CONTROL DATA
1604-A COMPUTER
PROGRAMMING MANUAL

1604-A INSTRUCTIONS

			Page				Page
00	ZRO	(not used)		40	SST	Selective Set	2-33
01	ARS	A Right Shift	2-13	41	SCL	Selective Clear	2-34
02	QRS	Q Right Shift	2-13	42	SCM	Selective Complement	2-33
03	LRS	AQ Right Shift	2-13	43	SSU	Selective Substitute	2-35
04	ENQ	Enter Q	2-25	44	LDL	Load Logical	2-35
05	ALS	A Left Shift	2-13	45	ADL	Add Logical	2-35
06	QLS	Q Left Shift	2-14	46	SBL	Subtract Logical	2-35
07	LLS	AQ Left Shift	2-14	47	STL	Store Logical	2-35
10	ENA	Enter A	2-25	50	ENI	Enter Index	2-26
11	INA	Increase A	2-25	51	INI	Increase Index	2-26
12	LDA	Load A	2-10	52	LIU	Load Index, U	2-12
13	LAC	Load A, Complement	2-10	53	LIL	Load Index, L	2-12
14	ADD	Add	2-17	54	ISK	Index Skip	2-16
15	SUB	Subtract	2-17	55	IJP	Index Jump	2-16
16	LDQ	Load Q	2-10	56	SIU	Store Index, U	2-12
17	LQC	Load Q, Complement	2-10	57	SIL	Store Index, L	2-12
20	STA	Store A	2-11	60	SAU	Substitute Address, U	2-15
21	STQ	Store Q	2-11	61	SAL	Substitute Address, L	2-15
22	AJP	A Jump	2-27, 30	62	INT	Input Transfer	2-40
23	QJP	Q Jump	2-28, 31	63	OUT	Output Transfer	2-40
24	MUI	Multiply Integer	2-18	64	EQS	Equality Search	2-36
25	DVI	Divide Integer	2-19	65	THS	Threshold Search	2-36
26	MUF	Multiply Fractional	2-20	66	MEQ	Masked Equality	2-37
27	DVF	Divide Fractional	2-20	67	MTH	Masked Threshold	2-37
30	FAD	Floating Add	2-20	70	RAD	Replace Add	2-38
31	FSB	Floating Subtract	2-21	71	RSB	Replace Subtract	2-38
32	FMU	Floating Multiply	2-22	72	RAO	Replace Add One	2-38
33	FDV	Floating Divide	2-23	73	RSO	Replace Subtract	2-39
34	SCA	Scale A	2-24	74	EXF	External Function	3-3
35	SCQ	Scale AQ	2-24	75	SLJ	Selective Jump	2-29, 31
36	SSK	Storage Skip	2-32	76	SLS	Selective Stop	2-29, 31
37	SSH	Storage Shift	2-32	77	SEV	(not used)	

CONTROL DATA 1604-A COMPUTER

PROGRAMMING MANUAL

245

PRINTED IN THE UNITED STATES OF AMERICA

RECORD OF CHANGE NOTICES

C. N. NO.	DATE ORIGINATED	DATE ENTERED	INITIALS	REMARKS

CONTENTS

Chapter 1 - Description

Chapter 2 - Description of Instructions

CONTENTS (Cont'd)

Chapter 2 - Description of Instructions (Cont'd)

Chapter 3 - Input/Output

CONTENTS (Cont'd)

Chapter 4 - Operation

Glossary

Appendix Section

FIGURES

Chapter 1 - Description

Chapter 3 - Input/Output

Chapter 4 - Operation

TABLES

Chapter 1 - Description

Chapter 3 - Input/Output

Chapter 4 - Operation


CONTROL DATA
1604-A

CHAPTER 1
DESCRIPTION

The CONTROL DATA* 1604-A is a stored-program, general-purpose digital computer with a large storage capacity, exceedingly fast computation and transfer speeds, and special provisions for input/output communication. The 1604-A is designed to handle large-volume data processing and to solve large-scale scientific problems. The compact equipment, constructed from solid-state components throughout, is suitable for use in a semi-permanent office environment.

1604-A CHARACTERISTICS

Stored-program general-purpose digital computer

Parallel mode of operation

48-bit word, 2 instructions per word

Single address logic

Operation code	6 bits
Designator	3 bits
Base Execution Address	15 bits

Six 15-bit index registers

Indirect addressing

Magnetic core storage
- 32,768 48-bit words

Two independent 16,384 word banks alternately phased

4.8 μsec effective cycle time (representative program)

6.4 μsec total cycle time

Input/output
- Transmission of 48-bit words
- Three separate buffer input channels
- Three separate buffer output channels
- High-speed transfer channel (4.8 μsec per word)

Program interrupt

Console, includes:
- Photo-electric paper tape reader
- Paper tape punch
- Electric typewriter
- Register contents displayed in octal

Flexible instructions
- Fixed-point arithmetic (integer and fractional)
- Floating-point arithmetic
- Logical and masking operations
- Indexing
- Storage searching

Binary arithmetic
- Parallel addition in 1.2 μsec without access
- Modulus $2^{48} - 1$ (one's complement)

Real-time clock

Completely solid-state
- Diode logic
- Transistor amplifiers

---

*Registered trademark of Control Data Corporation

LOGICAL DESCRIPTION

The 1604-A performs calculations and processes data in a parallel binary mode through the step-by-step execution of individual instructions which are stored internally along with the data.

Functionally, the computer may be divided into four major sections. Storage provides internal storage for data and instructions; Control coordinates and sequences all operations for executing an instruction by obtaining the instruction from storage and translating it into commands for the other sections; Arithmetic performs the arithmetic and logical operations required for executing instructions; and Input/Output provides communication between the computer and the external equipment.

The registers in the computer are identified by letters (table 1-1). The arithmetic properties of the registers are detailed in table 1-2. The operational registers usually hold the end result of an operation. Their contents are displayed on the console and may be changed manually.

TABLE 1-1. REGISTERS OF THE COMPUTER

Register	Function
A*	Accumulator
Q*	Auxiliary Arithmetic
B^1 through B^6*	Index registers (six)
P*	Program Address

Register	Function
U^1*	Program Control
U^2	Auxiliary Program Control
R	Address Buffer
CCR, CR	Buffer Control
X	Exchange

*Operational Registers

STORAGE SECTION

The magnetic core storage section of the 1604-A computer provides high-speed, random access storage for 32,768 words. It consists of two independent storage units each with a capacity of 16,384 words. These units operate together during the execution of a stored program and thus are considered as one 32,768 word storage system.

A word is 48 bits in length and is used in two ways: as two 24-bit instructions or as a 48-bit operand (data word). The location of each word in storage is identified by an assigned number or address. When a word is taken (read) from or entered (written) into storage, a reference is made to the storage address which holds the word. All odd storage addresses are located in one storage unit, all even addresses in the other.

The cycle time, or time for a complete storage reference, is 6.4 μsec. Since the storage cycles of the two sections overlap one another in the execution of a program, the average effective cycle time for random addresses is about 4.8 μsec.

CONTROL SECTION

The control section directs the operations required to execute instructions and to initiate the exchange of data with external equipment. It also establishes the timing relationships needed to perform the operations in the proper sequence.

The control section acquires a program word from storage, interprets it and sends the necessary commands to other sections. A program word is a pair of 24-bit instructions which together occupy one storage location as a 48-bit word. The higher-order 24 bits are the upper instruction; the remaining 24 bits, the lower instruction.

Instruction Format

f (6 bits)	b (3 bits)	m, y, or k (15 bits)
Operation Code	Index Designator	Base Execution Address

Each of the 62 instructions has a unique 6-bit operation code which specifies the operation to be performed.

The X (Exchange) register is the communication center of the computer. All internal transmissions between the arithmetic section and the rest of the computer are made through X.

INPUT/OUTPUT SECTION

The input/output section of the computer handles the flow of information to and from the computer. Prior to executing a program, the data and instructions which comprise the program (input) are loaded into computer storage. After computation is completed, the results (output) are transmitted from storage to an external equipment. All information is transmitted by separate input/output registers in the form of 48-bit words.

The computer communicates with external equipment through six independent buffer channels which provide for the normal exchange of data (figure 1-1).

Input:	Channel 1	Output:	Channel 2
	Channel 3		Channel 4
	Channel 5		Channel 6

The input and output buffer channels are paired, channels 1 and 2, channels 3 and 4, and channels 5 and 6. Every external equipment is connected to one of these channels. All six buffer channels may concurrently transmit information. However, only one external equipment can use any one buffer channel at any given instant.

In the 1604-A computer, input/output operations are independent of the main computer program. When data is to be transmitted, the main computer program initiates an automatic cycle which buffers data to or from computer storage. The main computer program then continues while the actual buffering of data is carried out independently and automatically.

This process of asynchronous input/output operations is called buffering. Buffer transmissions employ independent access to computer storage. Computation continues while the external equipment is loading or unloading information from computer storage at a rate dictated by the external equipment.

PROGRAM COMPATIBILITY

All programs written for the 1604 computer (except those using programmed interrupt lockout) can be run on the 1604-A computer by switching in an optional circuit. A red background light on the leftmost digit of the P register display indicates that the optional circuit is switched in for running 1604 programs.

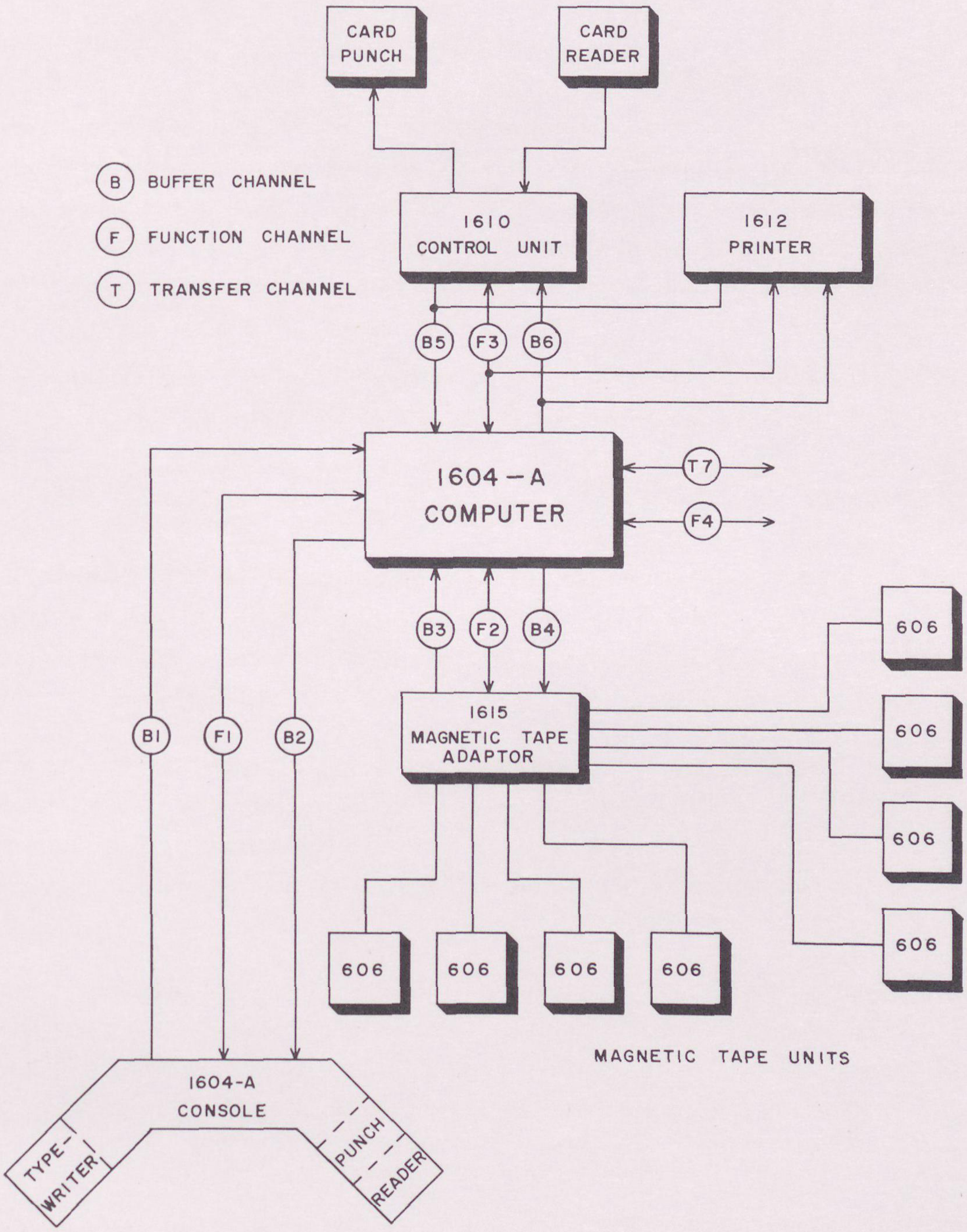


Figure 1-1. Typical 1604-A System

CHAPTER 2

DESCRIPTION OF INSTRUCTIONS

WORD FORMAT

A computer word consists of 48 bits and may be interpreted as one 48-bit data word or two 24-bit instructions. Each instruction is composed of three parts or codes: operation code, index designator, and execution address. The higher-order 24 bits of the word are called the upper instruction and the lower-order 24 bits are called the lower instruction.

bit 47		bit 24
Operation (Function) Code f	Index Designator b	Execution Address m, y, k
6 bits	3 bits	15 bits

Code	Range	
Operation f	01 - 76_8	Specifies the operation to be performed. A 00 or 77 code is interpreted as a fault, which stops computer operation.
Index Designator b	0 1-6 7	No address modification Relative address modification Specifies the index designator whose contents are to be added to the execution address (refer to jump and stop instructions for exceptions). Indirect addressing
Execution Address m, y, k	00000 through 77777_8	Used in one of three ways: 1) as a shift count, k 2) as an operand address, m 3) as an operand, y

EXECUTION ADDRESS

The base execution address may be used as : (1) a shift count, k; (2) an operand, y; (3) an address of an operand, m, in storage. The execution address may also be modified or unmodified depending on the index designator. If unmodified, the address is represented by the lower-case symbol k, y, or m; if the address is modified the symbols are capitalized. The following examples point out the relationship between the unmodified and modified execution address.

The modified shift count K is represented by:

1) $K = k + (B^b)$ where:

- K = modified shift count
- k = unmodified shift count (execution address)
- (B^b) = contents of index register b.

If the index designator = 0, then K = k.

The modified operand Y is represented by:

2) $Y = y + (B^b)$ where:

- Y = modified operand
- y = unmodified operand (execution address)
- (B^b) = contents of index register b.

If the index designator = 0, then Y = y.

The modified operand address M is represented by:

3) $M = m + (B^b)$ where:

- M = modified address of operand
- m = unmodified address of operand (execution address)
- (B^b) = contents of index register b.

If the index designator = 0, then M = m. Note that (3) is the only case in which the execution address is interpreted as an address of an operand.

ADDRESS MODIFICATION

The three possible modes of address modification are identified by the index designators as follows:

1) b = 0 No Address Modification. In this mode the execution address is interpreted without modification; nothing is added to or subtracted from it. (Direct addressing.)

2) b = 1-6 Relative Address Modification. In this mode the execution address is modified and is equal to the initial execution address plus the contents of the designated index register. One's complement arithmetic is used in determining the modified execution address.

3) b = 7 Indirect Addressing. In this mode the base execution address specifies the location of the operand address rather than the operand. The 48-bit word is read from storage and the lower-order 18 bits of the word are interpreted as the b designator (3 bits) and execution address (15 bits) of the present instruction. The new index designator may refer to any one of the three modes.

Examples:

1) No Address Modification

```
 f   b     m
LDA  0  address
```

This instruction is interpreted as load accumulator from the storage location designated by the sum of the execution address and the contents of the specified index register, B^b. Since b = 0, no index register is designated and m specifies the storage location whose contents are loaded into A.

2) Relative Address Modification

```
 f   b     m
LDA  6  address
```

$(B^6) = 00001_8$

In this example, the accumulator is loaded from the storage location designated by the execution address plus the contents of index register 6. Therefore, the contents of the storage location named by the execution address plus 00001_8 is loaded into the accumulator. $M = m + (B^b)$.

3) Indirect Addressing

```
                  f    b    m
Current
Instruction   = LDA    7  00100

                        (00100) = FAD 0 00300     FMU 6 00200
```

$(B^6) = 00001_8$

When the b designator of the current instruction is 7, the mode is indirect addressing. The lower 18 bits of the contents of the storage location designated by the execution address, 00100, are read from storage into the U^1 register where they are interpreted as the index designator and execution address of the current instruction.

The index designator is inspected again and because it is not 0 or 7 the relative address mode exists. (Note that the new index designator could reference any one of the three modes of address modification.) The execution address, 00200, plus the contents of B^6, 00001_8 specify the storage location whose contents will be loaded into the accumulator. $M = 00200_8 + (00001_8) = 00201_8$

EXECUTION OF A PAIR OF INSTRUCTIONS

Example:

	f b m	f b m
(00300) =	LDA 0 00310	ADD 1 00210
(00301) =	STA 0 00400	SLS 0 00301

$(B^1) = 00101_8$

The P register holds address 00300 (an even lowest bit indicates the address of the program step is in the even storage unit). The storage reference is initiated; the 48-bit word is read from address 00300 and entered into U^1. Computer operation is now dependent upon the interpretation of the 24-bit instruction in the upper half of U^1.

The operation code, LDA, and the index designator, 0, are translated. The function of the upper instruction, LDA, is to load the A register with the contents of the designated storage location. Because the index designator is 0, the execution address is not modified. The translation of the operation code initiates the sequence of the commands which execute the instruction and the operand in address 00310 is loaded into A.

The lower instruction in U^1 is transferred to U^1 upper and translated. The ADD instruction causes the quantity in storage location M to be added to the contents of the A register. Since the index designator is not 0 or 7, the contents of the index register are added to the execution address to form M. $M = m + (B^b) = 00210_8 + 00101_8 = 00311_8$. The contents of storage address 00311 are added to the contents of the A register completing the instruction. The contents of the P register are increased by one and the pair of instructions at address 00301 is read from storage and executed.

INSTRUCTIONS

The 62 computer instructions are described on the following pages, (EXF instructions are discussed in detail in chapter three). The title line contains the numeric code, the mnemonic code and format, name, and average execution time of the instruction. Abbreviations and symbols are defined as follows:

A	Accumulator
A_n	The binary digit in position n of the A register
$\rightarrow$	Transmit to
b	Index designator
B^b	Designated index register
Exit (Full)	Proceed to upper instruction of next program step
Half exit	Proceed to lower instruction of same program step
j	The condition designator for jump and stop instructions
k	Unmodified shift count
K	Modified shift count. $K = k + (B^b)$
LA	Lower address - execution address portion of lower instruction of a program step
m	Unmodified operand address
M	Modified operand address. $M = m + (B^b)$
( )	Contents of a register or storage location
( )'	One's complement contents of a register or storage location
( )f	Final contents of a register or storage location
( )i	Initial contents of a register or storage location
Q	Auxiliary arithmetic register
UA	Upper address
X	Exchange register
y	Unmodified operand
Y	Modified operand. $Y = y + (B^b)$

INSTRUCTION EXECUTION TIME

The time needed to execute an instruction varies from application to application because of the following factors.

If the instruction occupies the upper position in an instruction word, the time needed to read the word from storage must be considered.

If consecutive storage references are made to the same storage unit (even-even or odd-odd) the read access time from storage will be maximized.

If indirect addressing is specified, at least one additional reference will be needed to complete the instruction. (The new index designator may itself specify indirect addressing.)

If buffer operations are using storage, an instruction must wait until storage is released.

If a storage reference is made at the end of the preceding instruction, execution of the next instruction may be delayed.

The instruction execution times listed on the following pages were compiled by averaging the times for a long list of the same instructions. The list was arranged for typical values of the factors.

ORDER OF INSTRUCTIONS

Numeric Code	Mnemonic Code	Name	Timing*
DATA TRANSMISSION			
12	LDA	LOAD A	7.2
13	LAC	LOAD A COMPLEMENT	7.2
16	LDQ	LOAD Q	7.2
17	LQC	LOAD Q COMPLEMENT	7.2
20	STA	STORE A	7.2
21	STQ	STORE Q	7.2
52	LIU	LOAD INDEX (UPPER)	7.2
53	LIL	LOAD INDEX (LOWER)	7.2
56	SIU	STORE INDEX (UPPER)	7.2
57	SIL	STORE INDEX (LOWER)	7.2
SHIFTING			
01	ARS	A RIGHT SHIFT	2.8 + .4s**
02	QRS	Q RIGHT SHIFT	2.8 + .4s**
03	LRS	AQ RIGHT SHIFT	2.8 + .4s**
05	ALS	A LEFT SHIFT	2.8 + .4s**
06	QLS	Q LEFT SHIFT	2.8 + .4s**
07	LLS	AQ LEFT SHIFT	2.8 + .4s**
ADDRESS MODIFICATION			
60	SAU	SUBSTITUTE ADDRESS (UPPER)	7.2
61	SAL	SUBSTITUTE ADDRESS (LOWER)	7.2
54	ISK	INDEX SKIP	5.6
55	IJP	INDEX JUMP	4.4

*Timing is average execution time in μsec

**s = Number of places shifted

ARITHMETIC (Fixed)			
14	ADD	ADD	7.2
15	SUB	SUBTRACT	7.2
24	MUI	MULTIPLY INTEGER	25.2 + .8n*
25	DVI	DIVIDE INTEGER	65.2
26	MUF	MULTIPLY FRACTIONAL	25.2 + .8n*
27	DVF	DIVIDE FRACTIONAL	65.2
ARITHMETIC (Floating)			
30	FAD	FLOATING ADD	18.8
31	FSB	FLOATING SUBTRACT	18.8
32	FMU	FLOATING MULTIPLY	36.0
33	FDV	FLOATING DIVIDE	56.0
34	SCA	SCALE A	2.8 + .4s**
35	SCQ	SCALE AQ	2.8 + .4s**
NO ADDRESS			
04	ENQ	ENTER Q	3.0
10	ENA	ENTER A	
11	INA	INCREASE A	
50	ENI	ENTER INDEX	
51	INI	INCREASE INDEX	
JUMPS AND STOPS (Normal)			
22	AJP	A JUMP	7.2
23	QJP	Q JUMP	
75	SLJ	SELECTIVE JUMP	
76	SLS	SELECTIVE STOP	
JUMPS AND STOPS (Return)			
22	AJP	A JUMP	7.2
23	QJP	Q JUMP	
75	SLJ	SELECTIVE JUMP	
76	SLS	SELECTIVE STOP	

---

*n = Number of ones in multiplier

**s = Number of positions shifted

STORAGE TEST			
36	SSK	STORAGE SKIP	8.8
37	SSH	STORAGE SHIFT	12.8
LOGICAL			
40	SST	SELECTIVE SET	7.2
42	SCM	SELECTIVE COMPLEMENT	
41	SCL	SELECTIVE CLEAR	
43	SSU	SELECTIVE SUBSTITUTE	7.4
44	LDL	LOAD LOGICAL	
45	ADL	ADD LOGICAL	
46	SBL	SUBTRACT LOGICAL	
47	STL	STORE LOGICAL	7.2
STORAGE SEARCH			
64	EQS	EQUALITY SEARCH	4.0 + 3.6r*
65	THS	THRESHOLD SEARCH	
66	MEQ	MASKED EQUALITY	
67	MTH	MASKED THRESHOLD	
REPLACE			
70	RAD	REPLACE ADD	13.2
71	RSB	REPLACE SUBTRACT	
72	RAO	REPLACE ADD ONE	
73	RSO	REPLACE SUBTRACT ONE	
TRANSFER			
62	INT	INPUT TRANSFER	4.0 + 4.8r*
63	OUT	OUTPUT TRANSFER	

* r = Number of repeated executions

1) Relative addressing does not take place during LIU, LIL, SIU or SIL instructions. Only direct and indirect addressing are recognized.

2) All modes of address modification apply to the remaining data transmission instructions.

3) During the execution of data transmission instructions, one storage reference is made. If indirect addressing is designated, at least two storage references are made.

LDA bm 12 Load A 7.2 μsec

Replaces the contents of A with a 48-bit operand contained in storage location M. The initial contents of A are changed during execution; the contents of M remain unchanged.

LAC bm 13 Load A Complement 7.2 μsec

Replaces the contents of A with the complement of a 48-bit operand contained in storage location M. The initial contents of A are changed during execution; the contents of M remain unchanged.

LDQ bm 16 Load Q 7.2 μsec

Replaces the contents of Q with a 48-bit operand contained in storage location M. The initial contents of Q are changed during execution; the contents of address M remain unchanged.

LQC bm 17 Load Q Complement 7.2 μsec

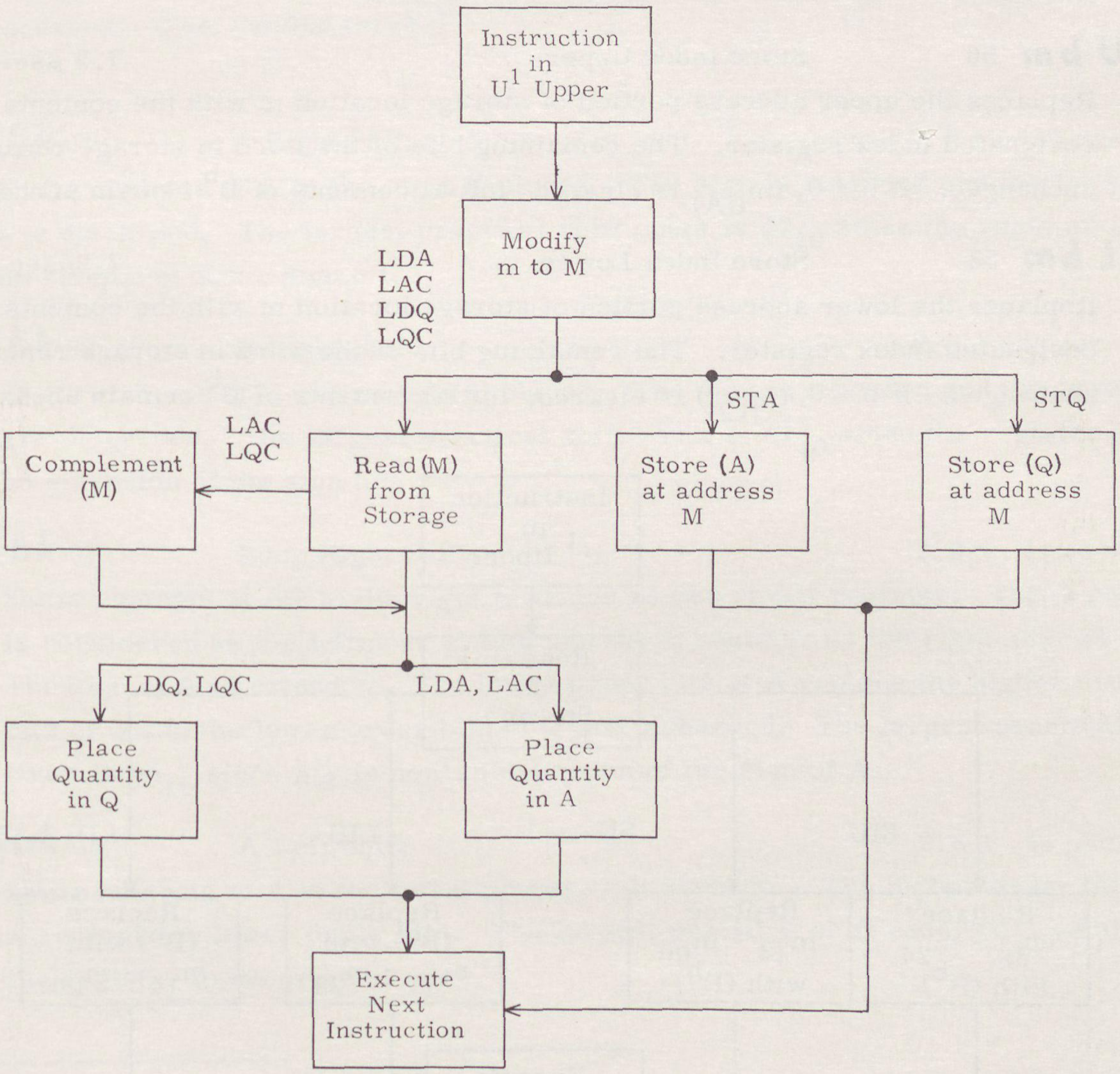
Replaces the contents of Q with the complement of a 48-bit operand contained in storage location M. The initial contents of Q are changed during execution; the contents of address M remain unchanged.

STA bm 20 Store A 7.2 μsec

Replaces the contents of the designated storage location, M, with the contents of A. The initial contents of A remain unchanged.

STQ bm 21 Store Q 7.2 μsec

Replaces the contents of the designated storage location, M, with the contents of Q. The initial contents of Q remain unchanged.



LDA, LAC, LDQ, LQC, STA, and STQ

LIU bm 52 Load Index Upper 7.2 μsec

Replaces the contents of the designated index register with the upper address portion of storage location m. If b = 0 this instruction becomes a pass (do nothing) instruction. Initial contents of m remain unchanged.

LIL bm 53 Load Index Lower 7.2 μsec

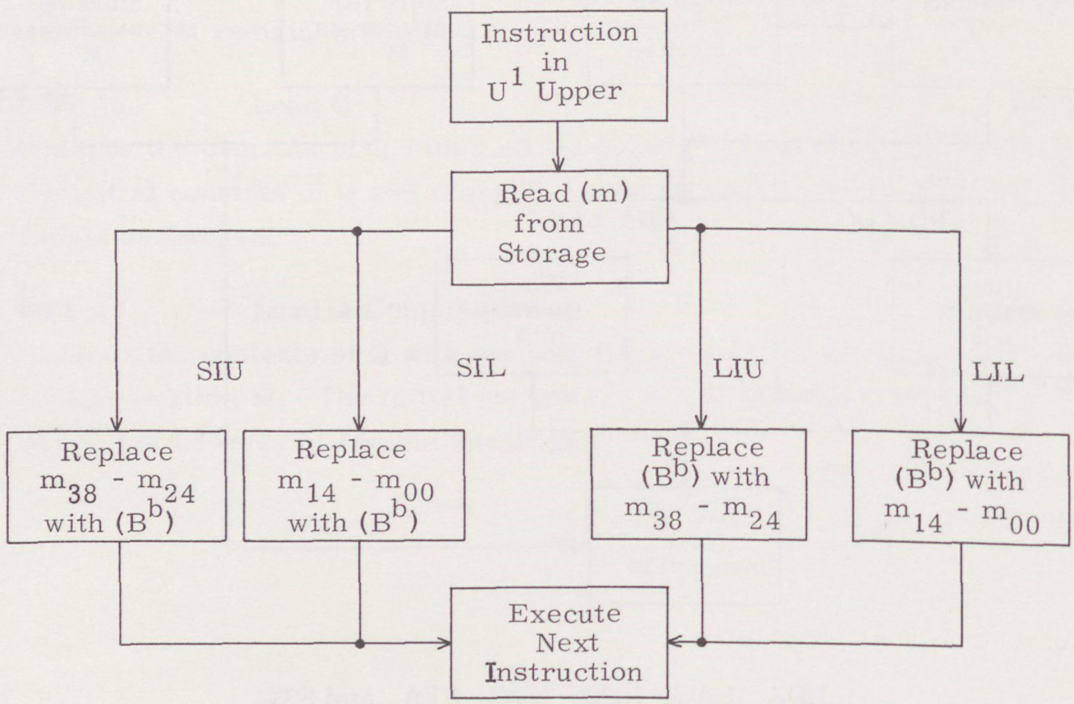
Replaces the contents of the designated index register with the lower address portion of storage location m. If b = 0 this instruction becomes a pass (do nothing) instruction. Initial contents of m remain unchanged.

SIU bm 56 Store Index Upper 7.2 μsec

Replaces the upper address portion of storage location m with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If b = 0, (m_{UA}) is cleared. Initial contents of B^b remain unchanged.

SIL bm 58 Store Index Lower 7.2μsec

Replaces the lower address portion of storage location m with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If b = 0, (m_{LA}) is cleared. Initial contents of B^b remain unchanged.



LIU, LIL, SIU, and SIL

SHIFTING

1) All modes of address modification apply to these instructions.

2) If the modified shift count, K, is greater than 127_{10}, a fault indicator is set. Regardless of the magnitude of count, however, the required number of shifts is executed. (K is reduced by one count for each shift executed and when K = 0, shifting stops.)

3) Shifting must be completed before an input/output or interrupt request can be processed. (See chapter three.)

ARSbk 01 A Right Shift 2.8 + .4s* μsec

Shifts contents of A to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is 47_{10} since the register is now an extension of the sign bit.

QRSbk 02 Q Right Shift 2.8 + .4s μsec

Shifts contents of Q to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is 47_{10} since the register is now an extension of the sign bit.

LRSbk 03 Long Right Shift 2.8 + .4s μsec

Shifts contents of AQ to the right K places as one 96-bit register. The A register is considered as the leftmost 48 bits and the Q register as the rightmost 48 bits. The sign of A is extended. The lower order bits of A replace the higher order bits of Q and the lower order bits of Q are discarded. The largest practical shift count is 95_{10} since AQ is now an extension of the sign of A.

ALSbk 05 A Left Shift 2.8 + .4s μsec

Shifts contents of A to the left K places, left circular. The higher order bits of A replace the lower order bits. The largest practical shift count 48_{10} returns the register to its original state.

---

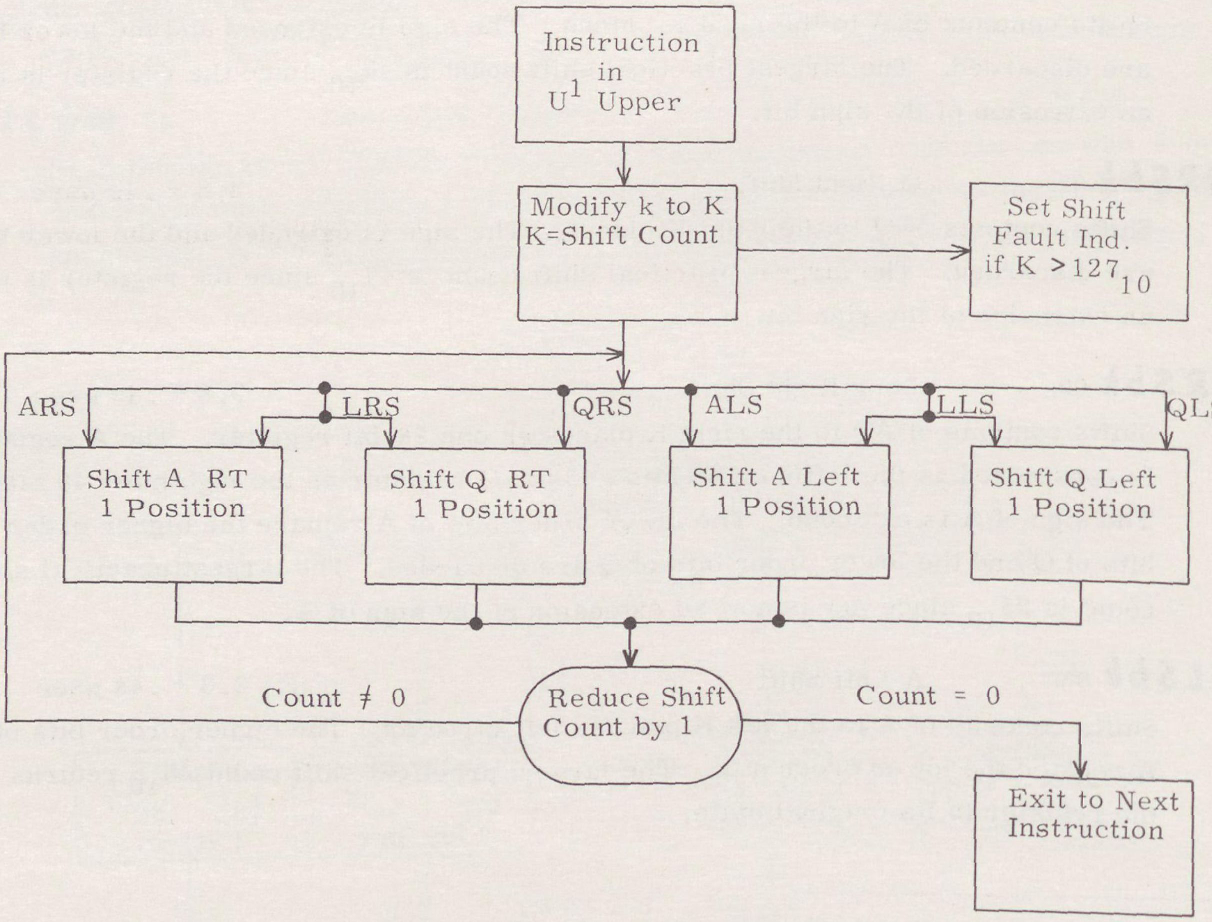
*s = Number of positions shifted

QLSbk 06 Q Left Shift $2.8 + .4s\ \mu sec$

Shifts contents of Q to the left K places, left circular. The higher order bits of Q replace the lower order bits. The largest practical shift count 48_{10} returns the register to its original state.

LLSbk 07 Long Left Shift $2.8 + .4s\ \mu sec$

Shifts contents of AQ to the left K places, left circular, as one 96-bit register. The higher order bits of A replace the lower order bits of Q and the higher order bits of Q replace the lower order bits of A. The largest practical shift count 96_{10} returns AQ to its original state.



Shift Instructions

ADDRESS MODIFICATION

1) All modes of address modification apply to SAU and SAL instructions.

2) Relative addressing cannot be used for ISK or IJP instructions. Only direct or indirect addressing are used.

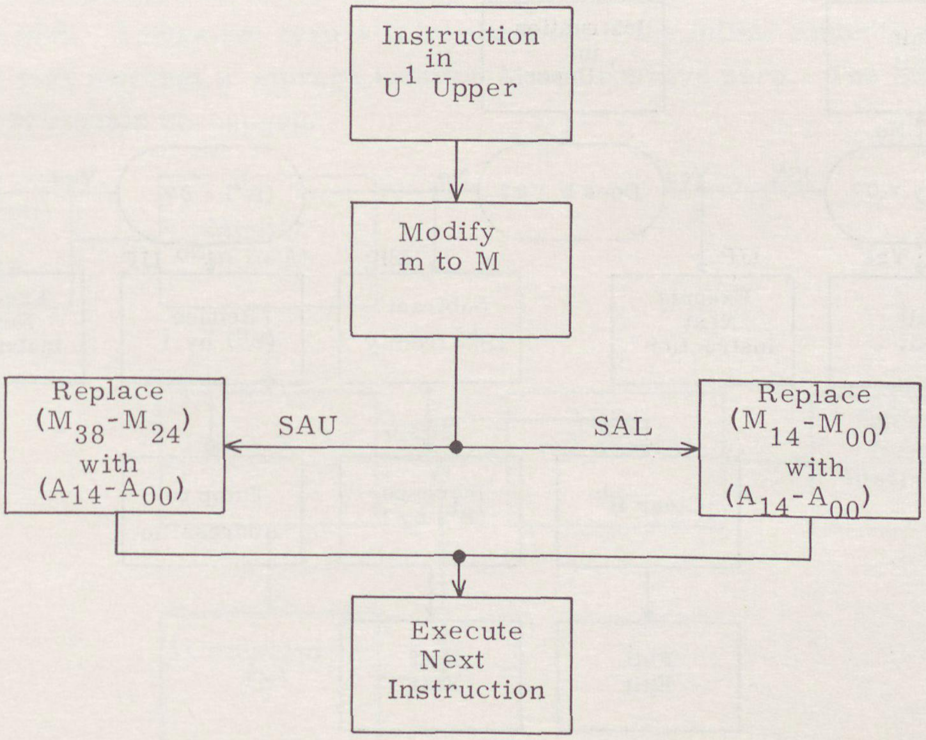
3) During execution of ISK and IJP instructions, no storage reference is made unless indirect addressing is specified which requires at least one reference. For SAU and SAL instructions, one reference is always made. If indirect addressing is designated, at least one additional reference will be needed to complete the instruction.

SAU b m 60 Substitute Address Upper 7.2 μsec

Replaces the upper address portion of M with the lower-order 15 bits of A. Remaining bits of M are not modified and the initial contents of A are unchanged.

SAL b m 61 Substitute Address Lower 7.2 μsec

Replaces the lower address portion of M with the lower-order 15 bits of A. Remaining bits of M are not modified and the initial contents of A are unchanged.



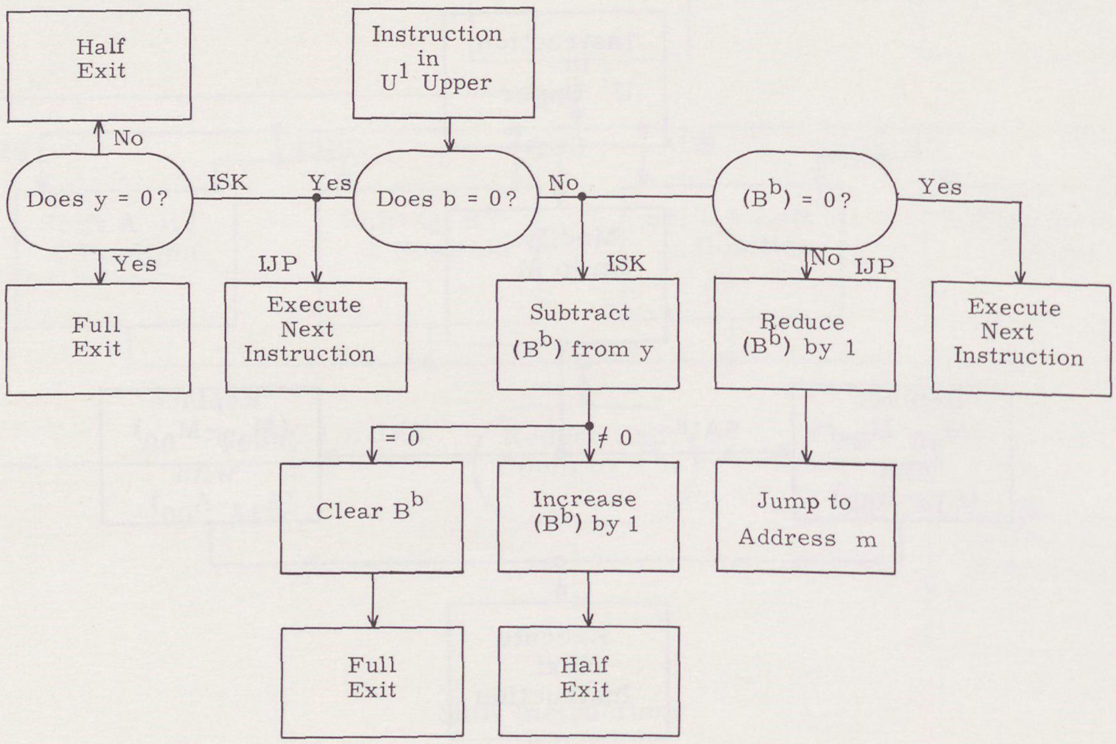
SAU and SAL

ISK by 54 Index Skip 7.2 μsec

Compares (B^b) with y. If the two quantities are equal, B^b is cleared and a full exit is performed. If the quantities are unequal, (B^b) is increased one count in the R register and a half exit is performed. Because the R register is a two's complement subtractive counter, it is possible to count through negative zero and positive zero. (See appendix.) If b = 0 and y ≠ 0, a half exit is taken. If b = 0 and y = 0, a full exit is taken. ISK is usually restricted to the upper instruction. If used as a lower instruction it will half exit upon itself until the full exit condition is satisfied; if b = 0 and y ≠ 0, the condition will never be satisfied.

IJP bm 55 Index Jump 7.2 μsec

Examines (B^b). If this quantity is not zero, the quantity is reduced one count and a jump is executed to address m. The counting operation is performed in the R register but negative zero is not generated because IJP terminates at positive zero. (See appendix.) The index jump can be used in the upper or lower instruction without reservation; it executes a normal jump upon satisfaction of the jump condition.



ISK and IJP

ARITHMETIC

Fixed

1) All modes of address modification apply to these instructions.

2) One storage reference is made for each instruction unless indirect addressing is designated. In this case, at least two references are made.

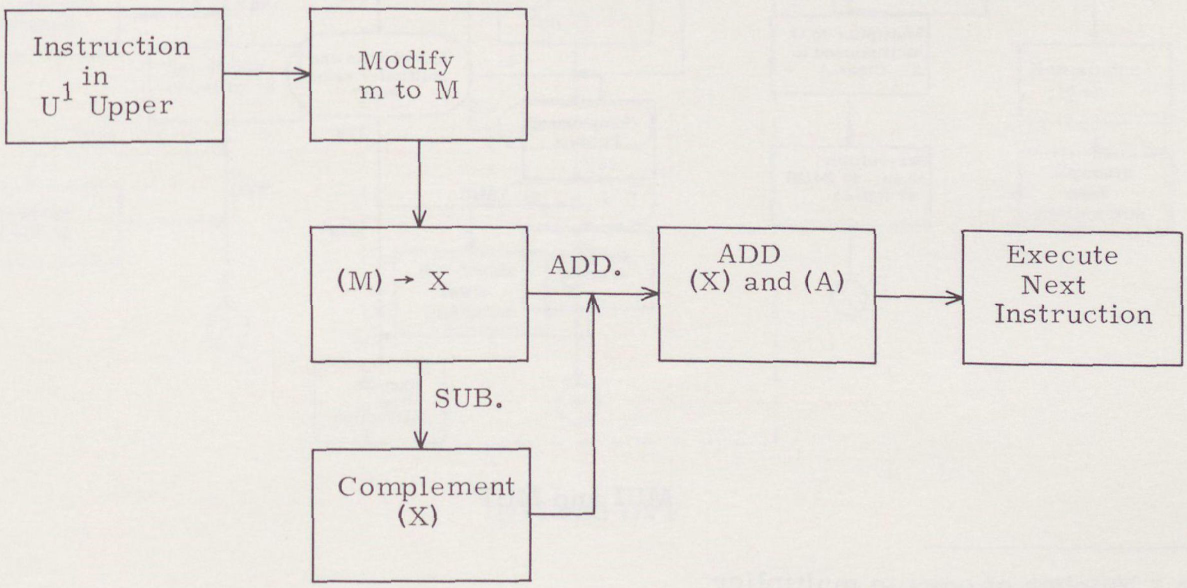
3) If the capacity of the A register $\pm(2^{47}-1)$ is exceeded during the execution of the instructions an arithmetic overflow fault is produced. When executing the DVI or DVF instructions, if the result exceeds the capacity of the Q register $\pm(2^{47}-1)$ a divide fault is produced. (Refer to appendix.)

ADD bm 14 Add 7.2 μsec

Adds a 48-bit operand obtained from storage location M to contents of A. A negative zero may be produced by this instruction if (A) and (M) are initially negative zero. The contents of storage address M remain unchanged.

SUB bm 15 Subtract 7.2 μsec

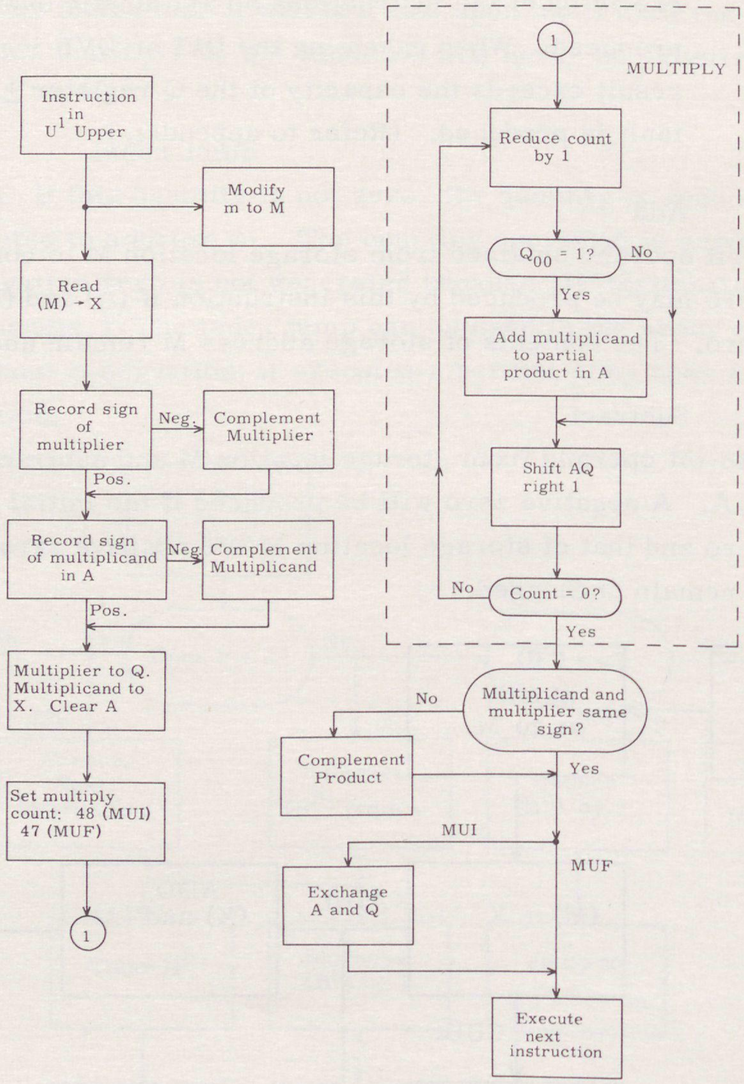
Obtains a 48-bit operand from storage location M and subtracts it from the initial contents of A. A negative zero will be produced if the initial contents of A are negative zero and that of storage location M are positive zero. The contents of address M remain unchanged.



ADD and SUB

MUIbm 24 Multiply Integer 25.2 + .8n* μsec

Forms a 96-bit product from two 48-bit operands. The multiplier must be loaded into A prior to execution of the instruction. The execution address specifies the storage location of the multiplicand. The product is contained in QA as a 96-bit quantity. The operands are considered as integers and therefore the binary point is assumed to be at the lower order (right hand) end of the A register.



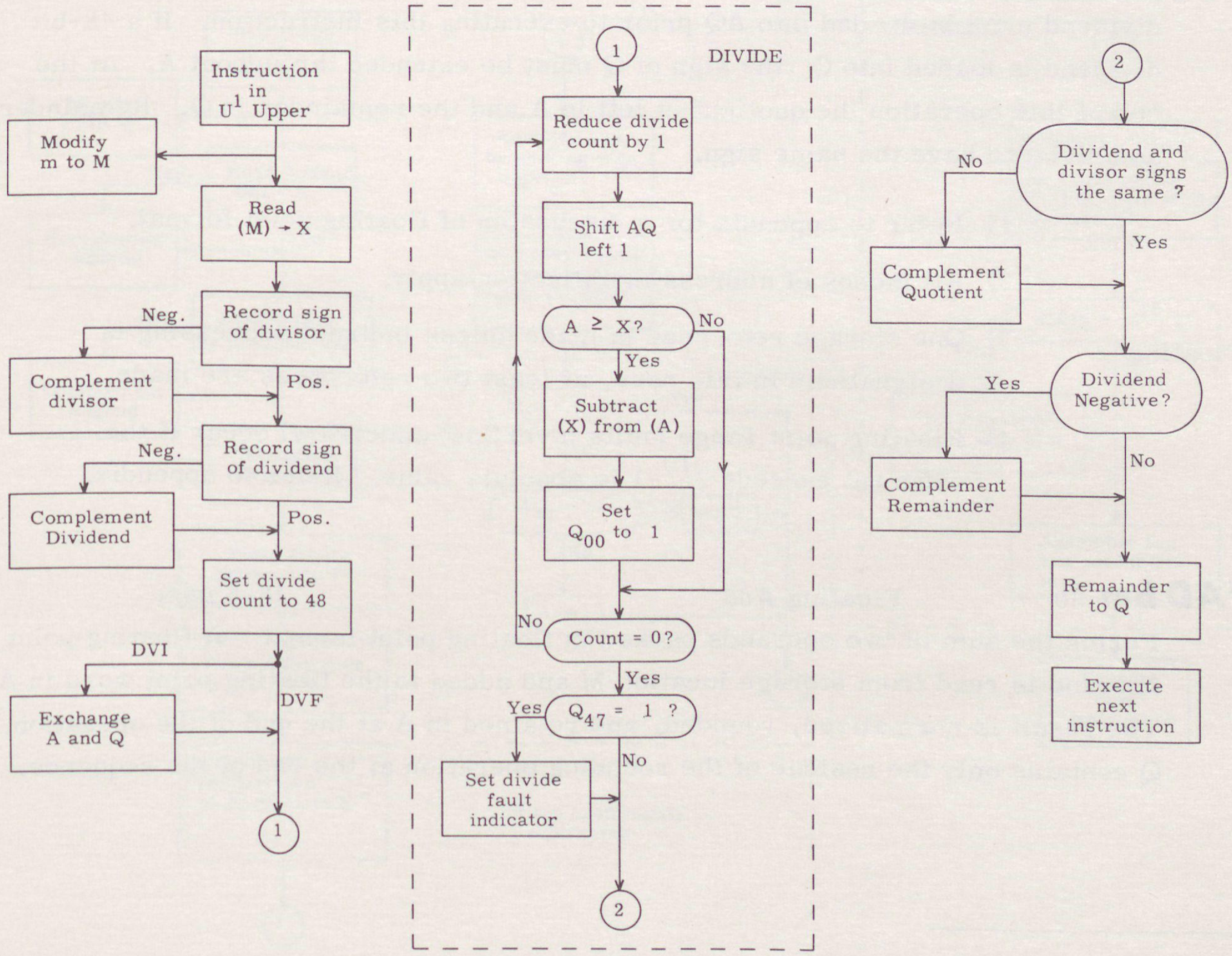
MUI and MUF

---

* n = Number of ones in multiplier

DVIbm 25 Divide Integer 65.2 μsec

Divides a 96-bit integer dividend by a 48-bit integer divisor. The 96-bit dividend must be formed in the QA register prior to executing the instruction. If a 48-bit dividend is loaded into A, the sign of Q must be set. That is, the sign of the dividend in A must be extended throughout Q. The 48-bit divisor is read from the storage location specified by the execution address. The quotient is formed in A and the remainder is left in Q at the end of the operation. Dividend and remainder have the same sign.



DVI and DVF

MUF bm 26 Multiply Fractional 25.2 + .8n* μsec

Forms a 96-bit product from two 48-bit operands. The operands are treated as fractions with the binary point immediately to the right of the sign bit. The multiplier must be loaded into A prior to executing the instruction. The multiplicand is read into X from the storage location specified by M. The 96-bit product is contained in AQ.

DVF bm 27 Divide Fractional 65.2 μsec

Divides a 96-bit quantity by a 48-bit divisor. All operands are treated as fractions with the binary point immediately to the right of the sign bit. The 96-bit dividend must be loaded into AQ prior to executing this instruction. If a 48-bit dividend is loaded into Q, the sign of Q must be extended throughout A. At the end of this operation the quotient is left in A and the remainder in Q. Remainder and dividend have the same sign.

Floating

1) Refer to appendix for a discussion of floating point format.
2) All modes of address modification apply.
3) One storage reference is made unless indirect addressing is designated. In this case, at least two references are made.
4) Floating point range faults (overflow-underflow) occur if the exponent exceeds $2^{10}-1$ in absolute value. Refer to appendix.

FAD bm 30 Floating Add 18.8 μsec

Forms the sum of two operands packed in floating point format. A floating point operand is read from storage location M and added to the floating point word in A. The result is normalized, rounded, and retained in A at the end of the operation. Q contains only the residue of the rounding operation at the end of the sequence.

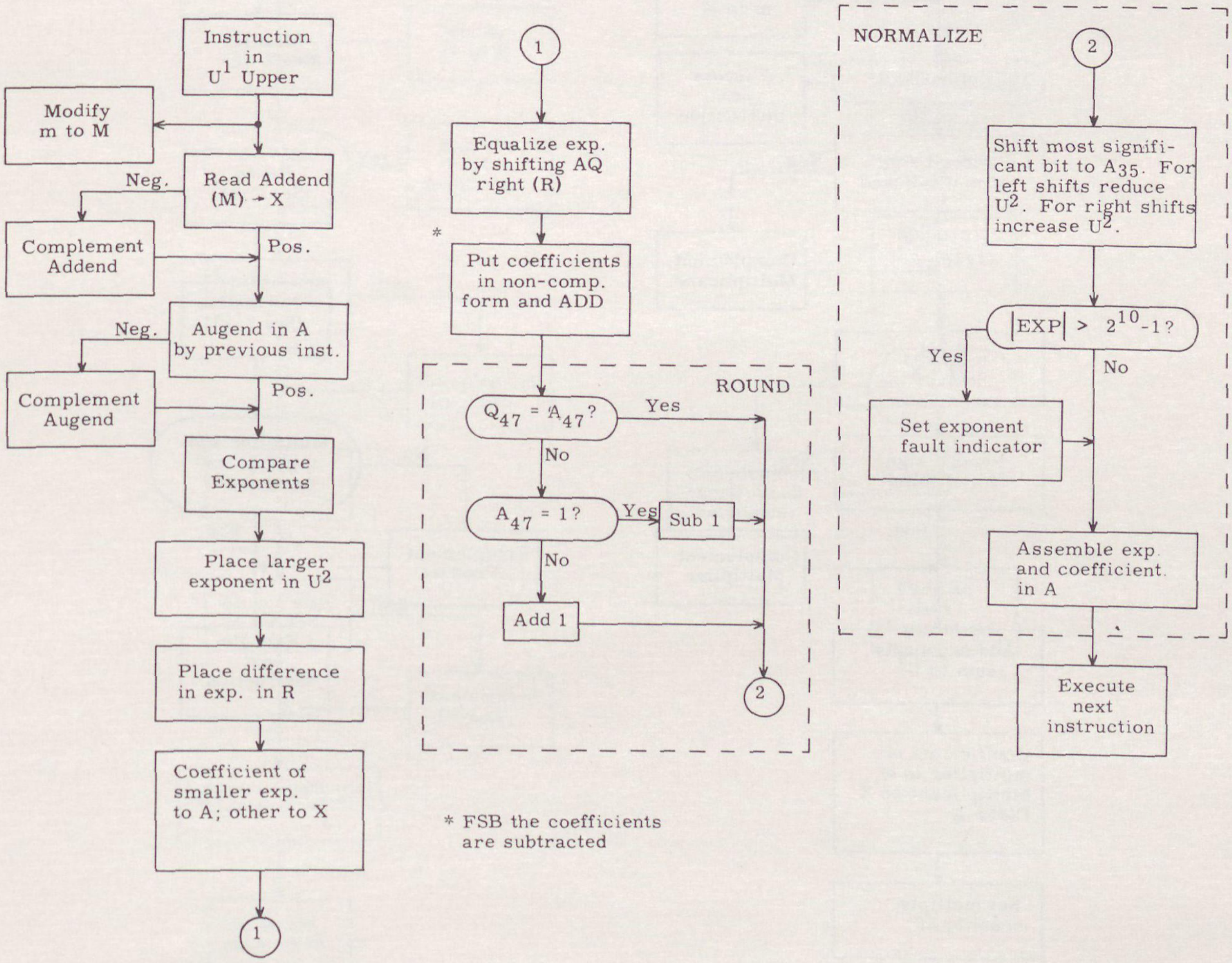
---

*n = Number of ones in multiplier

FSB bm 31 Floating Subtract 18.8 μsec

Forms the difference of two 48-bit operands in floating point format. The subtrahend is acquired from storage address M and is subtracted from the minuend in A. The result is rounded and normalized if necessary and retained in A. The residue from the rounding operation is left in Q at the end of the sequence.

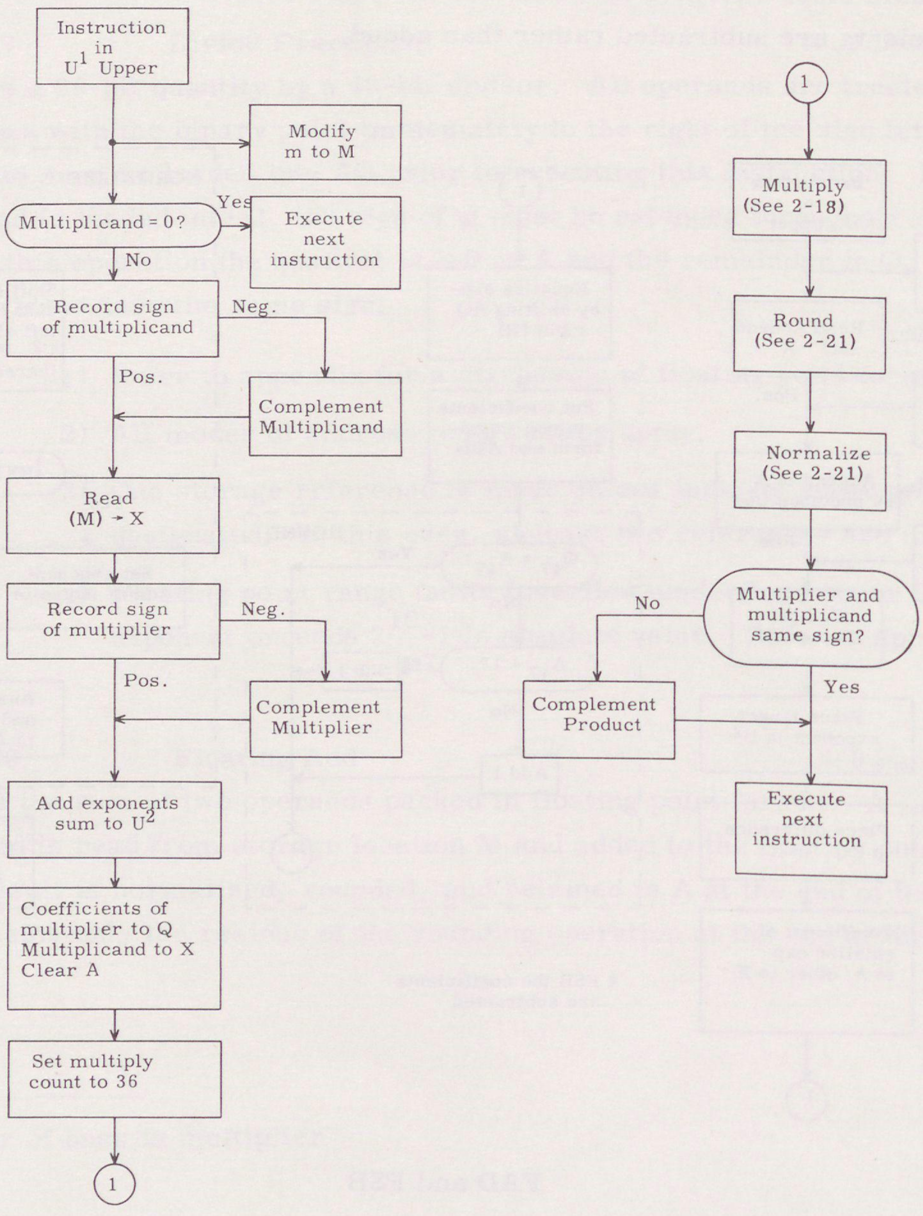
The basic steps executed in a FSB are the same as those for FAD except the coefficients are subtracted rather than added.



FAD and FSB

FMU bm 32 Floating Multiply 36.0 μsec

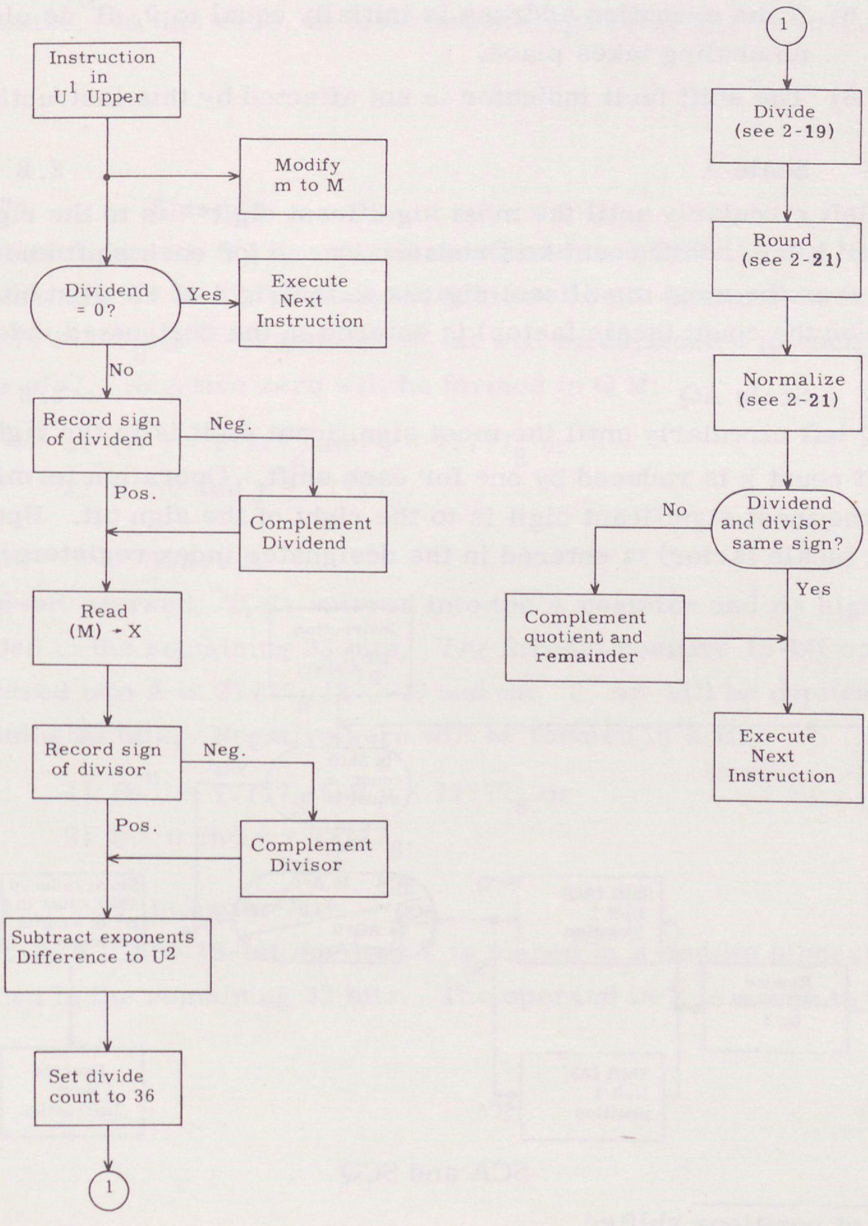
Forms the product of an operand in floating point format with the previous contents of A also in floating point format. The operand is read from storage location M. The product is rounded and normalized if necessary and retained in A. The residue from the rounding operation is left in Q at the end of the sequence.



FMU

FDV bm 33 Floating Divide 56.0 μsec

Forms the quotient of two 48-bit operands in floating point format. The dividend must be loaded into A prior to executing this instruction. The divisor is read from the storage location specified by M. The quotient is rounded and normalized if necessary and retained in A at the end of the operation. The residue from the rounding operation is left in Q at the end of the operation.



FDV

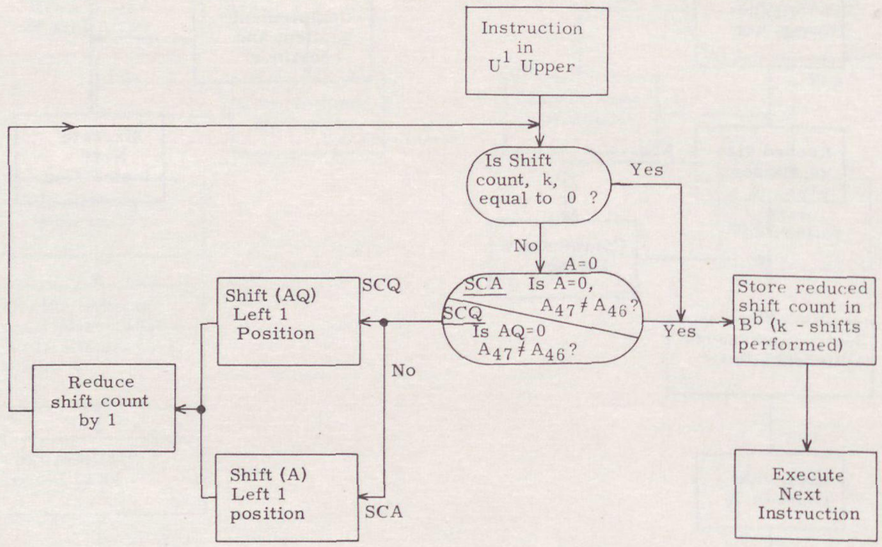
1) Address modification does not apply. Rather, the index register is used to preserve the scale factor.
2) If b = 0, scaling is executed but the scale factor is lost.
3) If b = 7, indirect addressing is used and at least one storage reference is made.
4) If (A) i is already scaled or equal to positive or negative zero, $k \rightarrow B^b$ and scaling is not executed.
5) If the execution address is initially equal to 0, B^b is cleared and no scaling takes place.
6) The shift fault indicator is not affected by this instruction.

SCA bk 34 Scale A 2.8 + .4s* μsec

Shifts A left circularly until the most significant digit** is to the right of the sign bit or until k = 0. Shift count k is reduced by one for each shift and terminates when k = 0 or the most significant digit is to the right of the sign bit. Upon termination the count (scale factor) is entered in the designated index register.

SCQ bk 35 Scale AQ 2.8 + .4s μsec

Shifts AQ left circularly until the most significant digit is to the right of the sign bit. Shift count k is reduced by one for each shift. Operation terminates when k = 0 or the most significant digit is to the right of the sign bit. Upon termination the count (scale factor) is entered in the designated index register.



SCA and SCQ

*s = Number of positions shifted

**When a negative number is being scaled, "0's" are significant digits.

NO ADDRESS

1) All modes of address modification apply to ENQ, ENA, and INA instructions.

2) Relative addressing cannot be used for ENI and INI instructions. Only direct and indirect addressing are used.

3) No storage reference is made during these instructions unless indirect addressing is designated. In this case, at least one storage reference is made.

ENQ by 04 Enter Q 3.0 μsec

The 15-bit operand, Y, is entered into Q and its highest order bit is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into Q is 37777_8 $(2^{14}-1)$ and its "0" bit will be duplicated in each of the remaining 33 bits of Q. Negative zero will be formed in Q if:

1) $(B^b) = 77777_8$ and $y = 77777_8$ or
2) $b = 0$ and $y = 77777_8$.

ENA by 10 Enter A 3.0 μsec

The 15-bit operand, Y, is entered into the A register and its highest order bit is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into A is 37777_8 $(2^{14}-1)$ and the "0" bit will be duplicated in each of the remaining 33 bits. Negative zero will be formed in A if:

1) $(B^b) = 77777_8$ and $y = 77777_8$ or
2) $b = 0$ and $y = 77777_8$.

INA by 11 Increase A 3.0 μsec

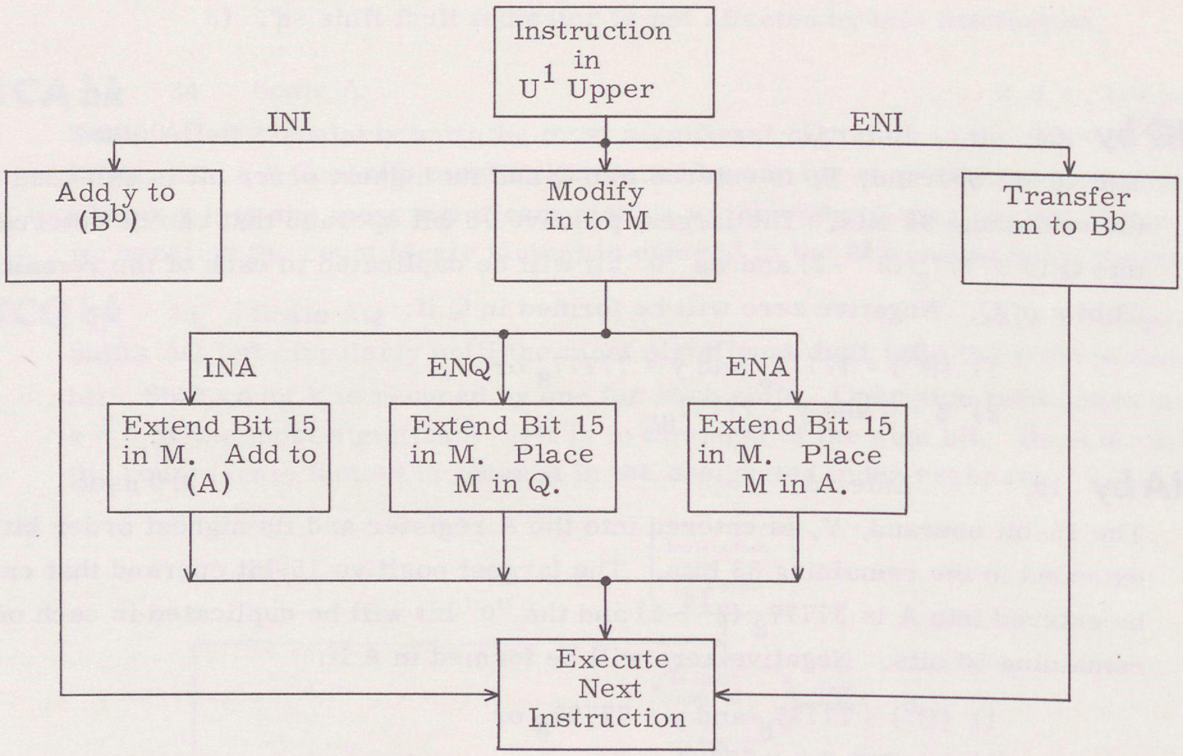
Adds Y to A. The 15-bit operand Y is placed in X and its highest order bit is extended in the remaining 33 bits. The operand in X is added to (A).

ENIby 50 Enter Index 3.0 μsec

Replaces (B^b) with the operand y. If b = 0, this instruction becomes a pass (do nothing) instruction.

INIby 51 Increase Index 3.0 μsec

Increases (B^b) by the operand y. If the b designator is zero, this instruction becomes a pass (do nothing) instruction.



No Address

JUMPS AND STOPS

Normal
1) Address modification does not apply to these instructions.
2) One storage reference is made.

A jump instruction causes a current program sequence to terminate and initiates a new sequence at a different location in storage. The Program Address register, P, provides the continuity between program steps and always contains the storage location of the current program step.

When a jump instruction occurs, P is cleared and a new address is entered. In all jump instructions, the execution address, m, specifies the beginning address of the new program sequence. The word at address m is read from storage, placed in U^1 and the upper instruction (first instruction of the new sequence) is executed.

Some of the jump instructions are conditional upon a register containing a specific value or upon the position of an operator's jump or stop key on the console. If the criterion is satisfied, the jump is made to location m. If it is not satisfied, the program proceeds in its regular sequence to the next instruction.

A jump instruction may appear in either position in a program step. If the jump instruction appears in the first (upper) part of the program step and the jump is taken, the second (lower) part of the program step is not executed. If the instruction appears in the lower part, the upper part is executed in the normal manner.

AJPjm 22 A Jump 7.2 μsec

Jumps to m if the conditions of the A register specified by the jump designator, j, exist. If not, the next instruction is executed.

j = 0 Jump if (A) = 0

j = 1 Jump if (A) ≠ 0

j = 2 Jump if (A) = +

j = 3 Jump if (A) = -

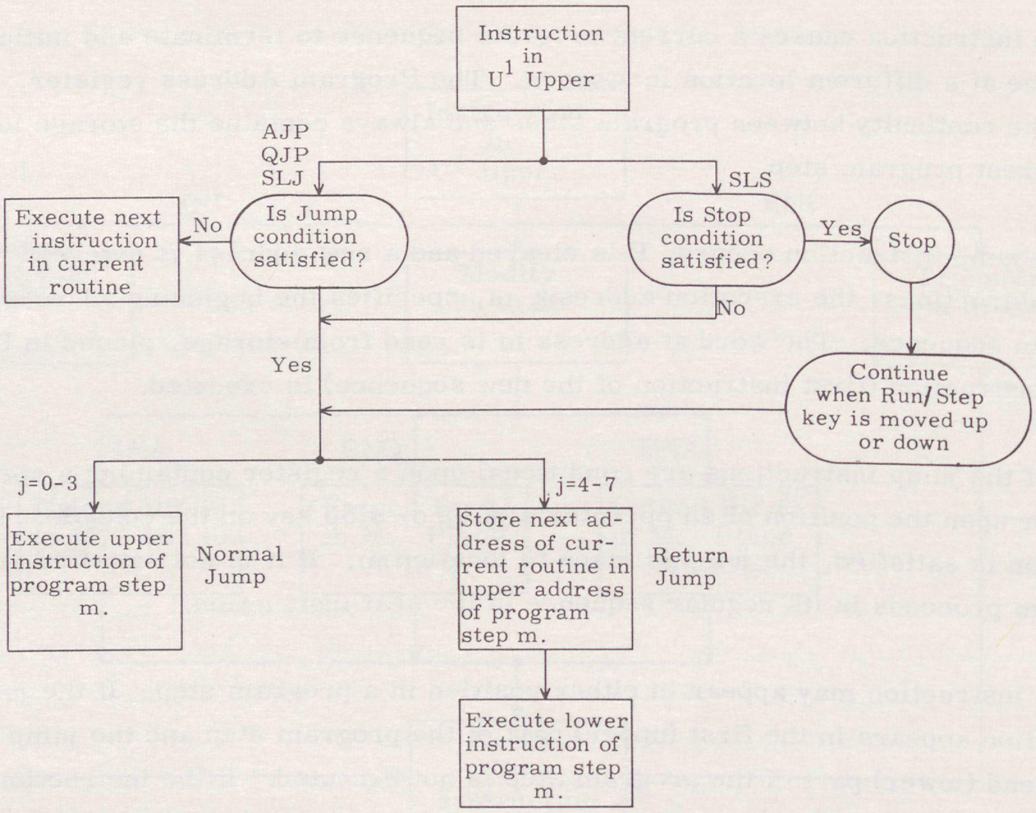
When (A) is negative zero the interpretation is:

j = 0 The jump is executed because, in this case, negative zero is recognized as positive zero.

j = 1 The jump is not executed.

j = 2 The jump is not executed because the sign bit is a "1".

j = 3 The jump is executed because the sign bit is a "1".



AJP, QJP, SLJ, and SLS

QJP jm 23 Q Jump 7.2 μsec

Jumps to m if the condition of the Q register specified by the jump designator, j, exists. If not, the next instruction is executed.

j = 0 Jump if (Q) = 0

j = 1 Jump if (Q) ≠ 0

j = 2 Jump if (Q) = +

j = 3 Jump if (Q) = -

When (Q) is negative zero the AJP interpretation applies.

SLJjm 75 Selective Jump 7.2 μsec

Jumps to m if the condition of the jump keys specified by j exists. If not, the next instruction is executed.

j = 0 Jump unconditionally

j = 1 Jump if jump key 1 is set

j = 2 Jump if jump key 2 is set

j = 3 Jump if jump key 3 is set

SLSjm 76 Selective Stop 7.2 μsec

Stops at present step in the sequence if the condition of the stop key specified by j exists. If the stop condition exists, the stop is executed, and the jump is executed unconditionally when the Run/Step key is moved to the RUN or STEP position. If the stop condition is not satisfied, the jump is executed unconditionally.

j = 0 Stop unconditionally

j = 1 Stop if stop key 1 is set

j = 2 Stop if stop key 2 is set

j = 3 Stop if stop key 3 is set

1) Address modification does not apply to these instructions.

2) One storage reference is made.

Return Jump

A return jump begins a new program sequence at the lower instruction portion of the program step to which the jump is made. At the same time, the address portion of the upper instruction of that program step is replaced with the address of the next program step in the main program. This instruction allows a return to the main program after completing the subprogram sequence.

AJP jm 22 A Jump 7.2 μsec

Executes a return jump to storage location m if the condition of the A register specified by j exists. If not, the next instruction is executed.

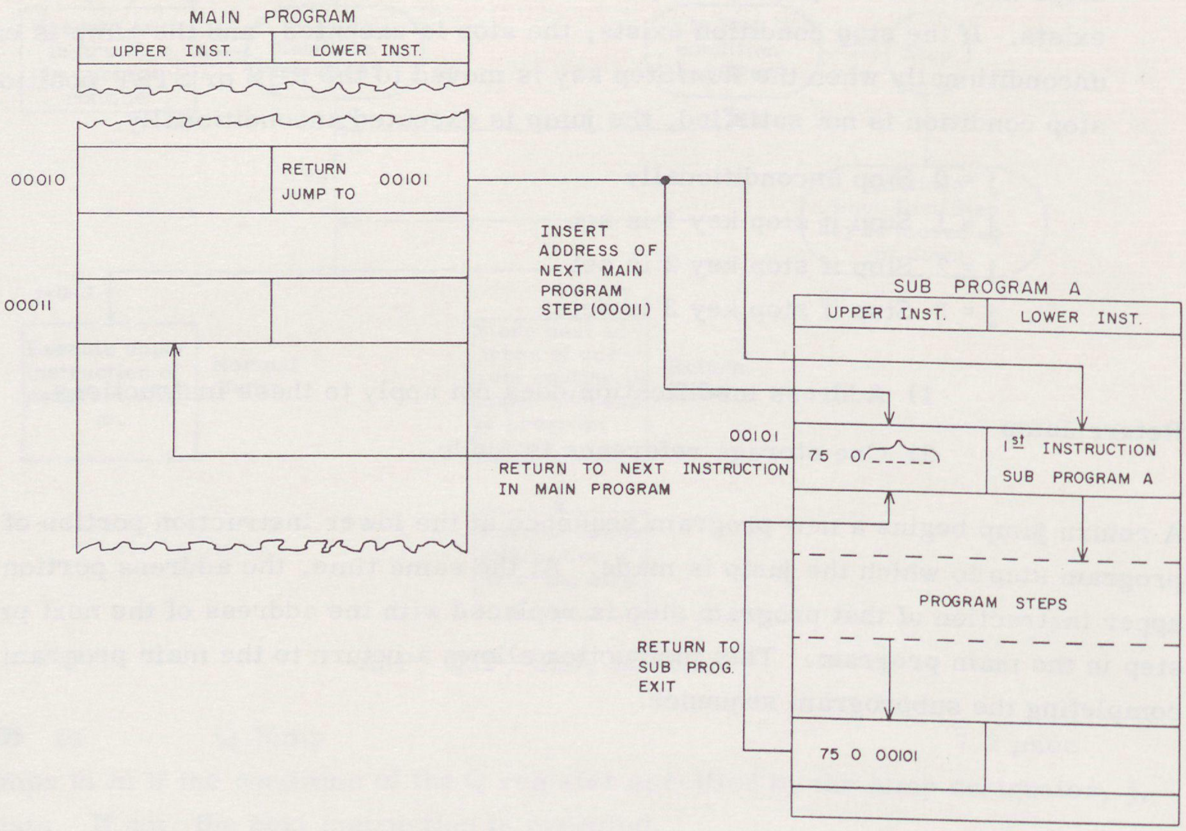
j = 4 Return jump if (A) = 0

j = 5 Return jump if (A) ≠ 0

j = 6 Return jump if (A) = +

j = 7 Return jump if (A) = -

Note: If (A) = negative zero, refer to the AJP instruction.



Return Jump

QJPjm 23 Q Jump 7.2 μsec

Executes a return jump to storage location m if the condition of the Q register specified by j exists. If not, the next instruction is executed.

j = 4 Return jump if (Q) = 0

j = 5 Return jump if (Q) $\neq$ 0

j = 6 Return jump if (Q) = +

j = 7 Return jump if (Q) = -

Note: If (Q) = negative zero, refer to the AJP instruction.

SLJjm 75 Selective Jump 7.2 μsec

Executes a return jump to storage location m on condition j where condition j represents the setting of the jump keys. If the condition is not satisfied, the next instruction is executed.

j = 4 Return jump unconditionally

j = 5 Return jump if jump key 1 is set

j = 6 Return jump if jump key 2 is set

j = 7 Return jump if jump key 3 is set

Note: The set position of a jump key is in the up position.

SLSjm 76 Selective Stop 7.2 μsec

Stops on condition j and executes a return jump to storage location m if the Run/Step key is moved in the RUN or STEP position. If the stop condition is satisfied, the stop is executed and the return jump is executed when the Run/Step key is moved in either position. If the stop condition is not satisfied, the stop is not executed and the return jump is executed unconditionally.

j = 4 Stop unconditionally

j = 5 Stop if stop key 1 is set

j = 6 Stop if stop key 2 is set

j = 7 Stop if stop key 3 is set

STORAGE TEST

1) All modes of address modification apply to these instructions.

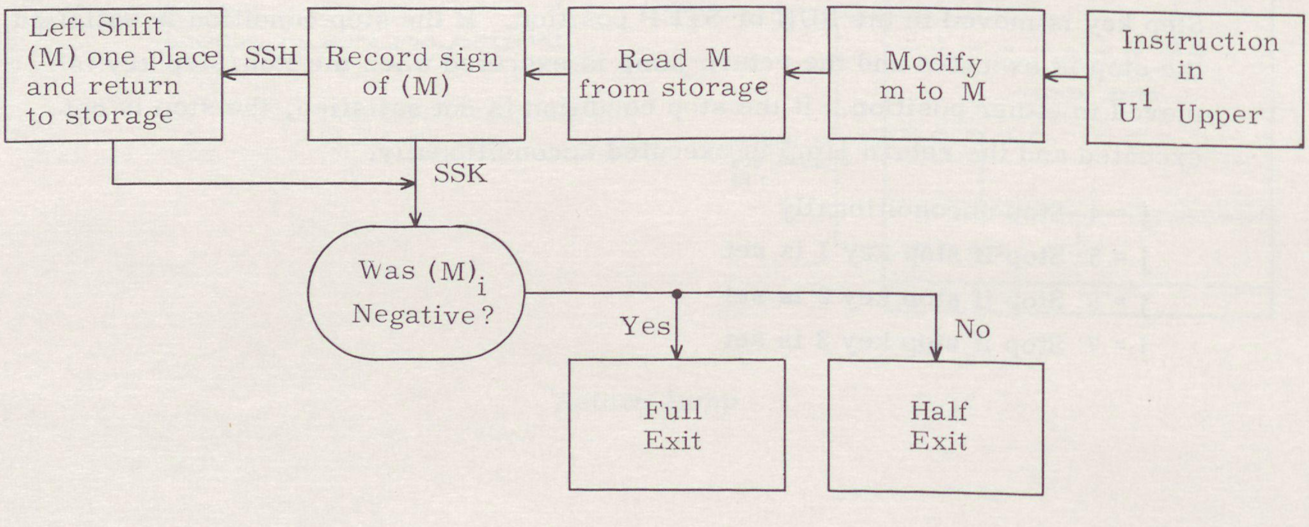
2) At least one storage reference is made unless indirect addressing is designated in which case at least two storage references are made.

SSK bm 36 Storage Skip 8.8 μsec

Senses the sign bit of the operand in storage location M. If the sign is negative, a full exit is taken. If the sign is positive, a half exit is taken. The contents of the operational registers are left unmodified. SSK is usually restricted to an upper instruction. If used as a lower instruction and the sign of (M) is negative, a full exit will be executed. If the sign is positive, it will half exit upon itself and never execute a full exit.

SSH bm 37 Storage Shift 12.8 μsec

Senses the sign bit of the quantity in storage location M. If the sign bit is negative a full exit is taken, and if the quantity is positive a half exit is taken. In either case the quantity is shifted left circularly one bit before the exit. This instruction is usually restricted to the upper position. If used as a lower instruction and the sign of (m) is positive, the instruction will half exit upon itself until a negative sign bit is found. The contents of the operational registers are left unmodified.



SSH and SSK

LOGICAL

1) All modes of address modification apply to these instructions.

2) The LDL, ADL, SBL and STL instructions achieve their result by forming a logical product. A logical product is a bit by bit multiplication of two binary numbers (logical AND condition):

$0 \times 0 = 0$	$1 \times 0 = 0$
$0 \times 1 = 0$	$1 \times 1 = 1$

3) A logical product is used, in many cases, to select specific portions of an operand for entry into another operation. For example, if only a specific portion of an operand in storage is to be added to (A), as the operand passes through X it is subjected to a mask comprised of a predetermined pattern of "0's" and "1's". Forming the logical product of (X) and the mask causes X to retain the original contents only in those stages which have corresponding "1's" in the mask. When only the selected bits remain in X, the instruction proceeds to conclusion.

SSTbm 40 Selective Set 7.2 μsec

Sets the individual bits of A to "1" where there are corresponding "1's" in the word at storage location M. "0" bits in the word at storage location M do not modify the corresponding bits in A. In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

SCMbm 42 Selective Complement 7.2 μsec

Individual bits of A are complemented where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's", the associated bits of A remain unchanged.

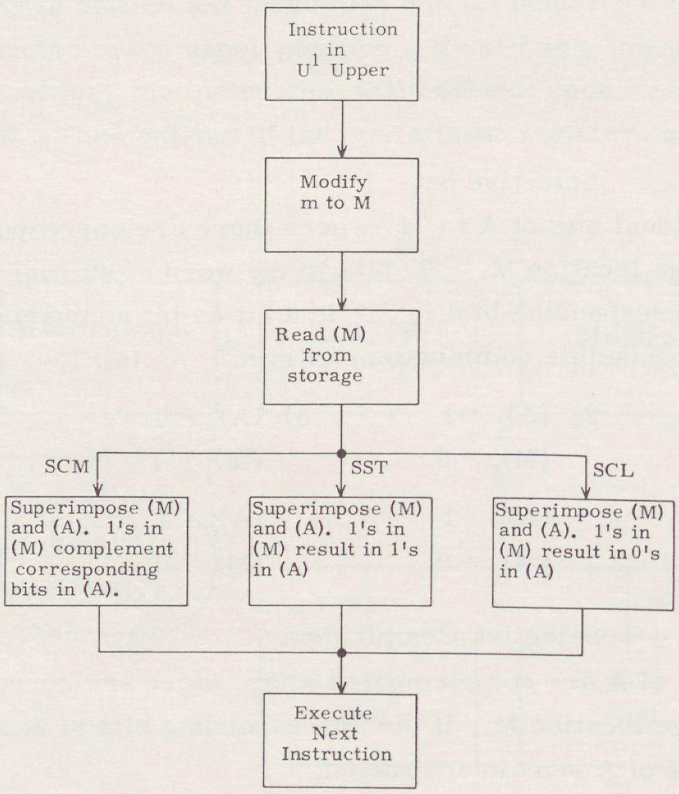
1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

SCL bm 41 Selective Clear 7.2 μsec

Clears individual bits of A where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's", the associated bits of A remain unchanged.

In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 0$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$



SCM, SST, and SCL

SSU bm 43 Selective Substitute 7.4 μsec

Substitutes selected portions of an operand at storage address M into the A register where there are corresponding "1's" in the Q register (mask). The portions of A not masked by "1's" in Q are left unmodified.

L D L bm 44 Load Logical 7.4 μsec

Loads A with the logical product of Q and the contents of the designated storage location, M. The operand can be in either Q or M.

ADL bm 45 Add Logical 7.4 μsec

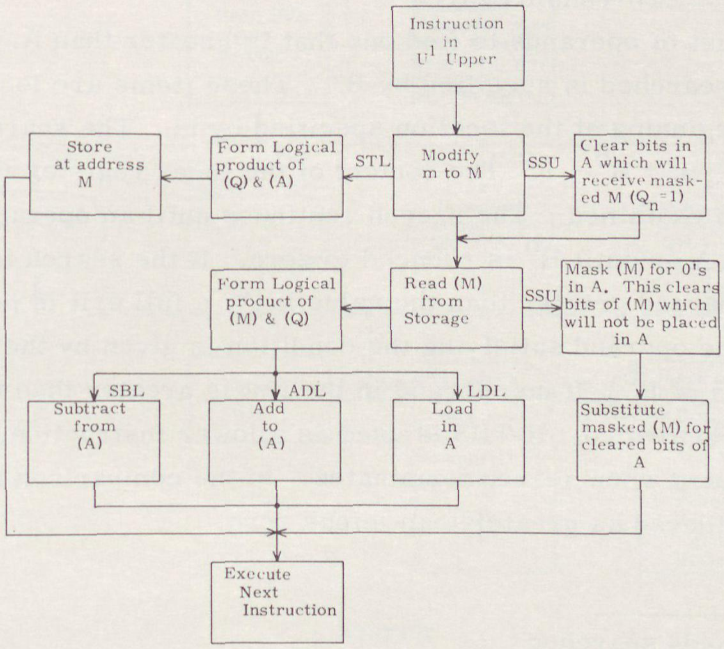
Adds to A the logical product of Q and the quantity in location M; the mask may be in Q or storage. Once the logical product is formed addition follows normal rules (appendix).

S B L bm 46 Subtract Logical 7.4 μsec

Subtracts from A the logical product of the Q register and the quantity in storage location M. The mask may be in Q or storage. When the logical product is formed, the subtraction proceeds in the normal manner. (See appendix.)

S T L bm 47 Store Logical 7.2 μsec

Replaces the bits in storage location M with the logical product of Q and A registers. Neither (A) nor (Q) are modified. The mask may be located in A or Q.



ADL, LKL, SBL, SSU, and STL

STORAGE SEARCH

1) If b = 0 in the following instructions only the word at storage location m will be searched.

2) If b = 7, indirect addressing is used to obtain the execution address and b designator.

3) If $(B^b) = 0$, no search is made.

EQS bm 64 Equality Search 4.0 + 3.6r* μsec

Searches a list of operands to find one that is equal to A. The number of items to be searched is specified by B^b. These items are in sequential addresses beginning at the location specified by m. The search begins with the last address, $m + B^b - 1$. B^b is reduced one count for each word that is searched until an operand is found that equals A or until B^b equals zero. If the search is terminated by finding an operand that equals A, a full exit is made. The address of the operand satisfying this condition is given by the sum of m and the final contents of B^b. If no operand is found that equals A, a half exit is taken. Positive zero and minus zero are recognized as the same quantity. When EQS is used as a lower instruction, the next instruction will always be executed when the search terminates.

THS bm 65 Threshold Search 4.0 + 3.6r μsec

Searches a list of operands to find one that is greater than A. The number of items to be searched is specified by B^b. These items are located in sequential addresses beginning at the location specified by m. The search begins with the last address, $m + B^b - 1$. The content of the index register is reduced by one for each operand examined. The search continues until an operand is reached that is greater than A or until B^b is reduced to zero. If the search is terminated by finding an operand greater than the value in A, a full exit is performed. The address of the operand satisfying the condition is given by the sum of m and the final contents of B^b. If no operand in the list is greater than the value in A, a half exit is performed. If THS is used as a lower instruction, the next instruction will be executed when search terminates. In the comparison made here positive zero is considered as greater than minus zero.

---

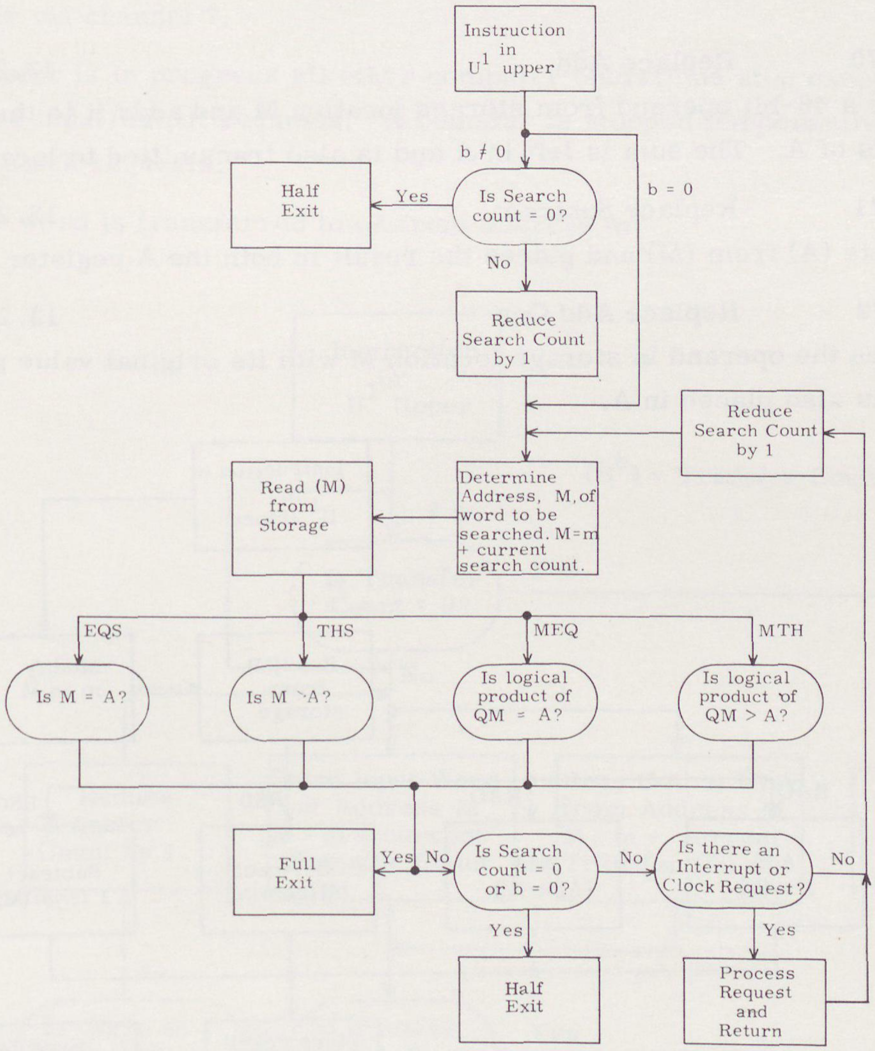
*r = Number of words searched

MEQ bm 66 Masked Equality Search 4.0 + 3.6r* μsec

Searches a list of operands to find one such that the logical product of (Q) and (M) is equal to (A). This instruction, except for the mask, operates in the same manner as an equality search.

MTH bm 67 Masked Threshold Search 4.0 + 3.6r μsec

Searches a list of operands to find one such that the logical product of (Q) and (M) is greater than (A). Except for the mask, this instruction operates in the same manner as the threshold search.



Search

*r = Number of words searched

REPLACE

1) All modes of address modification apply to these instructions.

2) During the execution of the replace instructions, two storage references are made. If indirect addressing is designated, at least three references are made.

3) If the capacity of the A register $\pm (2^{47} - 1)$ is exceeded during the execution of the following instructions, an arithmetic overflow fault is produced. (Refer to appendix.)

RAD bm 70 Replace Add 13.2 μsec

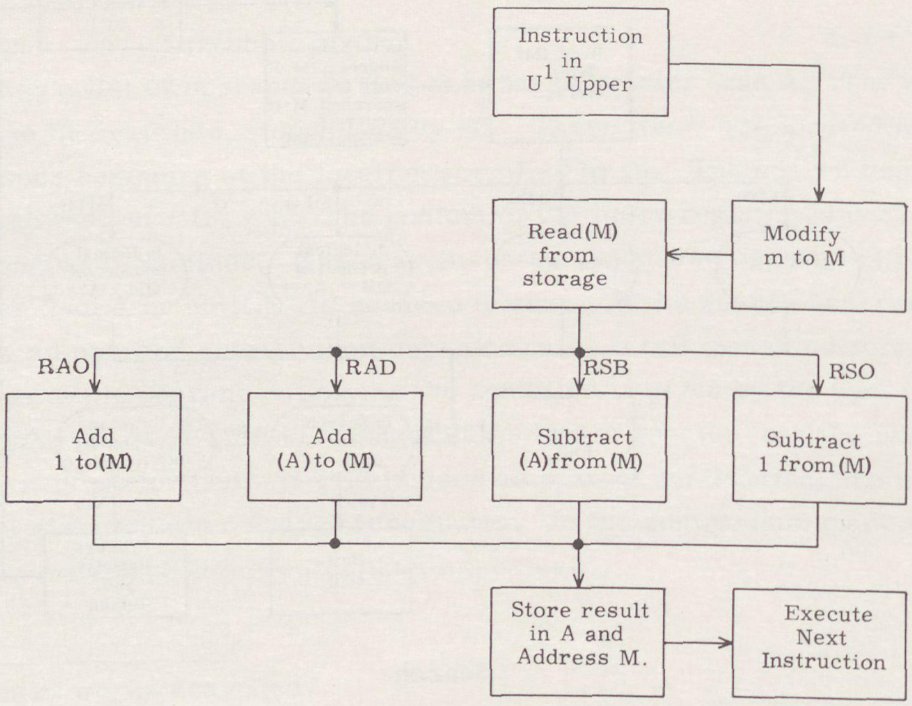
Obtains a 48-bit operand from storage location M and adds it to the initial contents of A. The sum is left in A and is also transmitted to location M.

RSB bm 71 Replace Subtract 13.2 μsec

Subtracts (A) from (M) and places the result in both the A register and location M.

RAO bm 72 Replace Add One 13.2 μsec

Replaces the operand in storage location M with its original value plus one. The result is also placed in A.



Replace

RSO bm 73 Replace Subtract One 13.2 μsec

Replaces the operand in storage location M with its original contents minus one. The difference is also left in A; the original contents of A and M are destroyed.

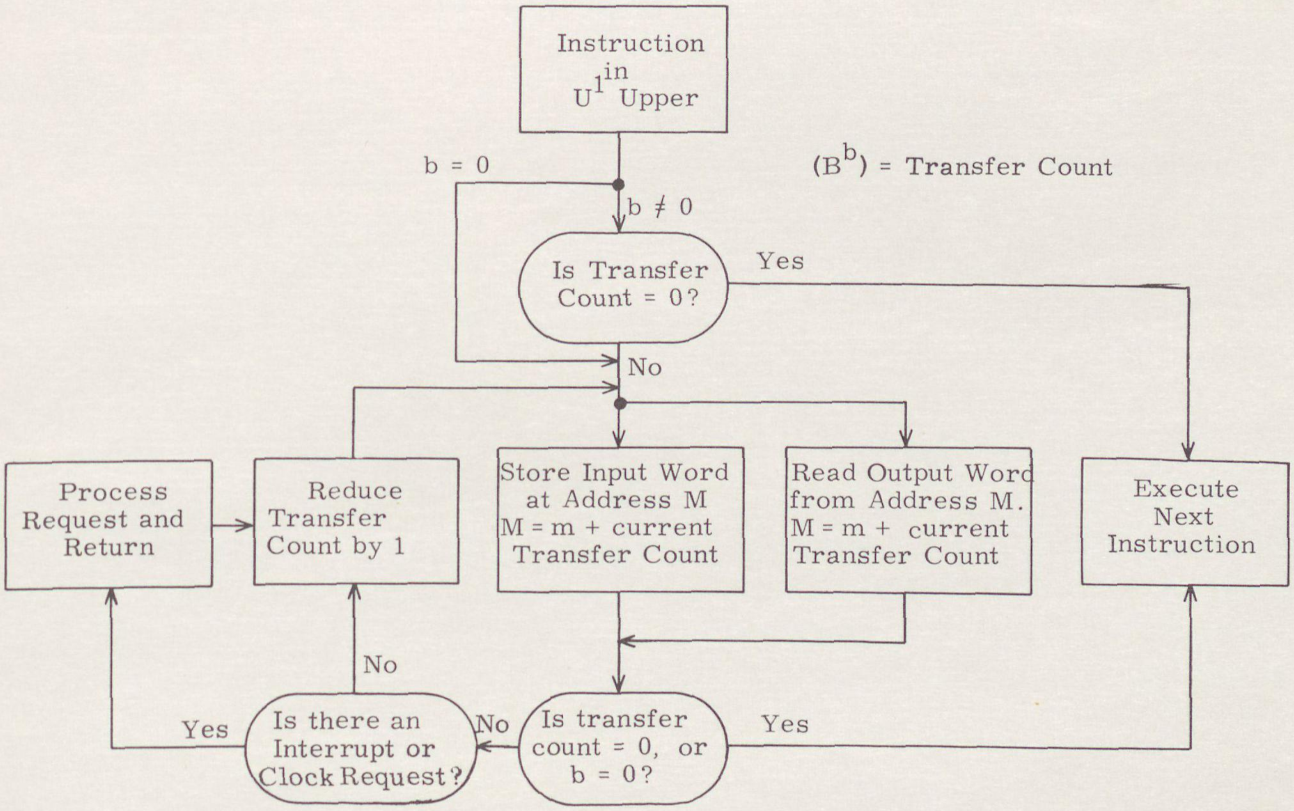
TRANSFER

1) Relative address modification is not used for the following instructions. Only direct and indirect addressing can be used.

2) The index registers contain the number of words to be transferred into or out of the computer via channel 7.

3) When a transfer is in progress all other computer operations stop except the processing of input/output requests. A transfer is stopped temporarily to process interrupter clock requests.

4) If b = 0, one word is transferred to or from address m.



Transfer

INT bm 62 Input Transfer 4.0 + 4.8r* μsec

Transfers a block of data from an external equipment into storage. The number of words to be transferred is specified by B^b. These words are stored in sequential addresses beginning at the location specified by the execution address, m. The transfer begins by storing the first input word in the last address in the sequence, $m + B^b - 1$. As each word is transferred, B^b is reduced by one until it is equal to zero.

OUT bm 63 Output Transfer 4.0 + 4.8r μsec

Transfers a block of data from computer storage to an external equipment. The number of words to be transferred is specified by B^b. The words to be transferred are located in sequential addresses beginning at the location specified by the execution address, m. The transfer begins by obtaining the first output word from the last address, $m + B^b - 1$. As each word is transferred B^b is reduced by one until it is equal to zero.

---

*r = Number of words transferred

CHAPTER 3

INPUT/OUTPUT

METHODS OF DATA EXCHANGE

The computer communicates with external equipment via a single transfer channel and six buffer channels. The transfer channel which provides for very high speed communication is program initiated and controlled. The buffer channels provide for the normal exchange of data and, although program initiated, operate independently of the program.

HIGH SPEED TRANSFER CHANNEL

The high speed transfer channel (channel 7) handles both input and output communications between computer and high speed equipments. Information is transferred between the computer and external equipment in blocks at a word by word rate. The transfer rate is usually dependent on the speed of the external equipment as the computer can perform transfers at a maximum (approximate) rate of one word every 4.8 μsec.

As many as 8 different equipments (optimum conditions) may be connected to the transfer channel. However, only one equipment can use the channel at any given instant and the current transfer operation must be completed before a different equipment can use the channel.

BUFFER CHANNELS

The six independent buffer channels are grouped in three pairs:

Input:	Channel 1	Output:	Channel 2
	Channel 3		Channel 4
	Channel 5		Channel 6

All six buffer channels can communicate concurrently with external equipments. This is accomplished by an auxiliary scanner which processes only one channel at a given instant - so that when more than one channel is active each channel is given a turn in rotation to buffer one word of information. The rate of data flow on each buffer channel is determined by the operating speed of the external equipment connected to that channel.

INITIATION AND CONTROL OF DATA EXCHANGE

TRANSFER

A transfer operation is initiated and controlled by the computer program. An INT or OUT instruction transfers the number of words designated by the contents of an index register. The starting storage location of the transfer is specified by the execution address of the instruction. (Refer to chapter 2 for a discussion of the INT or OUT instructions.)

All computer operations, with the exception of previously initiated buffers and processing of interrupt or clock requests, stop while the transferring of words is in progress. (Refer to page 3-6).

BUFFER

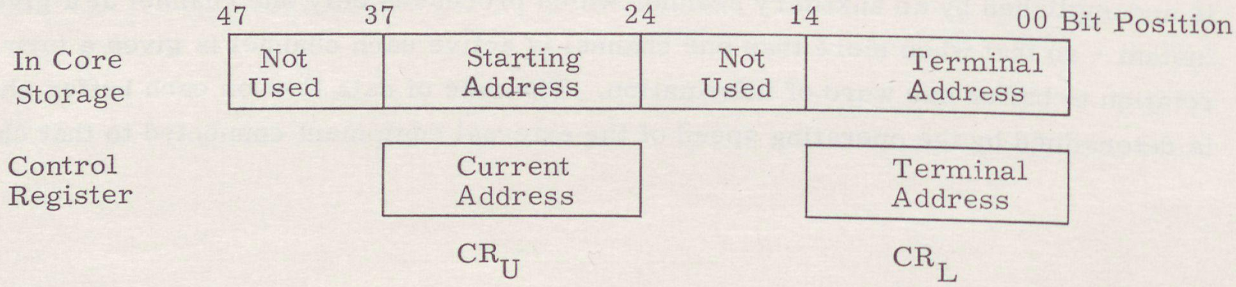
A buffer operation is program initiated but, in contrast to transfer operations, proceeds under controls that are independent of the main program.

Buffer Control Word

Information is buffered in blocks at a word by word rate. The initial and terminal storage addresses of the block comprise the buffer control word. Each of the six buffer channels has an assigned control register and storage address which holds the buffer control word.

Channel	Control Register	Address of Control Word
1	1	00001
2	2	00002
3	3	00003
4	4	00004
5	5	00005
6	6	00006

Buffer Control Word



The terminal address is one greater than the last address to be used in the buffer. Prior to initiating a buffer operation, the terminating address must be entered into the lower address of the control word. The starting address is automatically entered into both the Control register and core storage when the buffer is initiated by an EXF instruction. During the buffer operation, the CR_U is updated as each word is buffered and thus holds the current address of the buffer. The control word in core storage is not updated.

External Function (EXF) Instructions

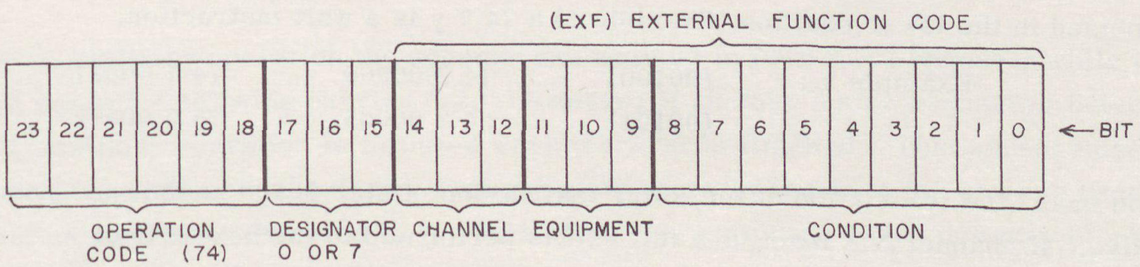
The EXF instructions initiate a buffer, sense for specified conditions, and select operations and equipment. EXF codes are listed in appendix 6.

There are three kinds of external instructions:

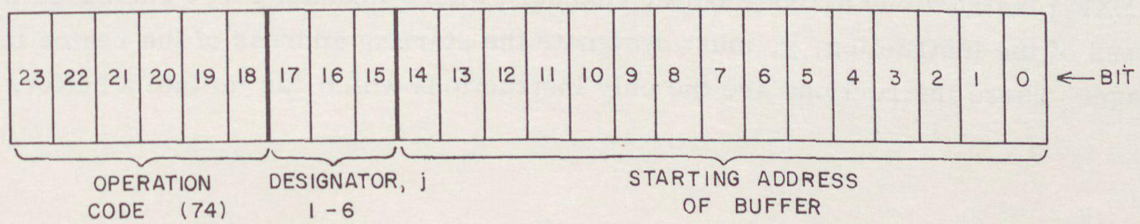
Select	74 0 XXXXX	
Sense	74 7 XXXXX	
Activate 0/I	74 j XXXXX	where j = 1-6

The composition of an external function instruction is shown below.

Select and Sense



Activate



The 74 0 (EXF Select) instructions select the external equipment which is to communicate with the computer and/or its mode of operation. The select instructions do not activate the buffer but, rather, establish initial operating conditions within the designated equipment so that information will be properly processed when the buffer is activated.

The EXF 7 instructions sense the condition of an external equipment or the internal conditions (faults) of the computer and will execute a full exit or half exit depending on the presence or absence of the condition.

The location of a 74 7 y instruction within an instruction word determines whether a skip or a wait will be performed.

When used in the upper instruction position (Example 1) a 74 7 y is a skip instruction.

Example 1 :	(00010)	74 7 00010	75 0 40000
	(00011)	53 1 00005	16 1 00032

In this example the translation of the upper instruction of a program step 00010 is Exit on Channel 1 active. If channel 1 is active the next instruction to be executed would be the upper instruction of step 00011, i.e., 53 1 00005. If channel 1 were inactive, the lower instruction of step 00010 would be executed.

When used in the lower position (Example 2) a 74 7 y is a wait instruction.

Example 2 :	(00100)	74 2 00600	74 7 00021
	(00101)	54 2 00005	75 0 00072

In this case, the translation of the lower instruction of step 00100 is Exit on Channel 2 inactive. If channel 2 is inactive a full exit is performed to the next pair of instructions, program step 00101. If, however, the channel is active, the instruction half exits and repeats itself until the channel becomes inactive. The sensing of conditions in no way alters the condition.

The EXF j instructions activate buffer channel j where j equals 1-6. The execution address of the instruction, y, must designate the starting address of the region in storage. These instructions are the only instructions which can initiate a buffer.

The following steps should be completed prior to initiating a buffer operation.

1) Sense for: (a) equipment ready and (b) channel inactive.
2) Select the external equipment and its mode of operation.
3) Substitute the terminal address into the buffer control word.

An equipment is ready if there is no motion, that is, a transmission is not taking place.

A buffer channel is active while data is being buffered. A buffer channel is inactive if the previous input/output operation has been completed.

The buffer must be satisfied (current address = terminating address) to inactivate the channel. This can be accomplished by activating the channel (74 j instruction) at the terminating address. This makes the channel inactive but no additional information is transmitted.

Auxiliary Scanner

After being initiated by the main program, data exchange on each buffer channel is controlled by buffer control section. In order that one buffer channel may not monopolize buffer control, an auxiliary scanner is used to initiate each one word buffer. The auxiliary scanner samples each buffer channel in the order: 1-3-2-6-4-5. When the scanner detects an auxiliary request from one of the buffer channels it stops and initiates a one-word auxiliary operation. Thus each channel has equal priority.

During an auxiliary operation the scanner can scan up to four more channels while the present operation is being carried out. However, if another action request is detected, another auxiliary operation is initiated when the first is finished. This arrangement gives auxiliary operations priority over program steps in requests for storage time. If no action request is detected on the four channels, a storage reference may be made by a program step and no auxiliary requests may be serviced until storage is released (6 μsec).

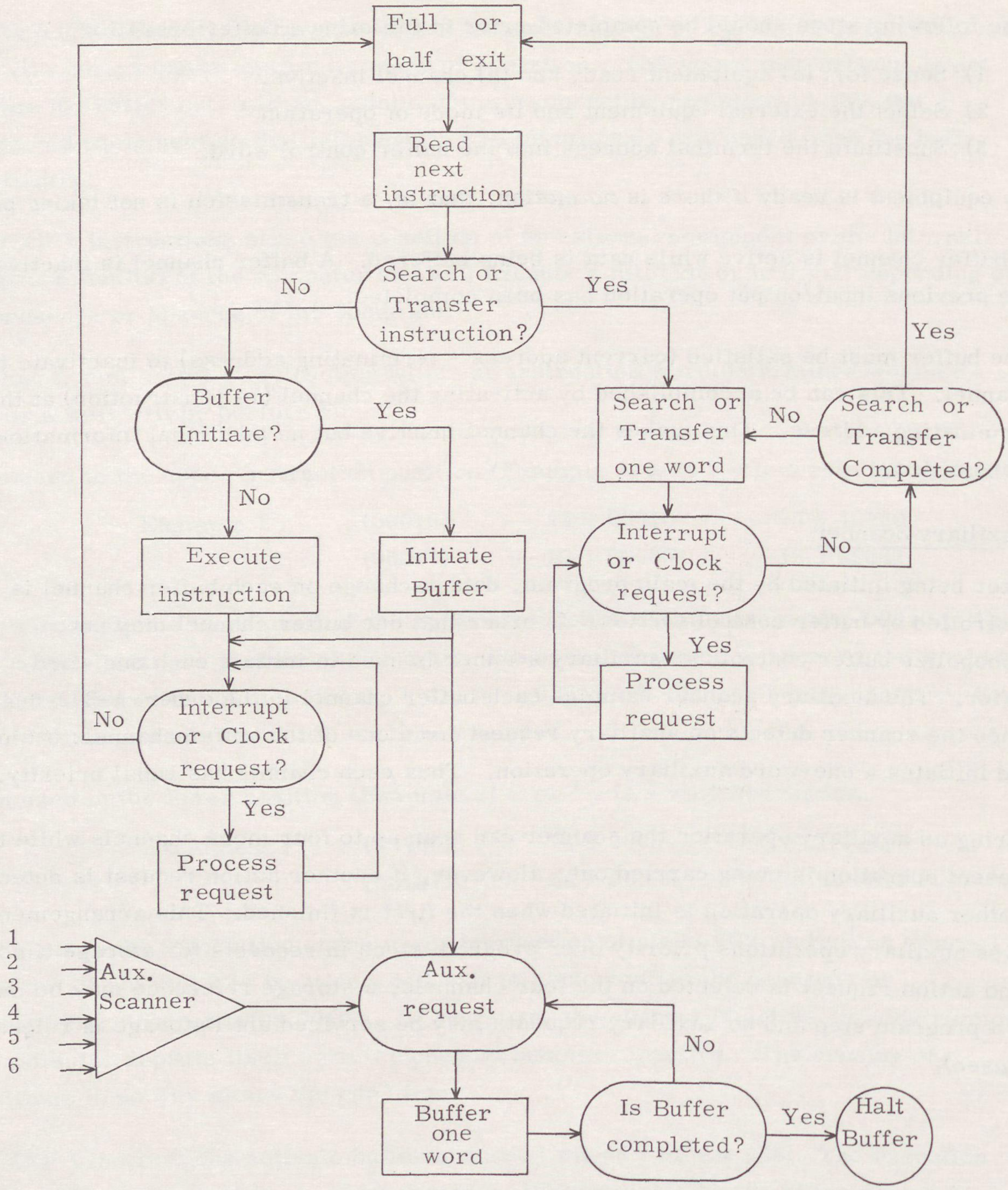


Figure 3-1. 1604-A Flow Chart

INTERRUPT

In each piece of external equipment as well as in parts of the internal computer control, certain conditions may arise which make it necessary that the main program be notified of their presence. The signal which notifies the computer of these conditions is called an interrupt and is program controlled. If an interrupt is desired when a particular condition arises, an external function select code (74.0) must select an interrupt on that condition. Unless such selection is made, no interrupt is produced when the condition arises. (See appendix 6 for the codes.)

When an interrupt occurs, the main program is halted and a previously programmed routine of instructions (interrupt routine) is performed which must determine the cause of the interruption and take appropriate action. After completing these operations, the interrupt routine must return to the main program. The main program resumes at the exact point from which the entrance to the interrupt routine was made.

INTERRUPT ROUTINE

The interrupt is processed by performing a jump to addresses 00007-00017. These are special addresses allocated for use as the entrance points to the interrupt routines and for the return from these routines to the main program.

Typically, addresses 00007-00017 (interrupt control words) contain two unconditional jump instructions. For example:

75 0 XXXXX   75 0 YYYYY

The upper instruction provides for the return to the main program upon completion of the interrupt routine. To accomplish this the upper address portion (XXXXX) is loaded with the contents of the P register when the interrupt routine is entered. The lower instruction jumps to the interrupt routine which begins with an instruction whose address is indicated in the lower address part of the control word (YYYYY).

In general, the interrupt routine (table 3-1) checks for all possible interrupt conditions by means of sense (74.7) instructions. After determining which selected condition caused the interrupt a jump is made to that portion of the routine which processes the interrupt.

After having been interrupted the computer cannot again be interrupted without returning to the main program via location 00007-00017.

TABLE 3-1. TYPICAL INTERRUPT SUBROUTINE

00007 through 00017	75 0 ____	75 0 int00		Exit/Entrance
	Address of next instruction in main program			
int00	74 7 00131	75 0 ovf00		Sense Overflow
int01	74 7 11101	75 0 crr00		
int02	74 7 ____	75 0 ____		
ovf00	____	____		Process Overflow
ovf01	____	____		
	____	____		
	____	74 0 00070	Clear Arithmetic Fault	
ovf	____	75 0 00007-17	Jump to Interrupt Address	

When an interrupt occurs, unless the indication of the condition or the interrupt selection is removed, the program will again be immediately interrupted upon return to the main program.

The internal faults for Divide, Shift, and Exponent Overflow and Underflow can be programmed in a similar way.

REAL TIME CLOCK

Address 00000 in core storage may be selected to provide a continuously operating record of elapsed time. The 48-bit quantity stored there is advanced by one every 1/60 of a second (accuracy is maintained by the 60-cycle power source). The content of address 00000, which may be sampled at any time by the program, gives elapsed time from the start of the real-time clock operation. The clock may be started by 74 0 01000 and is stopped by 74 0 02000. Starting the clock does not pre-set address 00000 in any way, but begins the periodic incrementing of its previous contents.

By selecting interrupt on arithmetic fault, and presetting the contents of address 00000, the real-time clock may be used to provide an interrupt of the main computer program. When interrupt occurs, check for clock overflow (location 0000).

CONSOLE INPUT/OUTPUT EQUIPMENT

Three input/output devices mounted on the console are standard equipment with the 1604-A computer. A Teletype BRPE 11 high speed punch and a CONTROL DATA 350 high speed reader provide for the processing of perforated paper tape. An electric typewriter provides for direct keyboard entry of data and for printed copy output. The console input/output units communicate with the central computer via buffer channels 1 (input) and 2 (output). Other input/output units may share these channels but console input/output units use only these channels.

Data may be transmitted between the console equipments and the computer in either the character or the assembly mode. In the character mode one 7-bit character is buffered at a time. The 7-bit character occupies the lowest bit positions of a 48-bit word; the upper 41 bits are "0's".

In the assembly mode the 48-bit word, consisting of eight 6-bit characters, is buffered. During an input buffer in the assembly mode eight successive characters are assembled into a 48-bit word and sent to the computer. The first character occupies the upper 6 bits of the word; the last character occupies the lower order 6 bits.

For an output buffer in the assembly mode a 48-bit word from the computer is disassembled into eight characters, the upper 6 bits first.

TYPEWRITER

The typewriter may be used as a keyboard input device or as an output device for producing printed copy; during output it types approximately 10 characters per second.

All of the typewriter characters and functions are represented by unique combinations of 6 bits. (Codes are in appendix 6.) During a keyboard input operation, striking a character key causes the coder to produce the code which is sent to the computer. Space is the only coded typewriter control function which is sent to the computer. For typewriter output, a 6-bit character code sent to the decoder causes the typewriter to print the selected character or perform the designated control function.

If the keyboard is selected by code 1114X, the interrupt signal occurs for each carriage return (CR).

If an illegal code (unlisted) is sent to the typewriter from the computer, operation hangs up. Striking the CR, backspace or shift keys will allow operation to be resumed.

A zero code (all "0" bits) which constitutes a do-nothing code is used to fill out a 48-bit word in the assembly mode.

PAPER TAPE READER

The CONTROL DATA 350 Paper Tape Reader enters information stored on punched paper tape into the computer. The reader, which is always connected to channel 1, operates at a maximum rate of 350 characters (frames) per second; the time interval between successive characters from the reader is 3.3 ms.

Manual controls for the reader are on the punch and reader control panel of the console. When the Reader Mode switch is set to ASSEMBLY, the tape is positioned at the first frame of the first word (load point); when it is set to CHARACTER, the tape moves ahead one frame.

Information is stored on paper tape in seven levels.* A frame, which is across the width of the tape, can store 7 bits (figure 3-2). The sprocket or feed holes between levels 3 and 4 generate signals to time and control the reading of the tape.

In the assembly mode, level 7 is used as a control rather than an information level. The first of the eight characters in a word is indicated by a hole in the control level.

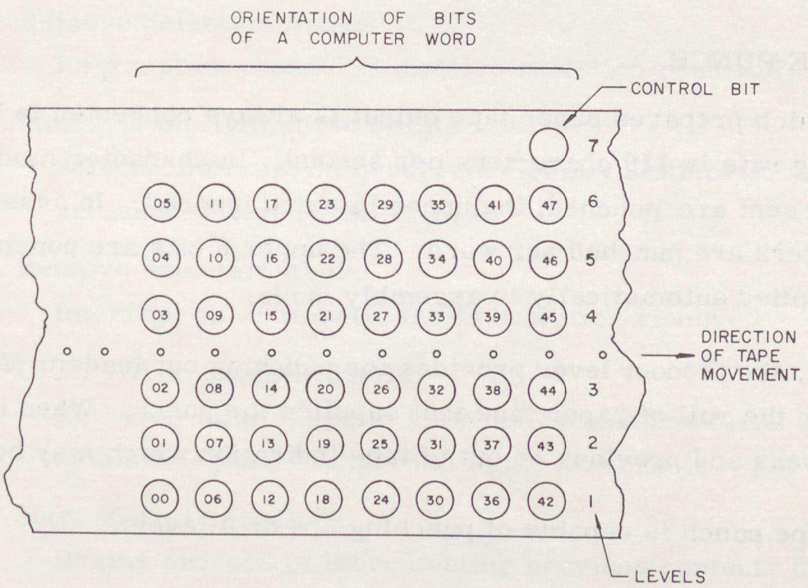


Figure 3-2. Seven-Level Punched Paper Tape

Reader tape motion stops on any one of three conditions:

1) When buffer operation terminates (assembly or character mode).
2) When the load point in assembly mode is reached.
3) Absence of a 7th level every 8th character in the assembly mode.

The reader End of Tape indicator is set on any of three conditions:

1) On a computer master clear.
2) Absence of a 7th level every 8th character in the assembly mode.
3) By a 74 0 11210 instruction. This instruction is used to indicate the end of information in the character mode.

---

*The CONTROL DATA 350 can also read 5- or 8- level tape.

After reading all information on the tape in the assembly mode, tape motion stops and the End of Tape indicator is set because the 7th level control bit is missing. In the character mode, however, motion stops when the buffer operation is satisfied but the End of Tape indicator remains cleared. A 74 0 11210 instruction may be programmed to set the End of Tape indicator after the buffer terminates. The state of the End of Tape indicator, regardless of the mode of operation, may be used to determine if all information on the paper tape has been read.

PAPER TAPE PUNCH

The punch which prepares paper tape output is always connected to buffer channel 2. The operating rate is 110 characters per second. In character mode, the lower 7 bits of each word sent are punched; the upper bits are ignored. In assembly mode, eight 6-bit characters are punched per word. The upper 6 bits are punched first, with the 7th level supplied automatically in assembly mode.

On the punch, the feedout lever provides for punching out leader. A microswitch is mounted near the roll of paper tape that supplies the punch. When the supply is low, the switch opens and provides an out of tape indication which may be sensed.

The paper tape punch is capable of punching 5, 6 or 8 levels.

INTERNAL EXF SELECT INSTRUCTIONS

74 0	000C0	Interrupt on Channel C Inactive Selects interrupt when channel C becomes inactive. C = 1 - 6 An interrupt signal is generated whenever the channel becomes inactive. More than one interrupt can be selected. The interrupt remains selected until cleared.
	000C1	Remove Selection Above Interrupt on channel C inactive selection removed
	00100	Interrupt on Arithmetic Faults Selects interrupt on occurrence of any arithmetic fault; remains selected until cleared
	00101	Remove Selection Above Interrupt on arithmetic faults selection removed
	00070	Clear All Arithmetic Faults Removes all arithmetic fault indications and turns off arithmetic fault background lights on console
	01000	Start Real-Time Clock Begins process of incrementing previous contents of address 00000 by one each 16.6 ms; address 00000 is not cleared by starting clock
	02000	Stop Real-Time Clock Halts process of incrementing address 00000. The contents of 00000 remain unchanged.
	C0000	Clear All Channel C Selections Clears all previous selections made on the designated channel C except interrupt on channel C inactive
	00200	Clear the Carriage-Return-Typed Indicator

INTERNAL EXF SENSE INSTRUCTIONS

74 7	000C0	Exit on Channel C Active Full exit if channel C is active; if not, half exit
	000C1	Exit on Channel C Inactive Full exit if channel C is inactive; if active, half exit
	00110	Exit on Divide Fault; half exit if no divide fault
	00111	Exit on No Divide Fault; half exit if divide fault
	00120	Exit on Shift Fault; half exit if no shift fault
	00121	Exit on No Shift Fault; half exit if shift fault
	00130	Exit on Overflow Fault; half exit if no overflow fault
	00131	Exit on No Overflow Fault; half exit if overflow fault
	00140	Exit on Exponent Fault; half exit if no exponent fault
	00141	Exit on No Exponent Fault; half exit if exponent fault
74 7	0C000	Exit on Channel C Interrupt
74 7	0C001	Exit on No Channel C Interrupt C = 1 = Channel 1 C = 2 = Channel 2 C = 3 = Channel 3 C = 4 = Channel 4 C = 5 = Channel 5 C = 6 = Channel 6
74 7	001T0	Exit on Channel T Interrupt
74 7	001T1	Exit on No Channel T Interrupt T = 6 = Channel 7 Output (odd) T = 7 = Channel 7 Input
74 7	00200	Exit on Lower — Monitor Interrupt Exit FF to determine whether last instruction in main program was lower or upper
74 7	00201	Exit on Upper — Monitor Interrupt Exit FF to determine whether last instruction in main program was lower or upper
74 7	00300	Exit on Clock Overflow
74 7	00301	Exit on No Clock Overflow

CONSOLE EXF SELECT CODES

(Always Channel 1 and 2)

INPUT

TYPEWRITER

74 0 11100 Select the Typewriter for Input and No Interrupt on Carriage Return

Selects keyboard (character mode only)

Interrupt selection cleared, Carriage Return indicator cleared

11140 Select the Typewriter for Input and Interrupt on Carriage Return

Selects keyboard (character mode only)

Interrupt selection set, Carriage Return indicator cleared. The next carriage return, which is not output, will set the Carriage Return FF and cause an interrupt. The interrupt selection can be cleared by the external master clear or the 74 0 11100 select only.

PAPER TAPE READER

74 0 11210 Set End of Tape Indicator

Sets the End of Tape indicator*

Clears interrupt on end of tape

11200 Select the Paper Tape Reader and No interrupt on End of Tape

Selects the reader

Interrupt on end of tape cleared

11220 Select the Paper Tape Reader and Interrupt on End of Tape

Selects the reader

Interrupt on end of tape set. If the End of Tape indicator is set, the interrupt will be immediate.

*This select is usually used in character mode operation only. The End of Tape indicates the logical end of tape, and can be cleared externally only by moving the switch (on the reader control) to the CHARACTER or ASSEMBLY position. Master clear selects the paper tape reader and sets the End of Tape indicator. When the End of Tape indicator has been set the reader is "not ready".

OUTPUT

TYPEWRITER

74 0	21100	Select the Typewriter for Output in the Assembly Mode Selects keyboard to print*
	21110	Select the Typewriter for Output in the Character Mode Selects keyboard to print*

PAPER TAPE PUNCH

74 0	21200	Select the Paper Tape Punch, Assembly Mode Selects the punch, sets mode to assembly Turns the punch motor on
	21210	Select the Paper Tape Punch, Character Mode Selects the punch, sets mode to character Turns the punch motor on
	21240	Turn the Punch Motor Off

CONSOLE EXF SENSE CODES

TYPEWRITER

(Sensed on Input Channel Only)

74 7	11100	Full Exit if Carriage Return Performed on Input If a carriage return (which was not the result of an output) has been performed since the last input select, a full exit is executed; if not, a half exit.
	11101	Full Exit if No Carriage Return Typed on Input If the Carriage Return indicator is not set, a full exit is executed; if set, a half exit.

*Will not change the Carriage Return FF nor the interrupt selection on channel 1. The code "00" will be ignored, all other illegal codes will cause the typewriter to hangup. It is released by manually performing a function, usually spacing.

11140 Full Exit Lower Case

If the typewriter keyboard is in the lower case a full exit is performed.

11141 Full Exit Upper Case

If the typewriter keyboard is in the upper case a full exit is performed.

PAPER TAPE READER

74 7 11200 Full Exit on End of Tape Indicator Set

If the End of Tape indicator is set a full exit is performed; if not, a half exit.

11201 Full Exit on No End of Tape Indicator Set

If the End of Tape indicator is not set a full exit is performed; if set, a half exit.

11210 Full Exit on Assembly Mode

If the paper tape reader is in the assembly mode a full exit is performed; if not, a half exit.

11211 Full Exit on Character Mode

If the paper tape reader is in the character mode a full exit is performed; if not, a half exit.

PAPER TAPE PUNCH

74 7 21200 Full Exit on Out of Tape

If the paper tape punch is out of tape, a full exit is performed; if not, a half exit.

21201 Full Exit on Not Out of Tape

If the paper tape punch is not out of tape, a full exit is performed; if out of tape, a half exit.

CHAPTER 4

OPERATION

DESCRIPTION OF INDICATORS AND CONTROL SWITCHES

All main computer controls and indicators are on the console. Functional significance of console background lights is listed in table 4-1; computer controls are described in table 4-2.

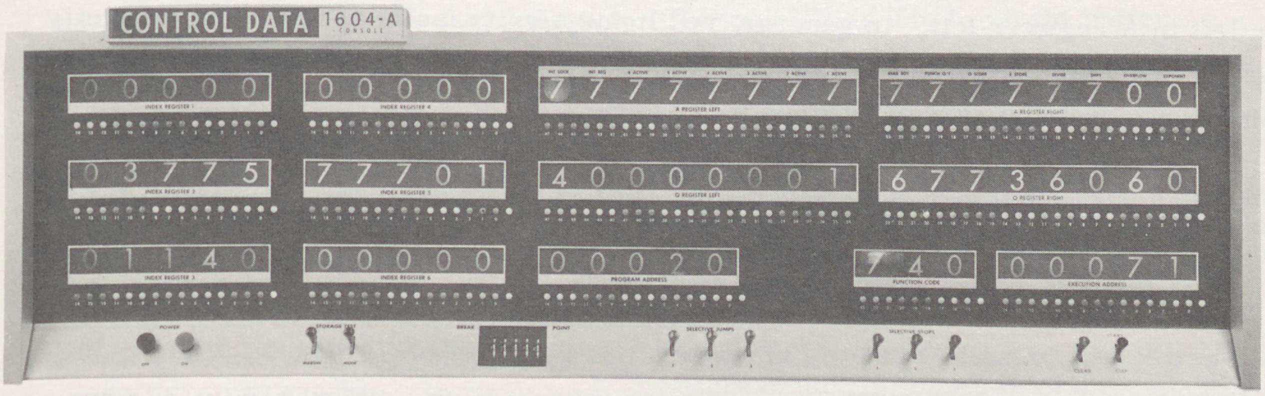


Figure 4-1. Center Panel of Console

The indicators are lamp modules, each of which displays a single octal digit. The lamps, in response to signals from the computer, display the contents of the operational registers in octal form only when the computer is stopped; the display is blank when the computer is running. Each indicator has three push buttons which are numbered in the powers of two, from right to left, starting with zero. Pressing a push button forces that particular stage of the register to the SET state. Each group of three buttons represents an octal digit. To aid in distinguishing between octal digits, the buttons for adjacent octal digits are different shades of blue.

At the right end of each register is a Clear push button (white). This button will clear all the FFs within that register. Set and Clear push buttons should be used only when the computer is stopped; otherwise errors may result.

Conditions which stop the computer are listed below. When these conditions exist register contents may be altered by setting or clearing.

1) Illegal function codes 00 and 77
2) Selective Stops (instruction 76)
3) Breakpoint Stop
4) Pressing Start/Step switch
5) Pressing Clear switch (internal master clear)

At some of the modules there are colored background lights which indicate certain internal conditions (figure 4-2, table 4-1). A light is identified by the register in which it is located and its position in the register. For example, AL-4 is fourth from the left in A register left. In general, red lights signify faults and blue lights signify special operating conditions. The background lights may be illuminated when the computer is running as well as when it is stopped.

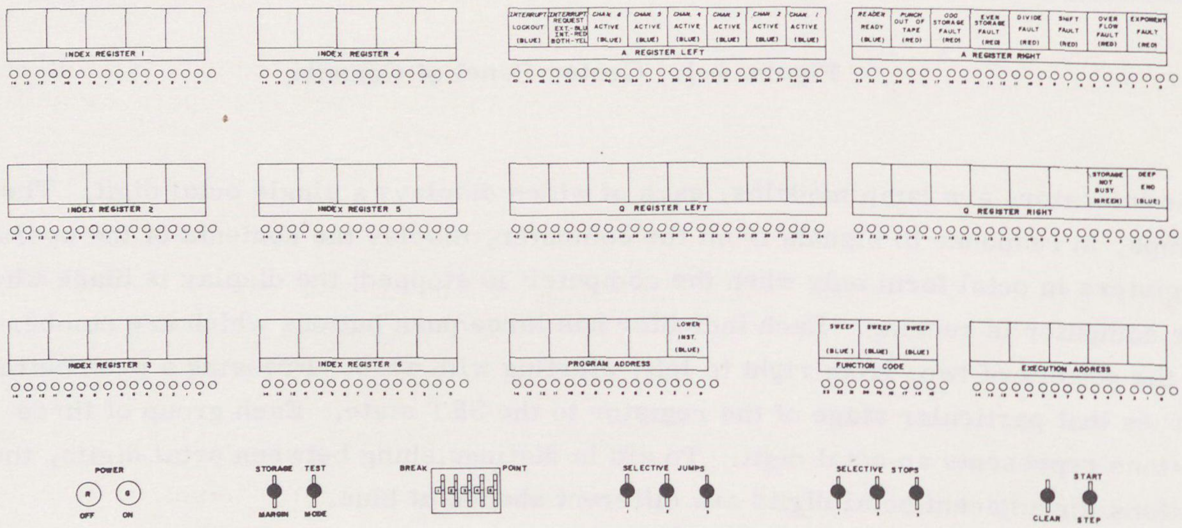


Figure 4-2. Console Display

TABLE 4-1. CONDITIONS INDICATED BY CONSOLE BACKGROUND LIGHTS

Light	Condition
AL-1 (blue)	Interrupt Lockout - Computer is in interrupt routine.
AL-2 (red) *	Internal Interrupt Request — Interrupt request signal is being received from interrupt circuit.
AL-2 (blue)	External Interrupt Request — Interrupt request signal is being received from interrupt circuit.
AL-3 (blue)	Channel 6 Active - Channel 6 is in use for output buffer.
AL-4 (blue)	Channel 5 Active - Channel 5 is in use for input buffer.
AL-5 (blue)	Channel 4 Active - Channel 4 is in use for output buffer.
AL-6 (blue)	Channel 3 Active - Channel 3 is in use for input buffer.
AL-7 (blue)	Channel 2 Active - Channel 2 is in use for output buffer.
AL-8 (blue)	Channel 1 Active - Channel 1 is in use for input buffer.
AR-1 (blue)	Reader Ready - (1) Paper tape is at load point, ready for an input buffer; or (2) input buffer paper tape is in progress.
AR-2 (red)	Punch Out of Tape - Punch tape reel is nearly empty.
AR-3 (red)	Odd Storage Fault - Fault in sequence chain of odd storage unit; storage unit is inoperative until master cleared.
AR-4 (red)	Even Storage Fault - Fault in sequence chain of even storage unit; storage unit is inoperative until master cleared.
AR-5 (red)	Divide Fault - Improper divide instruction executed.
AR-6 (red)	Shift Fault - Shift count greater than 127 (decimal).
AR-7 (red)	Overflow Fault - Required sum or difference exceeds capacity of A register.
AR-8 (red)	Exponent Fault - In a floating-point instruction, exponent of result is 2^{10} or greater.
QR-8 (blue)	Deep End - Computer fails to complete operation in step mode.
PA-5 (blue)	Lower Instruction - Lower instruction is indicated.
FUNCTION CODE (blue) (3 lights)	Sweep - Computer is in sweep mode (Mode switch is down).

*On both internal and external interrupt requests the light is yellow.

TABLE 4-2. MAIN COMPUTER CONTROLS

Control		Function
Power	ON - green	Applies a-c and d-c power to computer by energizing contactor in primary power lines of motor-generator.
push button	OFF - red	Removes d-c and a-c power from computer by de-energizing contactor in primary power lines of motor-generator.
Storage Test	MARGIN	Varies the bias applied to storage sense amplifiers. Used for maintenance purposes only; should be in neutral position at all other times.
Lever switch locks in up, down and neutral positions.	MODE	Up: an instruction is executed repeatedly in either the step or start mode. Down: contents of consecutive storage locations may be manually examined by pressing Step. Consecutive half-words are displayed in function code and execution address registers but are not executed.
Breakpoint Five 8-position switches can be set to octal address 00000 through 77777.		Provides for selection of any storage address as a breakpoint address. Computer stops when program address and breakpoint address are equal, just prior to performing the upper instruction at the breakpoint address.
Selective Jumps 1, 2, 3 Three lever switches lock in upper positions, momentary in down positions.		Provide manual conditions for instruction 75, normal jumps, b = 1, 2 or 3, return jumps, b = 5, 6 or 7.
Selective Stops 1, 2, 3 Three lever switches lock in upper position, momentary in down positions.		Provide manual conditions for stopping the computer on instruction 76, b = 1, 2, 3, 5, 6 or 7.
Clear Lever switch, momentary in up and down positions.		Up: master clears external equipment, causing most of the registers and control FFs of the external equipment to be cleared and the paper tape reader to be selected. Down: master clears the computer, clears all operational registers and most control FFs.

TABLE 4-2. MAIN COMPUTER CONTROLS (CONT'D)

Control	Function
Start/Step Lever switch, momentary in up and down positions.	START (up) selects high-speed mode in which a program of instructions and auxiliary operations proceeds until completed or stopped. STEP (down) selects step mode. Each time switch is pressed a single instruction is executed and computer stops (all buffer requests are completed before operation stops). Step selection overrides any previous selection of start.
Volume Control Black knob under console desk	Controls volume of signal from console loud-speaker.
*The Set push buttons, numbered in the powers of 2, beginning with zero. Each group of three is an octal digit.	Allow for manual entry of a quantity into a given register. Forces that particular stage of register to the set state.
*The Clear push button	Clears all FFs within that register.

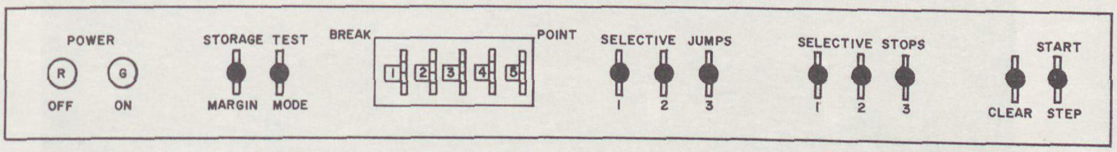


Figure 4-3. Manual Controls

*Should be used only when the computer is stopped.

TABLE 4-3. READER AND PUNCH CONTROLS

Switch	Function
Punch Motor	Turns punch motor on or off. (Motor may also be turned on under program control.)
Select/Tape Feed	Select enables use of the punch. Tape Feed causes leader to be punched.
Reader Motor	Turns reader motor on or off. (Motor cannot be turned on by any other means.)
Character/Assembly	In character mode each character is sent to computer separately. In assembly mode eight consecutive characters are assembled into a word to be sent to computer.

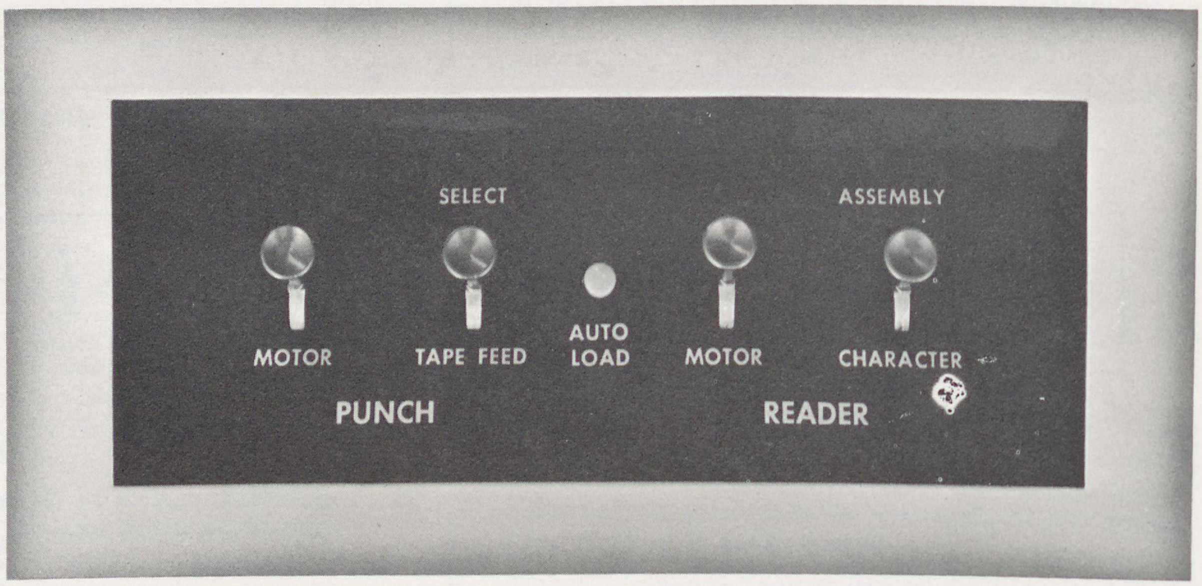


Figure 4-4. Reader, Punch, and Auto Load Controls

AUTO LOAD CONTROL

The Auto Load button initiates a bootstrap routine to read into memory the first record from magnetic tape #1, on 1615 or 1607 equipment #2, on channel 3.

Pressing the Auto Load button selects the tape and loads the bootstrap routine into memory locations 00000 and 00001, and puts an address of 32000 (arbitrary and > 00004) in the lower address of 00003.

The program appears as:

(00000)	74 0 32005	Rewind the tape
	74 7 32000	Wait for ready
(00001)	74 3 00002	Activate, FWA = 00002
	74 7 32000	Wait for ready
(00002)	XX X XXXXX	Will be the first
	XX X XXXXX	word read from tape
(00003)	74 3 00002	
	74 7 32000	

If Breakpoint is not set to 00000 or 00001 the routine will be executed. The first word read from tape will be read into location 00002 and be executed as soon as the tape is ready again.

Location 00003 will become XXX00004XXXYYYYY from the second word on tape, inserting a new terminating address.

OPERATION

The 1604-A is a stored-program computer. To load a program in the computer a load program (basic service library) is needed. The load program is entered manually. A paper tape reader, a paper tape punch, an electric typewriter, and a set of magnetic tapes are some of the important external devices used for communicating with the 1604-A. The programmer, before operating any of these devices, should make himself familiar with instructions for these devices and they should be followed in the order recommended.

LOAD PROGRAM ENTERING

A load program to be entered in storage is usually on bi-octal paper tape. The following procedure enters the load program:

1) Turn on power.
2) Master clear, both internal and external.
3) Press Start/Step switch once.
4) Clear function code and set to 200.
5) Clear execution address and set to 00001.
6) Set terminal address of buffer in lowest five octal digits of A register right.
7) Press Start/Step switch once.
8) Load tape into reader.
9) Turn on reader motor (wait 10 seconds).
10) Raise reader Mode switch to ASSEMBLY position.
11) Clear function code and set to 741.
12) Clear execution address and set to initial address of buffer.
13) Press Start/Step switch once. Wait until tape loads (console lights come on).
14) Press Clear switch.
15) Perform steps 2 through 8 of operation with pre-stored program.

STARTING OPERATION WITH PRE-STORED LOAD PROGRAM

When a general loading program which provides for loading other programs is held in storage, the starting procedure is as follows:

1) Turn on power (Power On, figure 4-3).
2) Make required manual selections:

Selective Jumps
Selective Stops
Breakpoint

3) Set in operation the external device or devices selected to communicate with the computer. (Follow the instructions for the devices given in this chapter.)

4) Master clear, both internal and external (press clear, then raise it).

5) Set Program Address register to address of first instruction of program.

6) Begin computer operation (set Start switch).

7) To shut down the equipment after the operation has stopped, follow the instructions as given for each external device.

8) Press Power Off button, which disconnects power from all equipments.

READER

The reader is a CONTROL DATA 350 paper tape reader (figure 4-5). It can read either a 5-, a 7-, or an 8-level tape. For a bi-octal tape with the 7th level control holes, assembly mode is selected; for a flex or other code, character mode is selected.

1) Check if tape basket is at the proper place. Do not allow the tape to fall on the floor.

2) Turn tape release lever clockwise to raise tape guide plate.

3) Select the desired tape level by means of the tape level switch.

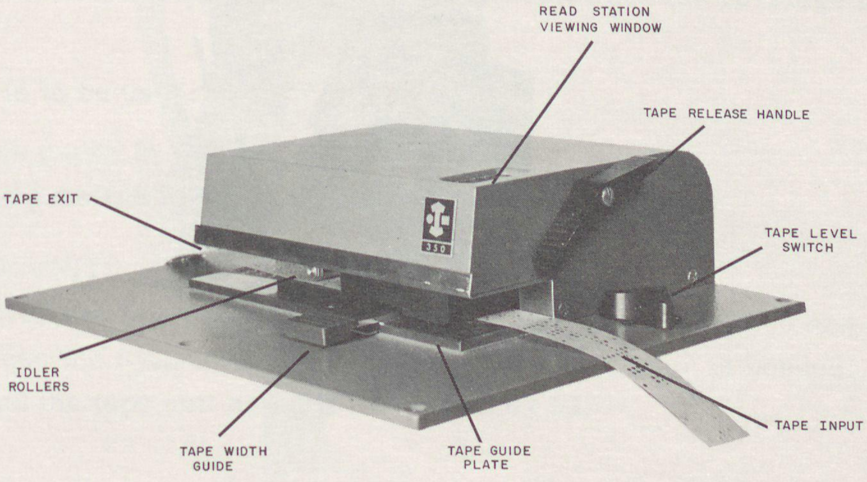


Figure 4-5. Paper Tape Reader

4) Holding the tape guide down, slide it so that the marker rests above the proper etched mark on the tape deck surface. The outer position is for 8-level tape, the center for 7-level, and the inner for 5-level.

5) Insert tape as shown in figure 4-6. Make sure that the tape is properly aligned.

6) Turn Tape Release lever counterclockwise to lower the tape guide.

7) Select the desired mode of operation by the Mode switch (figure 4-4) on the computer panel.

8) Turn on Reader switch on computer console (figure 4-4).

9) After the reader has read the tape, remove paper tape from reader and baskets; rewind tapes.

10) Turn off reader motor.

PUNCH

The paper tape punch (figure 4-6) is mounted on a hinged rack at the rear of the right wing of the console. Punch tape feeds out of a slot in the compartment door; the chad box is just inside the door.

1) To ensure proper performance of the punch, always keep the chad box clean.

2) Set Punch switch to SELECT (computer console) and check for sufficient paper in reel.

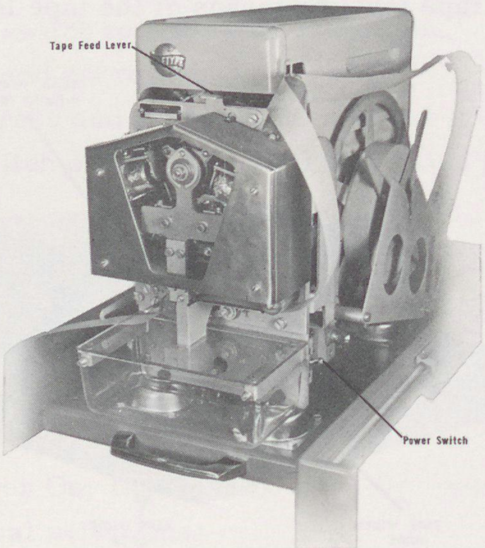


Figure 4-6. Paper Tape Punch

3) If you have used the punch, generate a foot of leader by pressing tape feed; remove feed; remove tape and wind it up.

4) Perform the following steps to replace a tape roll at punch.

   a) Remove the tape reel from cradle at side of punch.

   b) Unscrew tape hold-down assembly, remove old roll, and place new roll on reel. Replace hold-down assembly and mount reel in cradle.

   c) Thread tape as shown in figure 4-6. Bring tape around lower roller and into guides leading to punch block.

   d) Turn on punch motor and advance tape through the punch block by pressing the tape feed-out lever (top of punch block).

   e) Bring leader out through slot in door. Swing punch back into compartment.

TYPEWRITER

The typewriter has all of the characters and functions of a standard electric machine. As a keyboard entry device the typewriter is used only in the character mode. After the program selects keyboard and initiates an input buffer, each striking of a key causes a 6-bit coded character to be entered into the lower six positions of a computer word. The remaining bits of the word are all "0". If the keyboard is selected along with an interrupt feature, each carriage return or tab sends an interrupt signal to the computer. This notifies the program of the entry of data from the keyboard.

When the typewriter is used as an output device certain conditions cause it to hang up until the space bar is struck: receipt of an illegal typewriter code, a code to shift up when the carriage is already up, or a code to shift down when the carriage is already down.

If the typewriter is to be used:

a) Place paper in it.

b) Set the switch beneath the righthand corner to ON.

MAGNETIC TAPE UNITS

The tape units which can be used with 1604-A are CONTROL DATA 606 and CONTROL DATA 1607. To use the 606, the CONTROL DATA 1615 Adapter is needed. The codes for the adapter and the tape unit are given in appendix VI.

606 TAPE UNIT

Controls and Indicators

The manual controls and indicators for operating each tape unit are mounted on a panel located below the front door of the unit (figure 4-7). The functions of the controls are described in table 4-4.

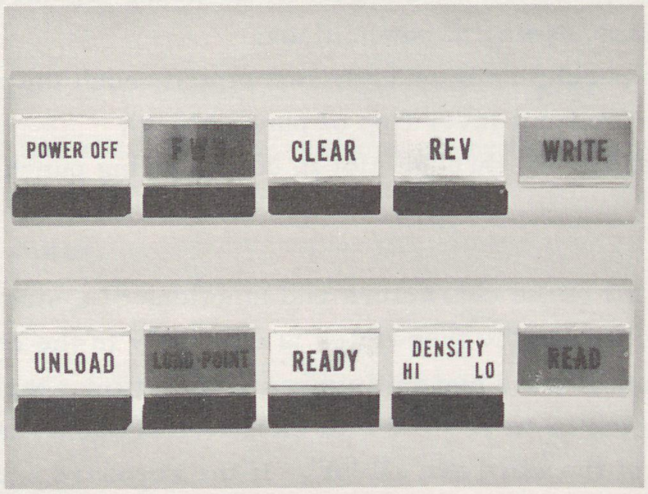


Figure 4-7. Operator Control Panel

TABLE 4-4. 606 CONTROLS AND INDICATORS

NAME		FUNCTION
POWER OFF	*S	Removes power from all components and power supplies.
	**I	Power is available to components and power supplies.
FWD	S	Moves tape forward at 150 ips. Motion stops when end of tape marker is sensed.
	I	Tape is moving forward at 150 ips.
CLEAR	S	Master clears all previous settings and conditions. Stops tape motion immediately. New Manual selections are necessary to reselect tape unit and/or operation required.
	I	606 is cleared

*Switch
**Indicator

TABLE 4-4. 606 CONTROLS AND INDICATORS (CONT'D)

NAME		FUNCTION
REV	*S	Rewinds tape at 225 ips. Motion stops when load point marker is sensed.
	**I	Tape is moving in reverse direction at 150 or 225 ips.
WRITE	I	Write operation is in progress.
UNLOAD	S	Moves tape at 225 ips to unload position (all tape on supply reel). Tape load procedure must be performed to resume operation.
	I	Tape is in unload status.
LOAD POINT	S	Moves tape forward at 150 ips to load point marker. Motion stops when marker is sensed.
	I	Tape is at load point marker.
READY	S	Places 606 under external control.
	I	Unit is under external control.
DENSITY	S	Changes density mode selection.
	I (Hi) I (Low)	High density mode selected. Low density mode selected.
READ	I	Read operation is in progress (not on when reading for horizontal checking during write operation).
UNIT SELECTION	S	10-position switch; 0-7 provide input designation while two standby positions disconnect unit from external control.
	I (White) I (Red)	Show selected number. Fault Condition (power failure, tape not in columns, etc.).
OVERHEAD LIGHTS	I	File protection ring is on reel (unit can write) and tape unit is not in the unload position.

* Switch
** Indicator

Tape Load Procedure

1) Make sure that tape unit is properly energized.

2) Slide front glass door down to lowest position (figure 4-8).

3) Check that supply reel has been file protected as necessary.

4) Mount reel on supply reel hub and tighten hub knob. For proper alignment, push reel firmly against hub stop before tightening knob.

5) Make sure that tape load arms are in up position.

6) Pull sufficient tape from supply reel to reach take-up reel. Thread tape on the outside of the supply tape load arm, over the head assembly, around the outside of the take-up load arm and over the top of the take-up reel hub two or three times.

7) Slide tape under head assembly.

8) Snap tape load arms down.

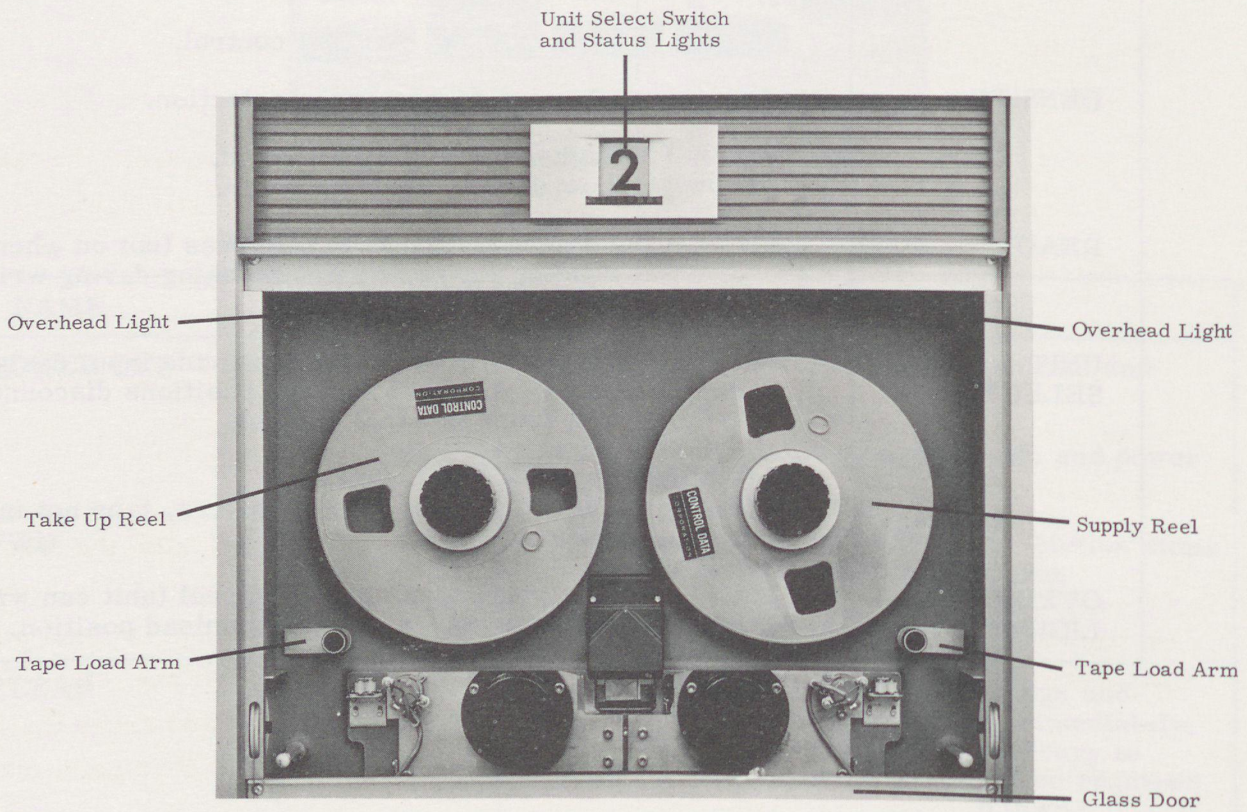


Figure 4-8. 606 Tape Load and Unload Mechanics

9) Set Unit Selection switch to one of ten positions (0-7 or standby) to assign a logical program selection number.

10) Press Clear switch.

11) Press Load Point switch. Tape will drop in columns, move forward, and stop on load point marker. The Load Point light will turn on. (If the light does not turn on, notify maintenance.) If tape continues moving forward for more than 3 or 4 seconds, it indicates either no load point marker was placed on the tape or the operator manually wound the marker onto the take-up reel during step 5.

12) If the unit is to be controlled, press the Ready switch. If it is to be manually operated and the Ready switch has been pushed, press the Clear switch.

13) Raise the front glass door completely.

If the supply reel contains a file protection ring, the overhead lights should be on, indicating that a write operation may be performed. If the lights are not on, notify maintenance.

Tape Unload Procedure

1) Press Clear switch.

2) Press Unload switch. All tape will automatically be drawn from the take-up reel and wound on the supply reel. The Unload indicator will light.

3) Slide down front door.

4) Loosen supply reel hub knob and remove supply reel.

5) Check if reel needs to be file protected and if it is labeled adequately prior to storage.

Special Instructions

In order to simulate an unload condition without removing all tape from the take-up reel, simultaneously press the Clear and Unload switches. The unload condition will be simulated but tape will not move. In order to place the unit in operational status, remove all tape from the vacuum columns by revolving the take-up reel clockwise and the supply reel counterclockwise. Snap the tape load arms down and press the Load Point switch. The tape will move forward and stop on the nearest load point marker. The Load Point indicator will turn on.

If all tape is unwound from the supply reel:

1) Snap tape load arms up, if necessary.

2) Guide tape around the tape load arms, over the head assembly, and wrap approximately ten turns around the supply reel.

3) Slide tape under head assembly.

4) Press the Load Point switch.

5) As soon as the Forward light turns on, press the Clear switch and then the Reverse switch. Tape will rewind on the nearest load point marker.

The following information is applicable when a number of load point or end of tape markers are used on a single tape.

To move forward from a reflective marker and stop at nearest end of tape marker, press the Forward switch.

To move forward off a reflective marker and stop at nearest load point or end of tape marker, press the Forward and then the Load Point switches. Load Point indicator will light if motion stops at load point marker.

To reverse from a reflection marker and stop at nearest load point marker, press the Unload, Clear, and Reverse switches, in that order.

Tape motion may be stopped at any time by pressing the Clear switch. An unload operation may be performed by pressing the Unload switch.

1607 TAPE UNIT

<u>Controls and Indicators</u>

Each tape unit is provided with push buttons for manual operation. These controls are mounted on a panel above the front door (figure 4-9, table 4-5).

<u>Tape Load Procedure</u>

1) Open door to handler.

2) Check that file reel to be loaded has been file protected as necessary.

3) Mount the reel on the file reel hub and tighten the hub knob. To insure proper reel alignment push the reel firmly against the reel hub stop before tightening the knob. If the file protection ring has been removed from the reel, check that the Write Lockout lamp turns on when the reel is loaded. If the lamp does not turn on call maintenance.

TABLE 4-5. 1607 CONTROLS AND INDICATORS

Control		Function
REWIND	*S	Controls manual rewind to load point.
	**I	Indicates rewind in progress.
CHANGE TAPE	S	Drops any manual selection and places tape unit in automatic or program control mode.
	I	When lighted, indicates tape rewound under program control and interlocked at load point. The interlock prevents operation of the tape unit until the Stop Manual switch is operated.
WRITE LOCKOUT	S	Drops power from unit and removes program designation.
	I	When lighted, indicates that tape unit is loaded with a reel which does not contain a file protection ring. The tape cannot be written as long as the light is on, but may be read.
1, 2, 3 or 4	S	Designates program selection of unit and applies power to unit. Each new unit designation cancels an existing designation.
	I	Indicates unit selection and power-on condition.
REVERSE	S	Initiates reverse tape motion during manual operation.
	I	Indicates reverse tape motion.
STOP MANUAL	S	Drops unit from program control or drops forward or reverse selection and places unit in manual mode.
	I	Indicates manual mode.
FORWARD	S	Indicates forward tape motion during manual mode.
	I	Indicates forward tape motion.

*switch
**indicator

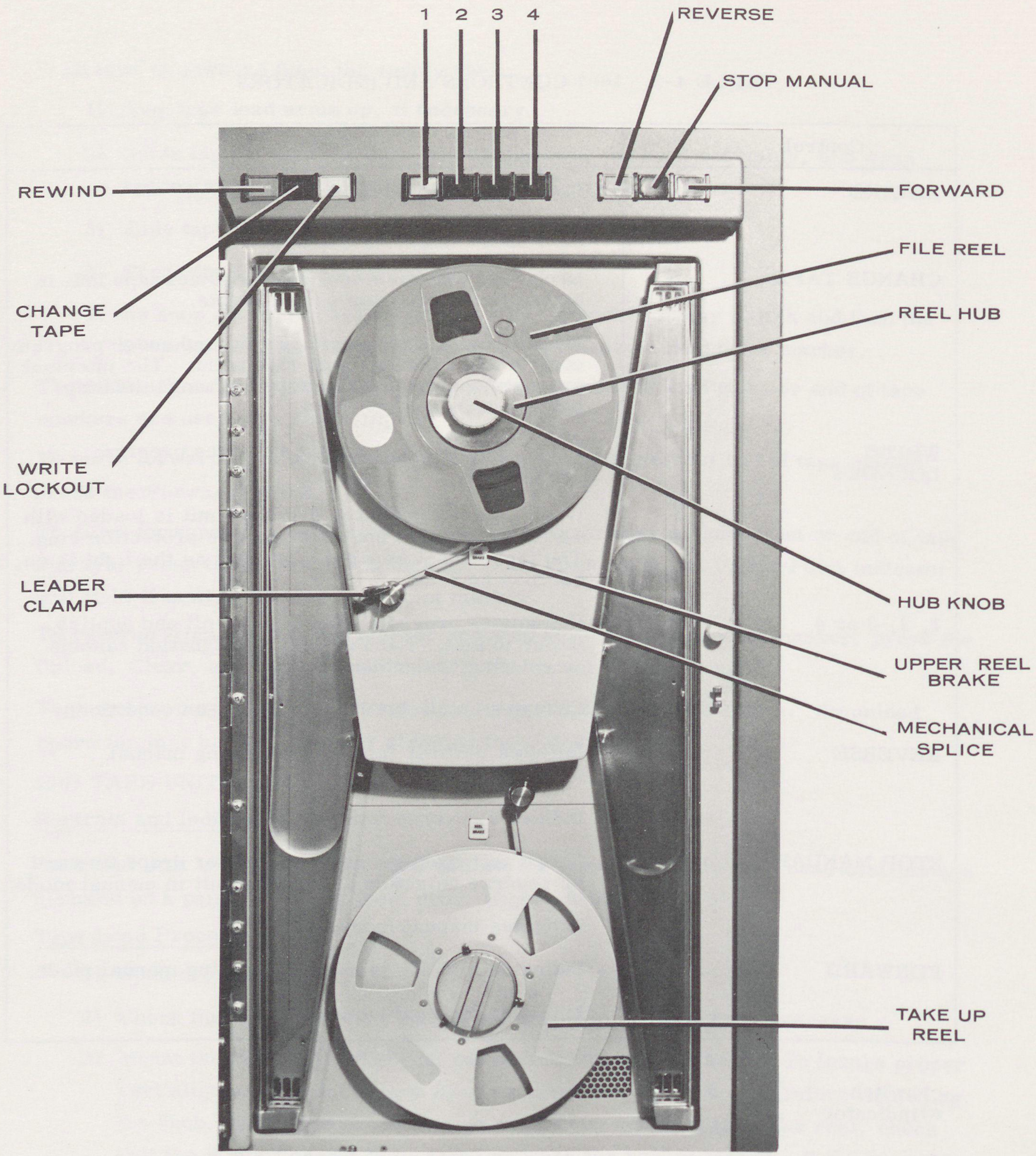


Figure 4-9. 1607 Tape Unit

4) Press upper Reel Brake pushbutton to release mechanical brake and check that pulling tape from reel causes it to rotate clockwise. Pull sufficient tape from reel to reach end of permanent machine leader held by leader clamp.

5) Connect file tab to permanent machine leader.

6) Take up slack by turning file reel while pressing upper Reel Brake push button.

7) Lift leader clamp and close door.

8) Press one of the unit selection switches (1, 2, 3, 4) to apply power to the unit and assign the unit a logical program selection number. Wait two minutes. The Stop Manual lamp should turn on; if not, call maintenance.

9) Press Stop Manual.

10) Press Rewind button. Unit is ready when Rewind lamp turns off. If Stop Manual lamp remains on, unit is not ready; call maintenance.

Tape Unload Procedure

1) Press Stop Manual button to select manual mode.

2) Press Reverse button to move tape backwards to change tape position.

3) Open front door of tape unit.

4) To secure tape, lower leader clamp.

5) Press the upper Reel Brake button to release the mechanical brake and pull tape from file reel to provide slack.

6) Unfasten mechanical splice which connects the file tab to the permanent machine leader.

7) Loosen file reel hub knob and remove the file reel.

8) Check if reel needs to be file protected and also if it is labelled adequately prior to storage.

FILE PROTECTION RING

The back of the file reel has a slot near the hub which accepts a plastic file protection ring (figure 4-10). Writing on a tape is possible only when the reel contains a file protection ring. The ring should be removed from the reel after writing is completed to avoid accidental rewriting. Tape may be read either with the ring in place or without it. On the 606 the overhead lights go on immediately after the tape load procedure is

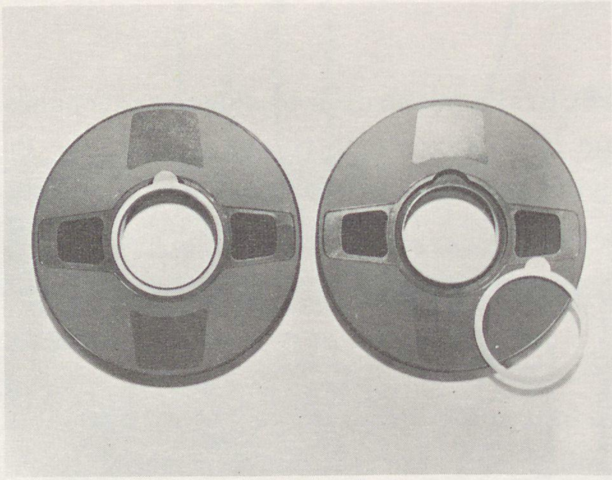


Figure 4-10. File Protection Ring

executed if the file protection ring is in place. The Write Lockout light on the 1607 is off if the file protection ring is in place.

EMERGENCY PROCEDURES

A fault indication, or a warning signal from the buzzer, may call for special procedures on the part of the operator.

TABLE 4-5. EMERGENCY PROCEDURE

Condition	Procedure
Punch out of tape	Load new roll of tape in punch at end of current operation.
Odd Storage Fault	Master clear. Restart program.
Even Storage Fault	Master clear. Restart program.
Deep End	Restart operation. If unable to proceed, master clear and restart program. If condition persists, notify maintenance.
Sweep	Place Mode switch in neutral position.
Buzzer Signal	Notify maintenance engineer immediately.

Faults for which the program provides corrective action are: Divide, Shift, Overflow and Exponent Faults. (Refer to appendix.)

GLOSSARY

ABSOLUTE ADDRESS	A specific storage location; contrast with relative address.
ACCESS TIME	The time needed to perform a storage reference, either read or write. In effect, the access time of a computer is one storage reference cycle.
ACCUMULATOR	A register with provisions for the addition of another quantity to its content. It is also the name of the A register.
ADDER	A device capable of forming the sum of two or more quantities.
ADDRESS	A 15-bit quantity which identifies a particular storage location.
ALPHABETIC CODING	A system of abbreviation used in preparing information for input into a computer, e.g., Q Right Shift would be QRS.
AND FUNCTION	A logical function in Boolean algebra that is satisifed (has the value "1") only when all of its terms are "1's". For any other combination of values it is not satisfied and its value is "0".
A REGISTER	Principal arithmetic register; operates as a 48-bit subtractive accumulator (modulus $2^{48}-1$).
BASE	A quantity which defines some system of representing numbers by positional notation; radix.
BIT	Binary digit, either "1" or "0".
BLOCK	A group of words transported in and out of storage as a unit.
BOOTSTRAP	The coded instructions at the beginning of an input tape, together with the manually entered instructions.
BORROW	In a subtractive counter or accumulator, a signal indicating that in stage n, a "1" was subtracted from a "0". The signal is sent to stage n+1 which it complements.
BRANCH	A conditional jump.
BREAKPOINT	A point in a routine at which the computer may be stopped by a manual switch for a visual check of progress.
B^1 - B^6 REGISTERS	Index registers used primarily for modification of execution address.

BUFFER — A device in which data is stored temporarily in the course of transmission from one point to another. To store data temporarily. The operation in which either a word from storage is sent to an external equipment via an output channel (output buffer), or a word is sent from an external equipment to storage via an input channel (input buffer).

CAPACITY — The upper and lower limits of the numbers which may be processed in a register or the quantity of information which may be stored in a storage unit. If the capacity of a register is exceeded, an overflow is generated.

CARRY — In an additive counter or accumulator, a signal indicating that in stage n, a "1" was added to a "1". The signal is sent to stage n+1, which it complements.

CHANNEL — A transmission path that connects the computer to an external equipment.

CHARACTER — Two types of information handled by the computer:

1) A group of 6 bits which represents a digit, letter or symbol. In the assembly mode, eight 6-bit characters make up a computer word.

2) A group of 7 bits which represents an item of information. In the character mode, this item is one 7-bit character and "0's" in the remaining (upper) 41 bits.

CLEAR — A command that removes a quantity from a register by placing every stage of the register in the "0" state.

CLOCK OVERFLOW — A clock overflow occurs whenever the capacity of the A register is exceeded during an advance clock instruction. This condition is indicated by a visible display and can be sensed by an EXF code.

CLOCK PHASE — One of two outputs from the master clock, "even" or "odd".

COMMAND — A signal that performs a unit operation, such as transmitting the content of one register to another, shifting a register one place to the left or setting a FF.

COMMON CONTROL REGISTER	A 30-bit register used to hold the initial and terminal addresses (CR_U and CR_L) of the current buffer operation while the comparator samples them. The CCR also has counting logic which is used to advance the address from CR_U.
COMPILER	A routine which automatically produces a specific program for a particular problem. The routine determines the meaning for information expressed in a psuedo-code, selects or generates the required subroutine, transforms the subroutine into specific coding, assigns storage registers, and enters the information as an element of the problem program.
COMPLEMENT	Noun: see One's Complement or Two's Complement. Verb: a command which produces the one's complement of a given quantity.
CONTENT	The quantity or word held in a register or storage location.
CONTROL REGISTERS 1-6	30-bit registers used to hold the address portions of the buffer control words. The upper address portion (CR_U) is advanced each time a word is buffered and is the current address for a buffer operation.
CORE	A ferromagnetic toroid used as the bistable device for storing a bit in a memory plane.
COUNTER	A register with provisions for increasing or decreasing its content by 1.
EVEN STORAGE	The storage unit which contains the 16,384 even addresses.
EXECUTION ADDRESS	The lower 15 bits of a 24-bit instruction. Most often used to specify the storage address of an operand. Sometimes used as the operand.
EXIT	Initiation of a second control sequence by the first, occurring when the first is near completion; the circuit involved in exiting.
EXTERNAL FUNCTION	1) External Function Select (74.0) sends a code to an external equipment to direct its operation. 2) External Function Sense (74.7) sends a code to an external equipment to sense its operating condition.

FAULT	Operational difficulty which stops operation or sets an indicator.
FIXED POINT	A notation or system of arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the binary point implicitly located at some predetermined position; contrasted with floating point.
FLIP-FLOP (FF)	A bistable storage device. A "1" input to the set side puts the FF in the "1" state; a "1" input to the clear side puts the FF in the "0" state. The FF remains in a state indicative of its last "1" input. A stage of a register consists of a FF.
FLOATING POINT	A means of expressing a number X by a pair of numbers, Y and Z, such that $X = Yn^{Z}$. Z is an integer, called the exponent or characteristic; n is a base, usually 2 or 10; and Y is called the fraction or mantissa.
FUNCTION CODE	The upper 9 bits of a 24-bit instruction consisting of the operation and index codes.
INDEX CODE	A 3-bit quantity, bits 15, 16, and 17 of an instruction; usually specifies an index register whose contents are added to the execution address; sometimes specifies the conditions for executing the instruction.
INSTRUCTION	A 24-bit quantity consisting of a function code, execution address, and index designator.
INTERRUPT MASKING REGISTER (IMR)	Consists of eight FFs which are set or cleared by EXF select codes to apply a mask to the interrupt lines. If one of these FFs is set it disallows the corresponding external interrupt.
INTERRUPT REQUEST	A signal received from an external equipment or internal logic that may cause a special sequence of instructions to be executed.
INVERTER	A circuit which provides as an output a signal that is opposite to its input. An inverter output is "1" only if all the separate OR inputs are "0".
JUMP	An instruction which alters the normal sequence control of the computer and, conditionally or unconditionally, specifies the location of the next instruction.

LOAD	To place a quantity from storage in a register.
LOCATION	A storage position holding one computer word, usually designated by a specific address.
LOGICAL PRODUCT	In Boolean algebra, the AND function of several terms. The product is "1" only when all the terms are "1"; otherwise it is "0". Sometimes referred to as the result of "bit-by-bit" multiplication.
LOGICAL SUM	In Boolean algebra, the OR function of several terms. The sum is "1" when any or all of the terms are "1"; it is "0" only when all are "0".
LOOP	Repetition of a group of instructions in a routine.
LOWER ADDRESS	The execution address portion of a lower instruction; bits 0 through 14 of a 48-bit register or storage location.
LOWER INSTRUCTION	See Program Step.
MASK	In some instructions, one quantity may determine what part of the other quantity is to be considered. If the first quantity, the mask, contains a "1", the corresponding bit of the second quantity is considered.
MASKED INTERRUPT REGISTER (MIR)	A rank of eight FFs through which external interrupt signals enter the 1604-A. The inputs to MIR can be masked (disallowed) by the IMR.
MASTER CLOCK	The source of standard signals required for sequencing computer operation. The clock determines the basic frequency of the computer.
MASTER CLEAR (MC)	A general command produced by placing the Clear switch up (external MC) or down (computer MC) which clears all the crucial registers and control FFs.
MNEMONIC CODE	A three-letter code which represents the function or purpose of an instruction. Also called Alphabetic Code.

MODULUS	An integer which describes certain arithmetic characteristics of registers, especially counters and accumulators, within a digital computer. The modulus of a device is defined by r^n for an open-ended device and r^n-1 for a closed (end-around) device, where r is the base of the number system used and n is the number of digit positions (stages) in the device. Generally, devices with modulus r^n use two's complement arithmetic; devices with modulus r^n-1 use one's complement.
NORMALIZE	To adjust the exponent and mantissa of a floating-point result so that the mantissa lies in the prescribed standard (normal) range.
NORMAL JUMP	An instruction that jumps from one sequence of instructions to a second, and makes no preparation for returning to the first sequence.
NUMERIC CODING	A system of abbreviation in which all information is reduced to numerical quantities.
ODD STORAGE	The storage unit which contains the 16,384 odd addresses.
ONE'S COMPLEMENT	With reference to a binary number, that number which results from subtracting each bit of the given number from "1". The one's complement of a number is formed by complementing each bit of it individually, that is, changing a "1" to "0" and a "0" to a "1". A negative number is expressed by the one's complement of the corresponding positive number.
ON-LINE OPERATION	A type of system application in which the input data to the system is fed directly from the external equipment to the computer.
OPERAND	Usually refers to the quantity specified by the execution address. This quantity is operated upon in the execution of the instruction.
OPERATIONAL REGISTERS	Registers which are displayed on the operator's console (B^1-B^6, A, Q, P, U^1).
OPERATION CODE	The upper 6 bits of a 24-bit instruction which identify the instruction. After the code is translated, it conditions the computer for execution of the specified instruction. This code, which is expressed by two octal digits, is designated by the letter f.

O^1 - O^4 REGISTERS	Output registers $O^{1,2,3}$ are used for output buffer operations; O^4 handles all high-speed output transfer operations.
OR FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") when any of its terms are "1". It is not satisfied when all terms are "0". Often called the 'inclusive' OR function.
OVERFLOW	The capacity of a register is exceeded.
PARITY CHECK	A summation check in which the binary digits in a character are added and the sum checked against a previously computed parity digit; i.e., a check which tests whether the number of ones is odd or even.
PARTIAL ADD	An addition without carries. Accomplished by toggling each bit of the augend where the corresponding bit of the addend is a "1".
P REGISTER	The Program Address Counter is a two's complement additive register (modulus 2^{15}) which generates in sequential order the storage addresses containing the individual program steps.
PROGRAM	A precise sequence of instructions that accomplishes a computer routine; a plan for the solution of a problem.
PROGRAM STEP	Two 24-bit instructions contained in one 48-bit storage address; the higher-order 24 bits are the upper instruction, lower-order 24 bits, the lower instruction. A pair of instructions is read from storage, and the upper instruction is executed first. The lower one is then executed, except when the upper one provides for skipping the lower one.
Q REGISTER	Auxiliary arithmetic register which assists the A register in the more complicated arithmetic operations (modulus 2^{48}-1).
RANDOM ACCESS	Access to storage under conditions in which the next position from which information is to be obtained is in no way dependent on the previous one.
R REGISTER	Address Buffer register. Two's complement subtractive register (modulus 2^{15}) which acts as an exchange register for transmissions involving index registers.
READ	To remove a quantity from a storage location.

READY	1) To input/output control signal sent by the computer or an external equipment. The ready signal indicates that a word or character is available for transmission. 2) A status response indicating that the external device being addressed is ready for operation.
RELATIVE ADDRESS	Identifies a word in a subroutine or routine with respect to its position. Relative addresses are translated into absolute addresses by the addition of some specific reference address, usually that at which the first word of the routine is stored.
REPLACE	In the title of an instruction, the result of the execution of the instruction is stored in the location from which the initial operand was obtained.
RESUME	The input/output control signal sent by either the computer or an external equipment to indicate that it is prepared to receive another word (48 bits) or character (usually 6 bits). The resume signal is thus a request for data.
RETURN JUMP	An instruction that jumps from a sequence of instructions to initiate a second sequence and prepares for continuing the first sequence after the second is completed.
ROUTINE	The sequence of operations which the computer performs under the direction of a program.
S^1 REGISTER	Storage Address register (even storage). Selects the storage address specified by the contents of the P register.
S^2 REGISTER	Storage Address register (odd storage). Selects the storage address specified by the contents of the P register.
SCALE FACTOR	One or more coefficients by which quantities are multiplied or divided so that they lie in a given range of magnitude.
SCANNER	A circuit used to search for one of a number of possible conditions and to initiate action when a condition is detected. The auxiliary scanner scans the six buffer channels for auxiliary requests; the interrupt scanner looks for interrupt requests from external equipments.

SECONDARY REGISTERS	Transient registers not displayed on the console (U^2, $S^{1,2}$, $Z^{1,2}$, R, X, O^1-O^6).
SHIFT	To move the bits of a quantity right or left.
SIGN BIT	In registers where a quantity is treated as signed by use of one's complement notation, the bit in the highest-order stage of the register. If the bit is "1", the quantity is negative; if the bit is "0", the quantity is positive.
SIGN EXTENSION	The duplication of the sign bit in the higher-order stages of a register. To omit the execution of a lower instruction in a program; occurs only if the upper instruction provides for skipping on a specified condition, and the condition is met.
STAGE	The FFs and inverters associated with a bit position of a register.
STORE	To transmit information to a device from which the unaltered information can later be obtained.
SUBINSTRUCTION	The index code specifies one of eight forms of the instruction indicated by the operation code. Such forms are called "subinstructions". Thus, 74.0 is a subinstruction of instruction 74.
TOGGLE	To complement each bit of a quantity as a result of an individual condition.
TRANSFER	High-speed data input/output transmission under direct program control.
TRANSMISSION, FORCED	A transfer of bits into a register which has not been cleared previously.
TRANSMISSION, ONES	A transfer of ones into a register which has been cleared.
TRANSMISSION, ZEROS	A transfer of zeros into a register which has been set.
TWO'S COMPLEMENT	Number that results from subtracting each bit of a number from "0". The two's complement may be formed by complementing each bit of the given number and then adding one to the result, performing the required carries.

U^1 REGISTER	Program Control register. Holds a program step while the two instructions contained in it are executed.
U^2 REGISTER	Auxiliary Program Control register. A 15-bit subtractive accumulator (modulus $2^{15}-1$) used primarily for modification of the base execution address.
UPPER ADDRESS	The execution address portion of an upper instruction; bit positions 24 through 38 of a 48-bit register or storage address.
UPPER INSTRUCTION	See Program Step.
WORD	A unit of information which has been coded for use in the computer as a series of bits. The normal work length is 48 bits.
WRITE	To enter a quantity into a storage location.
X REGISTER	Exchange register. All internal transmissions between the arithmetic section and the rest of the computer are made through X.
Z^1 REGISTER	Storage Restoration register (even storage). Holds the word to be written into a given storage location.
Z^2 REGISTER	Storage Restoration register (odd storage). Holds the word to be written into a given storage location.

APPENDIX SECTION

APPENDIX I
NUMBER SYSTEMS

Any number system may be defined by two characteristics, the radix or base and the modulus. The radix or base is the number of unique symbols used in the system. The decimal system has ten symbols, 0 through 9. Modulus is the number of unique quantities or magnitudes a given system can distinguish. For example, an adding machine with ten digits, or counting wheels, would have a modulus of $10^{10}-1$. The decimal system has no modulus because an infinite number of digits can be written, but the adding machine has a modulus because the highest number which can be expressed is 9,999,999,999.

Most number systems are positional, that is, the relative position of a symbol determines its magnitude. In the decimal system, a 5 in the units column represents a different quantity than a 5 in the tens column. Quantities equal to or greater than 1 may be represented by using the 10 symbols as coefficients of ascending powers of the base 10. The number 984_{10} is:

$$\begin{array}{rcrcr} 9 \times 10^2 & = & 9 \times 100 & = & 900 \\ +8 \times 10^1 & = & 8 \times 10 & = & 80 \\ +4 \times 10^0 & = & 4 \times 1 & = & \underline{4} \\ & & & & 984_{10} \end{array}$$

Quantities less than 1 may be represented by using the 10 symbols as coefficients of ascending negative powers of the base 10. The number 0.593_{10} may be represented as:

$$\begin{array}{rcrcr} 5 \times 10^{-1} & = & 5 \times .1 & = & .5 \\ +9 \times 10^{-2} & = & 9 \times .01 & = & .09 \\ +3 \times 10^{-3} & = & 3 \times .001 & = & \underline{.003} \\ & & & & 0.593_{10} \end{array}$$

BINARY NUMBER SYSTEM

Computers operate faster and more efficiently by using the binary number system. There are only two symbols 0 and 1; the base = 2. The following shows the positional value.

$$\begin{array}{ccccccc} \ldots\ 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & \\ =32 & =16 & =8 & =4 & =2 & =1 & \text{Binary point} \end{array}$$

The binary number 0 1 1 0 1 0 represents:

$$\begin{aligned}
0 \times 2^5 &= 0 \times 32 = 0 \\
+1 \times 2^4 &= 1 \times 16 = 16 \\
+1 \times 2^3 &= 1 \times 8 = 8 \\
+0 \times 2^2 &= 0 \times 4 = 0 \\
+1 \times 2^1 &= 1 \times 2 = 2 \\
+0 \times 2^0 &= 0 \times 1 = \underline{0} \\
& \qquad\qquad 26_{10}
\end{aligned}$$

Fractional binary numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5} . . .
Binary Point	.	$=1/2$	$=1/4$	$=1/8$	$=1/16$	$=1/32$

The binary number 0.10 110 may be represented as:

$$\begin{aligned}
1 \times 2^{-1} &= 1 \times 1/2 = 1/2 = 8/16 \\
+0 \times 2^{-2} &= 0 \times 1/4 = 0 \quad = 0 \\
+1 \times 2^{-3} &= 1 \times 1/8 = 1/8 = 2/16 \\
+1 \times 2^{-4} &= 1 \times 1/16 = 1/16 = \underline{1/16} \\
& \qquad\qquad 11/16_{10} = 0.6875
\end{aligned}$$

OCTAL NUMBER SYSTEM

The octal number system uses eight discrete symbols, 0 through 7. With the base eight the positional value is:

. . .	8^5	8^4	8^3	8^2	8^1	8^0
	32, 768	4,096	512	64	8	1

The octal number 513_8 represents:

$$\begin{aligned}
5 \times 8^2 &= 5 \times 64 = 320 \\
+1 \times 8^1 &= 1 \times 8 = 8 \\
+3 \times 8^0 &= 3 \times 1 = \underline{3} \\
& \qquad\qquad 331_{10}
\end{aligned}$$

Fractional octal numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

$$8^{-1} \quad 8^{-2} \quad 8^{-3} \quad 8^{-4} \quad \ldots$$

$$1/8 \quad 1/64 \quad 1/512 \quad 1/4096$$

The octal number 0.4520 represents:

$$
\begin{aligned}
4 \times 8^{-1} &= 4 \times 1/8 = 4/8 &= 256/512 \\
+5 \times 8^{-2} &= 5 \times 1/64 = 5/64 &= 40/512 \\
+2 \times 8^{-3} &= 2 \times 1/512 = 2/512 &= \underline{2/512} \\
& & 298/512 = 149/256_{10} = .5811
\end{aligned}
$$

ARITHMETIC

ADDITION AND SUBTRACTION

Binary numbers are added according to the following rules:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 with a carry of 1

The addition of two binary numbers proceeds as follows (the decimal equivalents verify the result):

Augend	0111	(7)
Addend	+0100	+(4)
Partial Sum	0011	
Carry	1	
Sum	1011	(11)

Subtraction may be performed as an addition:

8	(minuend)		8	(minuend)
-6	(subtrahend)	or	+4	(10's complement or subtrahend)
2	(difference)		2	(difference - omit carry)

The second method shows subtraction performed by the "adding the complement" method. The omission of the carry in the illustration has the effect of reducing the result by 10.

One's Complement

The 1604-A performs all arithmetic operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in one's complement notation.

The one's complement representation of a number is found by subtracting each bit of the number from 1. For example:

$$\begin{array}{rl} 1111 & \\ -1001 & \quad 9 \\ \hline 0110 & \quad \text{(one's complement of 9)} \end{array}$$

This representation of a negative binary quantity may also be obtained by substituting "1's" for "0's" and "0's" for "1's".

The value zero can be represented in one's complement notation in two ways:

$0000 \rightarrow 00_2$ Positive (+) Zero

$1111 \rightarrow 11_2$ Negative (-) Zero

The rules regarding the use of these two forms for computation are:

1) Both positive and negative zero are acceptable as arithmetic operands.
2) If the result of an arithmetic operation is zero, it will be expressed as positive zero. The one exception to this rule is when negative zero is added to negative zero. In this case, the result is negative zero.

One's complement notation applies not only to arithmetic operations performed in A, but also to the modification of execution addresses in the U^2 register. During address modification, the modified address will equal 77777_8 only if the unmodified execution address equals 77777_8 and b = 0 or $(B^b) = 77777_8$.

Two's Complement

The counters in the computer use two's complement arithmetic. A counter is a register with provisions for increasing its contents by one if it is additive (P register) or decreasing its contents by one if it is subtractive (R register). A two's complement counter is open-ended; there is no end-around carry or borrow.

Positive numbers have the same representation in both systems while negative values differ by one count.

Count	2's comp. rep.	1's comp. rep.
+2	00010	00010
+1	00001	00001
0	00000	00000
-1	11111	11110
-2	11110	11101

The difference in the representation of negative values in these two systems is due to the skipping of the "all one's" count in one's complement notation. In the one's complement system the end-around-carry feature of the register automatically changes a count of all one's to all zeros. (Note exception under one's complement.)

As an example, if the content of a subtractive counter is positive seven (0111) and is to be reduced by one, add the two's complement expression of negative one, (1111), to 0111 as shown below. The result is six.

$$\begin{array}{r} 0111 \\ \underline{+1111} \\ 0110 \end{array}$$

Note that the two's complement expression for a negative number may also be formed by adding one to the one's complement representation of the number.

MULTIPLICATION

Binary multiplication proceeds according to the following rules:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Multiplication is always performed on a bit-by-bit basis. Carries do not result from multiplication, since the product of any two bits is always a single bit.

Decimal example:

multiplicand	14	
multiplier	12	
partial products	28	
	14	(shifted one place left)
product	168_{10}	

The shift of the second partial product is a shorthand method for writing the true value 140.

Binary example:

multiplicand	(14)	1110	
multiplier	(12)	1100	
partial products		0000	
		0000	shift to place digits in proper columns
		1110	
		1110	
product	(168_{10})	10101000_2	

The computer determines the running subtotal of the partial products. Rather than shifting the partial product to the left to position it correctly, the computer right shifts the summation of the partial products one place before the next addition is made. When the multiplier bit is "1", the multiplicand is added to the running total and the results are shifted to the right one place. When the multiplier bit is "0", the partial product subtotal is shifted to the right (in effect, the quantity has been multiplied by 10_2).

DIVISION

The following example shows the familiar method of decimal division:

		14	quotient
divisor	13	$\overline{)185}$	dividend
		13	
		55	partial dividend
		52	
		3	remainder

The computer performs division in a similar manner (using binary equivalents):

```
                        1110     quotient  (14)
divisor         1101 | 10111001  dividend
                       1101
                       ----
                       10100
                        1101
                        ----
                        1110     partial dividends
                        1101
                        ----
                          11     remainder  (3)
```

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

CONVERSIONS

The procedures that may be used when converting from one number system to another are power addition, double dabble, and substitution.

Recommended Conversion Procedures (Integer and Fractional)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Double Dabble
Decimal to Octal	Double Dabble
Binary to Octal	Substitution
Octal to Binary	Substitution

GENERAL RULES

$r_i > r_f$: use Double Dabble, Substitution

$r_i < r_f$: use Power Addition, Substitution

r_i = Radix of initial system

r_f = Radix of final system

POWER ADDITION

To convert a number from r_i to r_f ($r_i < r_f$) write the number in its expanded r_i polynomial form and symplify using r_f arithmetic.

EXAMPLE 1 Binary to Decimal (Integer)

$$
\begin{aligned}
010\ 111_2 &= 1(2^4) + 0(2^3) + 1(2^2) + 1(2^1) + 1(2^0) \\
&= 1(16) + 0(8) + 1(4) + 1(2) + 1(1) \\
&= 16 + 0 + 4 + 2 + 1 \\
&= 23_{10}
\end{aligned}
$$

EXAMPLE 2 Binary to Decimal (Fractional)

$$
\begin{aligned}
.0101_2 &= 0(2^{-1}) + 1(2^{-2}) + 0(2^{-3}) + 1(2^{-4}) \\
&= 0 + 1/4 + 0 + 1/16 \\
&= 5/16_{10} = 0.3125
\end{aligned}
$$

EXAMPLE 3 Octal to Decimal (Integer)

$$
\begin{aligned}
324_8 &= 3(8^2) + 2(8^1) + 4(8^0) \\
&= 3(64) + 2(8) + 4(1) \\
&= 192 + 16 + 4 \\
&= 212_{10}
\end{aligned}
$$

EXAMPLE 4 Octal to Decimal (Fractional)

$$
\begin{aligned}
.44_8 &= 4(8^{-1}) + 4(8^{-2}) \\
&= 4/8 + 4/64 \\
&= 36/64_{10} = 0.5625
\end{aligned}
$$

DOUBLE DABBLE

To convert a whole number from r_i to r_f ($r_i > r_f$):

1) Divide r_i by r_f using r_i arithmetic
2) The remainder is the lowest order bit in the new expression
3) Divide the integral part from the previous operation by r_f
4) The remainder is the next higher order bit in the new expression
5) The process continues until the division produces only a remainder which will be the highest order bit in the r_f expression.

To convert a fractional number from r_i to r_f:

1) Multiply r_i by r_f using r_i arithmetic
2) The integral part is the highest order bit in the new expression
3) Multiply the fractional part from the previous operation by r_f.
4) The integral part is the next lower order bit in the new expression
5) The process continues until sufficient precision is achieved or the process terminates.

EXAMPLE 1 Decimal to Binary (Integer)

45 ÷ 2 = 22 remainder 1; record	1
22 ÷ 2 = 11 remainder 0; record	0
11 ÷ 2 = 5 remainder 1; record	1
5 ÷ 2 = 2 remainder 1; record	1
2 ÷ 2 = 1 remainder 0; record	0
1 ÷ 2 = 0 remainder 1; record	1
	101101

Thus: $45_{10} = 101101_2$

EXAMPLE 2 Decimal to Binary (Fractional)

.25 x 2 = 0.5; record	0
.5 x 2 = 1.0; record	1
.0 x 2 = 0.0; record	0
	.010

Thus: $.25_{10} = .010_2$

EXAMPLE 3 Decimal to Octal (Integer)

273 ÷ 8 = 34 remainder 1; record	1
34 ÷ 8 = 4 remainder 2; record	2
4 ÷ 8 = 0 remainder 4; record	4
	421

Thus: $273_{10} = 421_8$

EXAMPLE 4 Decimal to Octal (Fractional)

$.55 \times 8 = 4.4$; record	4
$.4 \times 8 = 3.2$; record	3
$.2 \times 8 = 1.6$; record	1
-- --	-
-- --	-
	.431...

Thus: $.55_{10} = .431\ldots_8$

SUBSTITUTION

This method permits easy conversion between octal and binary representations of a number. If a number in binary notation is partitioned into triplets to the right and left of the binary point, each triplet may be converted into an octal digit. Similarly each octal digit may be converted into a triplet of binary digits.

EXAMPLE 1 Binary to Octal

Binary = 110 000 . 001 010

Octal = 6 0 . 1 2

EXAMPLE 2 Octal to Binary

Octal = 6 5 0 . 2 2 7

Binary = 110 101 000 . 010 010 111

COMMON PURE NOTATIONS

Decimal	Binary	Octal
00	00000	00
01	00001	01
02	00010	02
03	00011	03
04	00100	04
05	00101	05
06	00110	06
07	00111	07
08	01000	10
09	01001	11
10	01010	12
11	01011	13
12	01100	14
13	01101	15
14	01110	16
15	01111	17
16	10000	20
17	10001	21

POWERS OF COMMON NUMBER SYSTEMS

$2^0 = 1$	$8^0 = 1$	$10^0 = 1$
$2^1 = 2$	$8^1 = 8$	$10^1 = 10$
$2^2 = 4$	$8^2 = 64$	$10^2 = 100$
$2^3 = 8$	$8^3 = 512$	$10^3 = 1,000$
$2^4 = 16$	$8^4 = 4,096$	$10^4 = 10,000$
$2^5 = 32$	$8^5 = 32,768$	$10^5 = 100,000$
$2^6 = 64$	$8^6 = 262,144$	$10^6 = 1,000,000$
$2^7 = 128$	$8^7 = 2,097,152$	
$2^8 = 256$	$8^8 = 16,777,216$	
$2^9 = 512$		
$2^{10} = 1,024$		

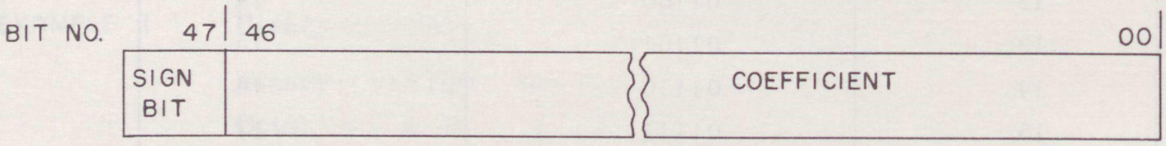
FIXED POINT AND FLOATING POINT NUMBERS

Any number may be expressed in the form kB^n, where k is a coefficient, B a base number, and the exponent n the power to which the base number is raised.

A fixed point number assumes:

1) The exponent n = 0 for all fixed point numbers.
2) The coefficient, k, occupies the same bit positions within the computer word for all fixed point numbers.
3) The radix (binary) point remains fixed with respect to one end of the expression.

A 1604 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any 1604 fixed point number designates the sign of the coefficient (47 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.



The coefficient may be an integer or fraction. The radix (binary) point, in the case of an integer, is assumed to be immediately to the right of the lowest order bit (00). In the case of the fraction, the point is just to the right of the sign bit.

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of places shifted) is expressed as the power of the base. For example, $5,100,000_{10}$ may be expressed as 0.51×10^7, 0.051×10^8, 0.0051×10^9, etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are multiplied, the result is a number < 1. If the same two fractions are added, subtracted, or divided, the result may be greater than one and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

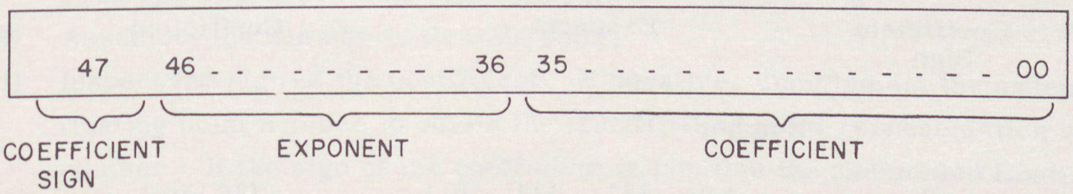
As an alternative to fixed point operation, a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

By shifting the radix point and increasing or decreasing the value of the exponent, widely varying quantities which do not exceed the capacity of the machine may be handled.

Floating point numbers within the computer are represented in a form similar to that used in "scientific" notation, that is, a coefficient or fraction multiplied by a number raised to a power. Since the computer uses only binary numbers, the numbers are multiplied by powers of two.

$F \cdot 2^E$ where: F = fraction
E = exponent

In floating point, different coefficients need not relate to the same power of the base as they do in fixed point format. Therefore, the construction of a floating point number includes not only the coefficient but also the exponent.



Coefficient

The coefficient consists of a 36-bit fraction in the 36 lower-order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than 1/2 but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent

The floating point exponent is expressed as an 11-bit quantity with a value ranging from 0000_8 to 3777_8. It is formed by adding a true positive exponent and a bias of 2000_8 or a true negative exponent and a bias of 1777_8. This results in a range of biased exponents as shown below.

True Positive Exponent	Biased Exponent	True Negative Exponent	Biased Exponent
+0	2000	-0	2000*
+1	2001	-1	1776
+2	2002	-2	1775
--	----	--	----
--	----	--	----
+1776	3776	-1776	0001
$+1777_8$	3777_8	-1777_8	0000_8

The exponent is biased so that floating point operands can be compared with each other in the normal fixed point mode.

As an example, compare the unbiased exponents of $+52_8$ and $+0.02_8$ (Example 1).

EXAMPLE 1 Number = +52

0	0 0 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient

Number = +0.02

0	1 1 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

In this case +0.02 appears to be larger than +52 because of the larger exponent. If, however, both exponents are biased, (Example 2) changing the sign of both exponents makes +52 greater than +0.02.

---

* Minus zero is sensed as positive zero by the computer and is therefore biased by 2000_8 rather than 1777_8.

EXAMPLE 2 Number = $+52_8$

0	1 0	000	000	110	(36 bits)
Coefficient Sign		Exponent			Coefficient

Number = $+0.02_8$

0	0 1	111	111	011	(36 bits)
Coefficient Sign		Exponent			Coefficient

When bias is used with the exponent floating-point operation is more versatile since floating-point operands can be compared with each other in the normal fixed point mode.

CONVERSION PROCEDURES

Fixed Point to Floating Point

1) Express the number in binary.
2) Normalize the number. A normalized number has the most significant 1 positioned immediately to the right of the binary point and is expressed in the range $1/2 \leq k < 1$.
3) Inspect the sign of the true exponent. If the sign is positive add 2000_8 (bias) to the true exponent of the normalized number. If the sign is negative add the bias 1777_8 to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.
4) Assemble the number in floating point.
5) Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive the assembled floating point number is the true representation.

EXAMPLE 1 Convert +4.0 to floating point

1) The number is expressed in octal.
2) Normalize. $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$.
3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.

4) Assemble number in floating point format.

Coefficient = $400\ 000\ 000\ 000_8$

Biased Exponent = 2003_8

Assembled word = $2003\ 400\ 000\ 000\ 000_8$

5) Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

EXAMPLE 2 Convert -4.0 to floating point

1) The number is expressed in octal.

2) Normalize. $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$

3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.

4) Assemble number in floating point format.

Coefficient = $400\ 000\ 000\ 000_8$

Biased Exponent = 2003_8

Assembled word = $2003\ 400\ 000\ 000\ 000_8$

5) Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for $-4.0 = 5774\ 377\ 777\ 777\ 777_8$

EXAMPLE 3 Convert 0.5_{10} to floating point

1) Convert to octal. $0.5_{10} = 0.4_8$

2) Normalize. $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$

3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 0.

4) Assemble number in floating point format.

Coefficient = $400\ 000\ 000\ 000_8$

Biased Exponent = 2000_8

Assembled word = $2000\ 400\ 000\ 000\ 000_8$

5) Since the sign of the coefficient is positive, the floating point representation of $+0.5_{10}$ is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777_8 instead of 2000_8 because of the negative exponent. The 1604, however, recognizes -0 as +0 and biases the exponent by 2000_8.

EXAMPLE 4 Convert 0.04_8 to floating point

1) The number is expressed in octal.
2) Normalize. $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$.
3) Since the sign of the true exponent is negative, add 1777_8 (bias) to the true exponent. Biased exponent = $1777_8 + (-3) = 1774_8$.
4) Assemble number in floating point format.

   Coefficient = $400\ 000\ 000\ 000_8$

   Biased Exponent = 1774_8

   Assembled word = $1774\ 400\ 000\ 000\ 000_8$
5) Since the sign of the coefficient is positive, the floating point representation of 0.04_8 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

<u>Floating Point to Fixed Point Format</u>

1) If the floating point number is negative, complement the entire floating point word and record the fact that the quantity is negative. The exponent is now in a true biased form.
2) If the biased exponent is equal to or greater than 2000_8 subtract 2000_8 to obtain the true exponent. If less than 2000_8 subtract 1777_8 to obtain true exponent.
3) Separate the coefficient and exponent. If the true exponent is negative the binary point should be moved to the left the number of bit positions indicated by the true exponent. If the true exponent is positive, the binary point shoud be moved to the right the number of bit positions indicated by the true exponent.
4) The coefficient has now been converted to fixed binary. The sign of the coefficient will be negative if the floating point number was complemented in step one. (The sign bit must be extended if the quantity is placed in a register.)
5) Represent the fixed binary number in fixed octal notation.

EXAMPLE 1 Convert floating point number $2003\ 400\ 000\ 000\ 000_8$ to fixed octal

1) The floating point number is positive and remains uncomplemented.
2) The biased exponent $> 2000_8$, therefore subtract 2000_8 from the biased exponent to obtain the true exponent of the number. $2003 - 2000 = +3$
3) Coefficient = $400\ 000\ 000\ 000_8 = .100_2$. Move binary point to the right 3 places.

   Coefficient = 100.0_2

4) The sign of the coefficient is positive because the floating point number was not complemented in step one.

5) Represent in fixed octal notation. $100.0 \times 2^0 = 4.0 \times 8^0$

EXAMPLE 2 Convert floating point number $5774\ 377\ 777\ 777\ 777_8$ to fixed octal

1) The sign of the coefficient is negative, therefore, complement the floating point number.

Complement = $2003\ 400\ 000\ 000\ 000_8$

2) The biased exponent (in complemented form) $> 2000_8$, therefore subtract 2000_8 from the biased exponent to obtain the true exponent of the number. 2003 - 2000 = +3

3) Coefficient = $4000\ 000\ 000\ 000_8 = 0.100_2$

Move binary point to the right 3 places.

Coefficient = 100.0_2

4) The sign of the coefficient will be negative because the floating point number was originally complemented.

5) Convert to fixed octal. $-100.0_2 = -4.0_8$

EXAMPLE 3 Convert floating point number $1774\ 400\ 000\ 000\ 000_8$ to fixed octal

1) The floating point number is positive and remains uncomplemented.

2) The biased exponent $< 2000_8$, therefore subtract 1777_8 from the biased exponent to obtain the true exponent of the number. $1774_8 - 1777_8 = -3$

3) Coefficient = $400\ 000\ 000\ 000_8 = .100_2$

Move binary point to the left 3 places.

Coefficient = $.000100_2$

4) The sign of the coefficient is positive because the floating point number was not complemented in step one.

5) Represent in fixed octal notation. $.000100_2 = .04_8$

APPENDIX II

FAULTS

Certain fault conditions may occur in the execution of a computer program which may be sensed by EXF instructions. The occurrence of the fault does not stop operation but sets an indicator that can be sensed. A fault is visually indicated on the console.

SHIFT FAULT

Any attempt to shift a register more than 127_{10} (177_8) places right or left results in a shift fault. If the fault exists, the indicator is set prior to execution of the shift instruction and the shift fault background light on the console display panel is lighted. The shifts will be performed regardless of the status of the fault indicator. If an interrupt has been selected, the main program will be interrupted after executing the shift instruction. The shift fault may be sensed by 47 7 00120, 1.

DIVIDE FAULT

A divide fault occurs in fixed point divide instructions (25 and 27) when the divisor is zero or the required quotient exceeds the 47-bit capacity of the quotient register, Q. The sign bit of Q is examined at the end of the division phase. If it is equal to "1", a divide fault has occurred. If an interrupt has been selected, the main program will be interrupted after the divide instruction is completed. A divide fault is sensed by a 74 0 00110, 1.

OVERFLOW FAULT

An overflow fault results when the capacity of the A register (2^{47}-1) is exceeded. The fault is detected at the time the operation causing the overflow takes place.

If an interrupt on arithmetic faults has been selected, the main program will be halted before another instruction can be executed.

An overflow may be sensed by a 74 7 00130, 1.

CLOCK OVERFLOW

A clock overflow results if the capacity of the A register is exceeded during an advance clock operation. If an interrupt on arithmetic faults has been selected, the interrupt will occur before an instruction can be executed after the advance clock operation. Clock overflow may be sensed by 74 7 00300, 1.

EXPONENT (Floating Point Range) FAULT

The exponent fault occurs during floating point instructions when the exponent of the result, after rounding and normalizing, is $\geq 2^{+10}$ (overflow) or $\leq 2^{-10}$ (underflow). The exponent fault is sensed by a 74 7 00140, 1.

EVEN AND ODD STORAGE FAULTS

These faults indicate a failure in computer storage and turn on background lights on the console display. The indicators may be cleared by an internal master clear. If a storage fault is produced, maintenance should be notified.

APPENDIX III

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

APPENDIX IV

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000	0000
to	to
0777	0511
(Octal)	(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	0512
to	to
1777	1023
(Octal)	(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	2048
to	to
4777	2559
(Octal)	(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	2560
to	to
5777	3071
(Octal)	(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)

APPENDIX V

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

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[illegible]	.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

APPENDIX VI

EXF AND CHARACTER CODES

1604-A EXF CODES

SELECT INTERNAL

74 0	000C0	Interrupt on Channel C inactive
	000C1	Remove Interrupt Selection on Channel C
	00100	Interrupt on Arithmetic Faults
	00101	Remove Interrupt on Arithmetic Faults
	01000	Start Real-Time Clock
	02000	Stop Real-Time Clock
	00070	Clear Arithmetic Faults
	C0000	Clear All Channel C Selection

C = channel = 1-6

SENSE INTERNAL

74 7	000C0	Exit on Channel C Active
	000C1	Exit on Channel C Inactive

C = channel = 1-6

001A0	Exit on Arithmetic Fault A
001A1	Exit on No Arithmetic Fault A

A = 1 : Divide
2 : Shift
3 : Overflow
4 : Exponent Overflow Fault
5 : Exponent Underflow Fault

74 7 0C000	Exit on Channel C Interrupt
74 7 0C001	Exit on No Channel C Interrupt

C = 1 = Channel 1 (odd)
C = 2 = Channel 2 (even)
C = 3 = Channel 3 (odd)
C = 4 = Channel 4 (even)
C = 5 = Channel 5 (odd)
C = 6 = Channel 6 (even)

74 7 001T0	Exit on Channel T Interrupt
74 7 001T1	Exit on No Channel T Interrupt
	T = 6 = Channel 7 - (Output)
	T = 7 = Channel 7 - (Input)
74 7 00200	Exit on Lower (Monitor Interrupt Exit flip-flop)
74 7 00201	Exit on Upper (Monitor Interrupt Exit flip-flop)
74 7 00300	Exit on Clock Overflow
74 7 00301	Exit on No Clock Overflow

CONSOLE EQUIPMENT

(CHANNEL PAIR 1 and 2)

SELECT

INPUT	74 0	11140	Select Typewriter for Input, and Interrupt on Carriage Return
		100	Select Typewriter for Input, and No Interrupt on C.R.
		200	Select Paper Tape Reader, and No Interrupt on End of Tape
		210	Set End of Tape Indicator
		220	Select Paper Tape Reader, and Interrupt on End of Tape
OUTPUT	74 0	21100	Select Typewriter for Output, Assembly Mode
		110	Select Typewriter for Output, Character Mode
		200	Select Paper Tape Punch, Assembly Mode
		210	Select Paper Tape Punch, Character Mode
		240	Turn Paper Tape Punch Motor Off

SENSE

INPUT	74 7	11200	Exit on Paper Tape Reader, End of Tape
		201	Exit on Paper Tape Reader, No End of Tape
		210	Exit on Paper Tape Reader in Assembly Mode
		211	Exit on Paper Tape Reader in Character Mode
		140	Exit on Typewriter in Lower Case
		141	Exit on Typewriter in Upper Case
		100	Exit on Carriage Return or Tab from Typewriter
		101	Exit on No Carriage Return or Tab from Typewriter

OUTPUT	74 7	21200	Exit on Paper Tape Punch Out of Tape
		201	Exit on Paper Tape Punch Not Out of Tape

1607 EXF CODES

(CHANNEL C)

SELECT

INPUT	74 0	C20N1	Select Read Tape N, Binary Mode
		0N2	Select Read Tape N, Coded Mode
		001	Read Selected Tape, Binary Mode
		002	Read Selected Tape, Coded Mode
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
OUTPUT	74 0	C20N1	Select Write Tape N, Binary Mode
		0N2	Select Write Tape N, Coded Mode
		001	Write Selected Tape, Binary Mode
		002	Write Selected Tape, Coded Mode
		003	Write End of File Mark on Selected Tape
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock

SENSE

INPUT	74 7	C2000	Exit on Ready to Read
		001	Exit on Not Ready to Read
		002	Exit on Read Parity Error
		003	Exit on No Read Parity Error
		004	Exit on Read Length Error
		005	Exit on No Read Length Error
		006	Exit on End of File Mark
		007	Exit on No End of File Mark

OUTPUT	74 7	C2000	Exit on Ready to Write
		001	Exit on Not Ready to Write
		002	Exit on Write Reply Parity Error
		003	Exit on No Write Reply Parity Error
		004	Exit on Write Reply Length Error
		005	Exit on No Write Reply Length Error
		006	Exit on End of Tape Marker
		007	Exit on No End of Tape Marker

1607 EXF CODES

(CHANNEL C)

SELECT

INPUT	74 0	C77N1	Select Read Tape N, Binary Mode
		7N2	Select Read Tape N, Coded Mode
		001	Read Selected Tape, Binary Mode
		002	Read Selected Tape, Coded Mode
		004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
		101	Turn Off "Tape Indicator" on Read Unit
		102	Set Low Density on Read Unit
		103	Set High Density on Read Unit
		104	Search File Mark Forward on Read Unit
		105	Search File Mark Backward on Read Unit
		106	Remove Interrupt Selection on Read Unit
OUTPUT	74 0	C77N1	Select Write Tape N, Binary Mode
		7N2	Select Write Tape N, Coded Mode
		001	Write Selected Tape, Binary Mode
		002	Write Selected Tape, Coded Mode
		003	Write End of File Mark on Selected Tape

OUTPUT	74 0	C7004	Interrupt When Selected Tape Ready
		005	Rewind Selected Tape
		006	Backspace Selected Tape
		007	Rewind Selected Tape with Interlock
		101	Turn Off "Tape Indicator" on Write Unit
		102	Set Low Density on Write Unit
		103	Set High Density on Write Unit
		104	Skip Bad Spot on Selected Write Unit
		106	Remove Interrupt on Write Unit

SENSE

INPUT	74 7	C7000	Exit on Ready to Read
		001	Exit on Not Ready to Read
		002	Exit on Read Parity Error
		003	Exit on No Read Parity Error
		004	Exit on Read Length Error
		005	Exit on No Read Length Error
		006	Exit on End of File Mark
		007	Exit on No End of File Mark
		106	Exit When Read Unit is Rewinding or at Load Point
		107	Exit When Read Unit is Not Rewinding or is at Load Point
OUTPUT	74 7	C7000	Exit on Ready to Write
		001	Exit on Not Ready to Write
		002	Exit on Write Reply Parity Error
		003	Exit on No Write Reply Parity Error
		004	Exit on Write Reply Length Error
		005	Exit on No Write Reply Length Error
		006	Exit on End of Tape Marker
		007	Exit on No End of Tape Marker
		106	Exit when Write Unit is Rewinding or at Load Point
		107	Exit when Write Unit is Not Rewinding or is at Load Point

1610 EXF CODES

(CHANNEL C)

SELECT

INPUT	74 0	C4001	Select Primary Read Station
		002	Select Secondary Read Station
		003	Select Primary and Secondary Read Stations
		005	Select Primary Read Station and Interrupt
		006	Select Secondary Read Station and Interrupt
		007	Select Primary and Secondary Read Stations and Interrupt
OUTPUT	74 0	C4001	Select Printer
		002	Select Punch
		005	Select Printer and Interrupt
		006	Select Punch and Interrupt

SENSE

INPUT	74 7	C4002	Exit on Reader Ready
		003	Exit on Reader Not Ready
		004	Exit on 1604 Selected
		005	Exit on 1604 Not Selected
OUTPUT	74 7	C4002	Exit on Printer Ready
		003	Exit on Printer Not Ready
		004	Exit on Punch Ready
		005	Exit on Punch Not Ready
		010	Exit on 1604 Selected
		011	Exit on 1604 Not Selected

1612 EXF CODES

(CHANNEL C)

SELECT

OUTPUT (ONLY)	74 0 C6000	Select Printer
	001	Single Space the Printer
	002	Double Space the Printer
	003	Select Format Channel 7
	004	Select Format Channel 8
	010	Clear Monitor Channels 1 - 6
	01N	Select Monitor Channel N : N = 1 - 6

SENSE

OUTPUT (ONLY)	74 7 C6000	Exit on Printer Ready
	001	Exit on Printer Not Ready

1615 FUNCTION CODES

($N = 1_8 - 10_8$)

OUTPUT

74 0 C	20N1	Select Tape N To Write Binary
	20N2	Select Tape N To Write Coded
	2001	Prepare Selected Tape To Write Binary
	2002	Prepare Selected Tape To Write Coded
	2003	Write End-Of-File Mark On Selected Tape
	2004	Select Interrupt When Write Tape Next Ready
	2005	Rewind Selected Write Tape
	2006	Backspace Selected Write Tape
	2007	Rewind-Unload Selected Write Tape
	2400	Clear Interrupt Selections On Write Tape
	2401	Set Low Density On Selected Write Tape
	2402	Set High Density On Selected Write Tape
	2403	Skip Bad Spot On Selected Write Tape
	2404	Select Interrupt On Next Error

SENSE		
74 7 C	2000	Exit On Ready To Write
	2001	Exit On Not Ready To Write
	2002	Exit On Write Reply Parity Error
	2003	Exit On No Write Reply Parity Error
	2004	Exit On Write Reply Length Error
	2005	Exit On No Write Reply Length Error
	2006	Exit On End Of Tape Marker
	2007	Exit On Not End Of Tape Marker
	2400	Exit On Ready To Select
	2401	Exit On Not Ready To Select
	2402	Exit On Load Point
	2403	Exit On Not Load Point
	2404	Exit On Interrupt On Write Tape
	2405	Exit On No Interrupt On Write Tape
	2406	Exit On Write Program Error
	2407	Exit On No Write Program Error
INPUT		
74 0 C	20N1	Select Tape N To Read Binary One Record
	20N2	Select Tape N To Read Coded One Record
	22N1	Select Tape N To Read Binary One File
	22N2	Select Tape N To Read Coded One File
	2001	Prepare Selected Tape To Read Binary One Record
	2002	Prepare Selected Tape To Read Coded One Record
	2201	Prepare Selected Tape To Read Binary One File
	2202	Prepare Selected Tape To Read Coded One File
	2003	Move Selected Read Tape Forward One Record
	2203	Search File Mark Forward
	2004	Select Interrupt When Read Tape Next Ready
	2005	Rewind Selected Read Tape
	2006	Backspace Selected Read Tape
	2206	Search File Mark Backward
	2007	Rewind-Unload Selected Read Tape
	2400	Clear Interrupt Selections On Read Tape
	2401	Set Low Density On Selected Read Tape
	2402	Set High Density On Selected Read Tape
	2404	Select Interrupt On Next Error

SENSE

74 7 C	2000	Exit On Ready To Read
	2001	Exit On Not Ready To Read
	2002	Exit On Read Parity Error
	2003	Exit On No Read Parity Error
	2004	Exit On Read Length Error
	2005	Exit On No Read Length Error
	2006	Exit On End Of Tape Marker
	2007	Exit On Not End Of Tape Marker
	2400	Exit On Ready To Select
	2401	Exit On Not Ready To Select
	2402	Exit On Load Point
	2403	Exit On Not Load Point
	2404	Exit On Interrupt On Read Tape
	2405	Exit On No Interrupt On Read Tape
	2406	Exit On Read Program Error
	2407	Exit On No Read Program Error

SATELLITE EXTERNAL FUNCTION CODES

1604-A EXTERNAL FUNCTION CODES

OUTPUT SELECT

74 0 C	2500	Release Direct Selections
	2501	Select Write Control For 160
	2502	Release Write Control To 1604
	2503	Select Direct 1604 To 160
	2504	Select Action Request
	2520	Clear Communication Flag 2
	2540	Set Communication Flag 1
	2560	Clear Communication Flag 1

OUTPUT SENSE

74 7 C	2500	Exit On Write Control Available
	2501	Exit On Write Control Not Available
	2520	Exit On Communications Flag 2 Set
	2521	Exit On Communications Flag 2 Not Set
	2560	Exit On Communications Flag 1 Set
	2561	Exit On Communications Flag 1 Not Set

INPUT SELECT

74 0 C	2501	Select Read Control For 160
	2520	Release Read Control To 1604
	2503	Select Direct 160 To 1604
	2505	Release Interrupt

INPUT SENSE

74 7 C	2500	Exit On Read Control Available
	2501	Exit On Read Control Not Available
	2504	Exit On 160 Interrupt
	2505	Exit On No 160 Interrupt

160 EXTERNAL FUNCTION CODES

WRITE SELECT

6050	Release Action Request
6051	Set Communications Flag 2
6052	Release Write Control To 1604
6055	Clear Communications Flag 1
6056	Clear Communications Flag 2

READ SELECT

5051	Set Communications Flag 1
5052	Release Read Control To 1604
5053	Select Interrupt

STATUS RESPONSE

4XXX	Read Control Available
2XXX	Write Control Available
1XXX	Direct 160 To 1604
X4XX	Direct 1604 To 160
XXX2	160 Action Request
XXX1	Communications Flag 1 Set

APPENDIX VII

Magnetic Tape BCD Codes

Character	Code (Octal)	Character	Code (Octal)
A	61	2	02
B	62	3	03
C	63	4	04
D	64	5	05
E	65	6	06
F	66	7	07
G	67	8	10
H	70	9	11
I	71	&	60
J	41	-	40
K	42	(blank)	20
L	43	/	21
M	44	. (period)	73
N	45	$	53
O	46	*	54
P	47	, (comma)	33
Q	50	%	34
R	51	#	13
S	22	@	14
T	23	⌑	74
U	24	0 (numerical zero)	12
V	25	record mark	32
W	26	0 (minus zero)	52
X	27	0 (plus zero)	72
Y	30	group mark	77
Z	31	tape mark	17
0	12		
1	01		

APPENDIX VIII

Flexowriter Codes

UC	LC	CODE
A	a	30
B	b	23
C	c	16
D	d	22
E	e	20
F	f	26
G	g	13
H	h	05
I	i	14
J	j	32
K	k	36
L	l	11
M	m	07
N	n	06
O	o	03
P	p	15
Q	q	35
R	r	12
S	s	24
T	t	01
U	u	34
V	v	17
W	w	31
X	x	27

UC	LC	CODE
Y	y	25
Z	a	21
0	0	56
1	1	74
2	2	70
3	3	64
4	4	62
5	5	66
6	6	72
7	7	60
8	8	33
9	9	37
-	-	52
ɪ	/	44
(	)	54
+	,	46
=	.	42
:	;	50

CR	45
Upper Case (UC)	47
Lower Case (LC)	57
Back Space (BS)	61
Color Shift (CS)	02
Tabulate (TAB)	51
Stop	43
Space	04
Tape Feed	00
Delete	77

Note: 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

APPENDIX IX

Punched Card Codes

Char	Card	BCD	Char	Card	BCD	Char	Card	BCD	Char	Card	BCD
			+	12	60	---	11	40			20
1	1	01	A	12 1	61	J	11 1	41	/	0 1	21
2	2	02	B	12 2	62	K	11 2	42	S	0 2	22
3	3	03	C	12 3	63	L	11 3	43	T	0 3	23
4	4	04	D	12 4	64	M	11 4	44	U	0 4	24
5	5	05	E	12 5	65	N	11 5	45	V	0 5	25
6	6	06	F	12 6	66	O	11 6	46	W	0 6	26
7	7	07	G	12 7	67	P	11 7	47	X	0 7	27
8	8	10	H	12 8	70	Q	11 8	50	Y	0 8	30
9	9	11	I	12 9	71	R	11 9	51	Z	0 9	31
0	0	12	+ 0	12 0	72	- 0	11 0	52			
=	8,3	13	.	12 8,3	73	$	11 8,3	53	,	0 8,3	33
-	8,4	14	)	12 8,4	74	*	11 8,4	54	(	0 8,4	34

APPENDIX X

Input / Output Typewriter Codes

CHARACTERS UC	LC	CODE	CHARACTERS UC	LC	CODE
A	a	30	X	x	27
B	b	23	Y	y	25
C	c	16	Z	z	21
D	d	22	)	0	56
E	e	20	*	1	74
F	f	26	@	2	70
G	g	13	#	3	64
H	h	05	$	4	62
I	i	14	%	5	66
J	j	32	¢	6	72
K	k	36	&	7	60
L	l	11	½	8	33
M	m	07	(	9	37
N	n	06	_	-	52
O	o	03	?	/	44
P	p	15	"	'	54
Q	q	35	°	+	46
R	r	12	.	.	42
S	s	24	:	;	50
T	t	01	,	,	40
U	u	34	÷	=	02
V	v	17	tab	tab	51
W	w	31	space		04
Backspace		61	Carriage Return		45
Lower Case		57	Upper Case		47

APPENDIX XI

1612 Printer Codes

CHAR	CODE	CHAR	CODE	CHAR	CODE	CHAR	CODE
Blank	20	F	66	V	25	≤	15
0	12	G	67	W	26	‡	16
1	01	H	70	X	27	[	17
2	02	I	71	Y	30	]	32
3	03	J	41	Z	31	→	35
4	04	K	42	.	73	≡	36
5	05	L	43	-	40	~, ∧	37
6	06	M	44	+	60	% or ∨	52
7	07	N	45	=	13	$ or ¬	53
8	10	O	46	(	34	↑	55
9	11	P	47	)	74	↓	56
A	61	Q	50	/	21	>	57
B	62	R	51	*	54	<	72
C	63	S	22	,	33	≥	75
D	64	T	23	:	00	?	76
E	65	U	24	≠	14	;	77

In last column, codes ~ % $ appear if business application, ∧ ∨ ¬ for scientific application.

1604-A INSTRUCTIONS

			Page
ADD	Add	14	2-17
ADL	Add Logical	45	2-35
AJP	A Jump	22	2-27,30
ALS	A Left Shift	05	2-13
ARS	A Right Shift	01	2-13
DVF	Divide Fractional	27	2-20
DVI	Divide Integer	25	2-19
ENA	Enter A	10	2-25
ENI	Enter Index	50	2-26
ENQ	Enter Q	04	2-25
EQS	Equality Search	64	2-36
EXF	External Function	74	3-3
FAD	Floating Add	30	2-20
FDV	Floating Divide	33	2-23
FMU	Floating Multiply	32	2-22
FSB	Floating Subtract	31	2-21
IJP	Index Jump	55	2-16
INA	Increase A	11	2-25
INI	Increase Index	51	2-26
INT	Input Transfer	62	2-40
ISK	Index Skip	54	2-16
LAC	Load A, Complement	13	2-10
LDA	Load A	12	2-10
LDL	Load Logical	44	2-35
LDQ	Load Q	16	2-10
LIL	Load Index, L	53	2-12
LIU	Load Index, U	52	2-12
LLS	AQ Left Shift	07	2-14
LQC	Load Q, Complement	17	2-10
LRS	AQ Right Shift	03	2-13
MEQ	Masked Equality	66	2-37
MTH	Masked Threshold	67	2-37

			Page
MUF	Multiply Fractional	26	2-20
MUI	Multiply Integer	24	2-18
OUT	Output Transfer	63	2-40
QJP	Q Jump	23	2-28,31
QLS	Q Left Shift	06	2-14
QRS	Q Right Shift	02	2-13
RAD	Replace Add	70	2-38
RAO	Replace Add One	72	2-38
RSB	Replace Subtract	71	2-38
RSO	Replace Subtract One	73	2-39
SAL	Substitute Address, L	61	2-15
SAU	Substitute Address, U	60	2-15
SBL	Subtract Logical	46	2-35
SCA	Scale A	34	2-24
SCL	Selective Clear	41	2-34
SCM	Selective Complement	42	2-33
SCQ	Scale AQ	35	2-24
SEV	(not used)	77	
SIL	Store Index, L	57	2-12
SIU	Store Index, U	56	2-12
SLJ	Selective Jump	75	2-29,31
SLS	Selective Stop	76	2-29,31
SSH	Storage Shift	37	2-32
SSK	Storage Skip	36	2-32
SST	Selective Set	40	2-33
SSU	Selective Substitute	43	2-35
STA	Store A	20	2-11
STL	Store Logical	47	2-35
STQ	Store Q	21	2-11
SUB	Subtract	15	2-17
THS	Threshold Search	65	2-36
ZRO	(not used)	00	

CONTROL DATA

CORPORATION

501 PARK AVENUE, MINNEAPOLIS 15, MINNESOTA • FEDERAL 9-0411