If it is an input instruction, the typewriter will type the address of the input instruction and then request the input. The operator should provide the input. The contents of the accumulator will then be typed out.

d. On an output instruction, the first number typed after the address is the output word and the second number is the contents of the accumulator.

e. To stop the trace, stop the program as in section 6 above and then start under typewriter control and type siXXX. The start command will reset the Trace operation.

8. Sample Program - A short example of a CALINT routine involving the use of indexed constants is presented below.

Problem: Evaluate a polynomial where coefficients A_0, A_1, A_2, A_3 and A_4 are in c10, c11, c12, c13 and c14 and a table of variables X^0, X^1, X^2, X^3 and X^4 is in v6, v7, v8, v9 and v10.

Method: The problem is essentially performed:

c (10 + 0) xv (6 + 0) + c (10 + 1) xv (6 + 1) + . . . + c (10 + 4) xv (6 + 4) =

$a_0 + a_1 X^1 + . . . + a_4 X^4$

Coding:

LOCN	B	OPN	C	COMMENTS
i1		sn	0.	set index register to 0
		z	c1	form sum in c1 set 0
	nc10	x	nv6	$A_k X^k$
		+	c1	$+ \sum A_{k-1} X^{k-1}$
		p	c1	
		nt	4.	increase index; end test
		go	i3	loop
		h		halt

9. CALINT Subroutines - As a supplement to the instructions contained in CALINT, a group of subroutines have been written to increase the variety of problems which may be solved by the user. Included in this package are the following: Square Root, Exponential, Sine, Cosine, Tangent, Arc-tangent, and Natural Logarithm. These subroutines are directly addressable, and a description of their use by the programmer may be found in the CALINT 3-61 write-up. The following is a description of the individual subroutines and the mathematical method used in their solution.

a. Square Root

Space Required: Instructions I1 through I20. **

Accuracy: The use of this method gives eight decimal digits of accuracy.

Mathematical Method: Newton's approximation for the square root.

$Xi+1 = 1/2 (Xi + N/Xi)$, where N is the argument for which the square root is desired, and an initial approximation of $Xi = N$ is made.

Error Stops: HALT I20 with a negative argument.

b. Exponential

Space Required: Instructions I21 through I51. **

Accuracy: The use of this method gives at least seven decimal digits of accuracy.

Mathematical Method: Maclaurin's series for the exponential.

$$\text{Exp } x = \sum_{n=0}^{\infty} \frac{X^n}{n!}$$

Error Stops: HALT I49 with an argument greater than 2300.28.

c. Sine - Cosine

Space Required: Instructions I52 through I113. **

Accuracy: The accuracy of this method is dependent on the magnitude of the argument. For arguments between 0 and 1, the accuracy is eight decimal places; for arguments between 1 and 10, the accuracy is seven decimal places; for arguments between 10 and 100, the accuracy is six decimal places; etc.

Mathematical Method: Method by Hans J. Maehly*

$$\text{SIN } X = X(S_1+X^2(S_2+X^2(S_3+X^2(S_4+X^2(S_5+X^2S_6)))))$$

$S_1 = 1.$

$S_2 = -.166666666$

$S_3 = .833333073\ e\text{-}2$

$S_4 = -.198408338\ e\text{-}3$

$S_5 = .275240118\ e\text{-}5$

$S_6 = -.2386893\ e\text{-}7$

Error Stops: HALT I113 with argument greater than 1.e9.

d. Tangent

Space Required: Instructions I114 through I125. **

Accuracy: This method gives an accuracy which is one place less than that of the Sine.

Mathematical Method: Trigonometric identity.

$$\text{Tangent } X = \frac{\text{Sine } X}{\text{Cosine } X}$$

Note: If Cosine X = 0, 1.e388 is placed in the accumulator.

Error Stops: None.

e. Arctangent

Space Required: Instructions I126 through I173. **

Accuracy: This method gives an accuracy of at least seven decimal digits.

Mathematical Method: Method by Hans J. Maehly*

$$PSI = t\left[d_0+t^2\left(d1+\frac{e1}{t^2+d_2+\frac{e_2}{t^2+d_3}}\right)\right]$$

Arctan z = PSI + PSI k

GROUP I: t = z, PSI k = 0
z less than $\sqrt{2}$ - 1

GROUP II: t = z-1/z+1, PSI k = PI/4
$\sqrt{2}$ - 1 less z less $\sqrt{2}$ + 1

GROUP III: t = 1/-z, PSI k = PI/2
z greater than $\sqrt{2}$ + 1

Error Stops: None.

f. Natural Logarithm

Space Required: Instruction I174 through I204. **

Accuracy: This method gives an accuracy of at least seven decimal digits.

Mathematical Method: Maclaurin's series for the natural logarithm.

$$\text{Ln } x = \sum_{n=1}^{\infty} \frac{(X-1)^n}{n}$$

Error Stops: HALT I204 with argument less than or equal to zero.

---

* Hans J. Maehly, "Approximations for the Control Data 1604", Department of Mathematics, Syracuse University.

** All of the subroutines in this package make use of an additional area of twenty-six permanent constants and seven common erasable constants. The permanent constants, C1 through C26, are part of this package and contain some constants which may be useful to the programmer such as; PI, Ln 10, ... The common erasable constants are C27 through C33.

22 BIT ARITHMETIC

The 22 bit arithmetic subroutines include Add, Multiply, and Divide. Either positive or negative numbers may be used. The subroutines are so coded that they can be stored anywhere in the memory, with the exception of the first 64 words.

Addresses 0002, 0003, and 0004 contain the starting addresses of the Add, Multiply, and Divide subroutines, respectively. Addresses 0010, 0011, 0020, 0021, 0030, 0031, 0040, and 0044 are used by the different subroutines and must be reserved. All results appear in 0030, 0031. Address 0007 is used as the exit address of each of the subroutines.

Two 12 bit words are used for each 22 bit quantity; the left-most bit in each word is not used. The most significant bit of the 22 bits is a sign bit.

The annotated subroutines (below) are entered at addresses 0100-0446 but (as mentioned above) could be located almost anywhere else in memory. With the routines located as shown, these address assignments are fixed:

0002	0175	Entrance for Add Subroutine
0003	0104	Entrance for Multiply Subroutine.
0004	0337	Entrance for Divide Subroutine

If the subroutines are stored elsewhere, (only) these three addresses need be changed.

ADD SUBROUTINE

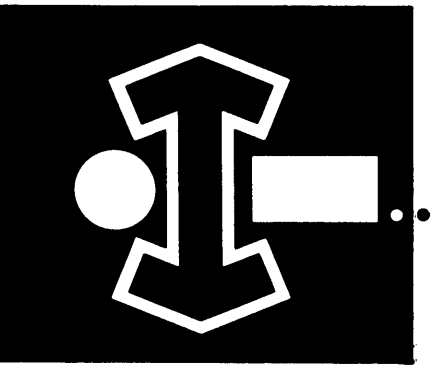
The Add subroutine adds the 22 bit quantity in 0010, 0011 to that in 0020, 0021. The result appears in 0030, 0031. The average execution time is 185 microseconds.

MULTIPLY SUBROUTINE

The Multiply subroutine multiplies the 12 bit quantity in 0040 to the 12 bit quantity in 0044. The left-most bit is a sign bit. The (22 bit) result appears in 0030, 0031. The average execution time is 1.8 milliseconds.

DIVIDE SUBROUTINE

The Divide subroutine divides the 22 bit quantity in 0010, 0011 by that in 0020, 0021. The result appears in 0030, 0031. The average executior time is 1.8 milliseconds.



CONTROL DATA CORPORATION

501 PARK AVENUE • MINNEAPOLIS 15, MINNESOTA