

A
BELVIN

CONTROL DATA
924 COMPUTER
PROGRAMMING MANUAL

924 INSTRUCTIONS

		Page				Page	
ADD	Add	14	2-17	QLS	Q Left Shift	06	2-27
ADL	Add Logical	45	2-24	QRS	Q Right Shift	02	2-26
AJP	A Jump	22	2-36, 40	QTI	Q to Index	27	2-16
ALS	A Left Shift	05	2-27	RAD	Replace Add	70	2-29
ARS	A Right Shift	01	2-26	RAO	Replace Add One	72	2-29
ATI	A to Index	26	2-16	RSB	Replace Subtract	71	2-29
CMA	Complement A	52 1	2-25	RSO	Replace Subtract One	73	2-29
CMQ	Complement Q	52 2	2-25	SAL	Substitute Address	61	2-13
DVI	Divide	25	2-19	SBL	Subtract Logical	46	2-24
ENA	Enter A	10	2-15	SCA	Scale A	34	2-28
ENI	Enter Index	50	2-16	SCL	Selective Clear	41	2-23
ENQ	Enter Q	04	2-15	SCM	Selective Complement	42	2-22
EQS	Equality Search	64	2-33	SCQ	Scale AQ	35	2-28
EXF	External Function	74	3-3	SIL	Store Index	57	2-13
IJP	Index Jump	55	2-38	SKH	Skip High	30	2-31
INA	Increase A	11	2-21	SKL	Skip Low	31	2-31
INI	Increase Index	51	2-21	SLJ	Selective Jump	75	2-37, 40
INQ	Increase Q	56	2-21	SLS	Selective Stop	76	2-37, 40
ISK	Index Skip	54	2-31	SSH	Storage Shift	37	2-30
LAC	Load A Complement	13	2-11	SSK	Storage Skip	36	2-30
LDA	Load A	12	2-11	SST	Selective Set	40	2-22
LDL	Load Logical	44	2-24	SSU	Selective Substitute	43	2-24
LDQ	Load Q	16	2-11	STA	Store A	20	2-12
LIL	Load Index	53	2-13	STL	Store Logical	47	2-24
LLS	AQ Left Shift	07	2-26	STQ	Store Q	21	2-12
LQC	Load Q Complement	17	2-11	SUB	Subtract	15	2-17
LRS	AQ Right Shift	03	2-26	TAL	Tally	62	2-20
MEQ	Masked Equality	66	2-34	THS	Threshold Search	65	2-33
MTH	Masked Threshold	67	2-34	UJP	Unconditional Jump	32	2-37
MUI	Multiply	24	2-18	URJ	Unconditional Return Jump	33	2-41
PTS	Pattern Search	63	2-34	XAQ	Interchange A and Q	52 0	2-12
QJP	Q Jump	23	2-37, 40	XEC	Execute	60	2-41

BELVIN

CONTROL DATA
924 COMPUTER
PROGRAMMING MANUAL

CONTENTS

Chapter 1. Description

Physical Description	1-1
924 Characteristics	1-3
Logical Description	1-4
Storage Section	1-4
Input-Output	1-6
Arithmetic Section	1-6
Control Section	1-8

Chapter 2. Description of Instructions

Word Format	2-1
Execution Address	2-2
Address Modification	2-2
Sequential Execution of Instructions	2-4
Instructions	2-5
Instruction Execution Time	2-6
Order of Instructions	2-7
Full-Word Transmission	2-11
Address Transmission	2-13
Full-Word Arithmetic	2-17
Address Arithmetic	2-20
Logical	2-22
Shifting	2-26
Replace	2-29
Storage Test	2-30
Skip	2-31
Storage Search	2-33
Jumps and Stops	2-36
Normal	2-36
Return	2-39

Chapter 3. Input-Output

Methods of Data Exchange	3-1
Buffer Channels	3-1
Initiation and Control of Data Exchange	3-1
Console Input-Output Equipment	3-9
Paper Tape Reader	3-9
Paper Tape Punch	3-11
Internal EXF Select Instructions	3-13
Buffer Mode Instructions	3-13
Channel Clear Instructions	3-13
Internal EXF Sense Instructions	3-14
Arithmetic Fault Instructions	3-14
Paper Tape Reader	3-14
Paper Tape Punch	3-15
Console EXF Sense Instructions	3-16
Paper Tape Reader	3-16
Paper Tape Punch	3-16

Chapter 4. Operation

Description of Indicators and Control Switches	4-1
Reader and Punch Controls	4-6
1607 Controls and Indicators	4-7
Operation	4-8
Starting Operation With Pre-Stored Load Program	4-8
Starting Operation Without Pre-Stored Load Program	4-11
Shutting Down Equipment	4-11
Additional Procedures	4-12
Replacing Tape Roll at Punch	4-12
File Protection Ring	4-13
Emergency Procedures	4-13

Glossary

Appendix Section

I	Number Systems	1
II	Faults	12
III	Table of Powers of 2	14
IV	Octal-Decimal Integer Conversion Table	15
V	Octal-Decimal Fraction Conversion Table	19
VI	EXF and Character Codes	22
VII	924 Repertoire	35

FIGURES

Chapter 1. Description

1-1	Main Cabinet Interior	1-2
1-2	924 Input-Output System	1-7

Chapter 3. Input-Output

3-1	Instruction Searching	3-6
3-2	Seven Level Punched Paper Tape	3-10

Chapter 4. Operation

4-1	Center Panel of Console	4-1
4-2	Console Display	4-2
4-3	Manual Controls	4-5
4-4	Reader and Punch Controls	4-6
4-5	Paper Tape Reader	4-9
4-6	Tape Unit	4-10
4-7	Paper Tape Punch	4-12
4-8	File Protection Ring	4-13

TABLES

Chapter 1. Description

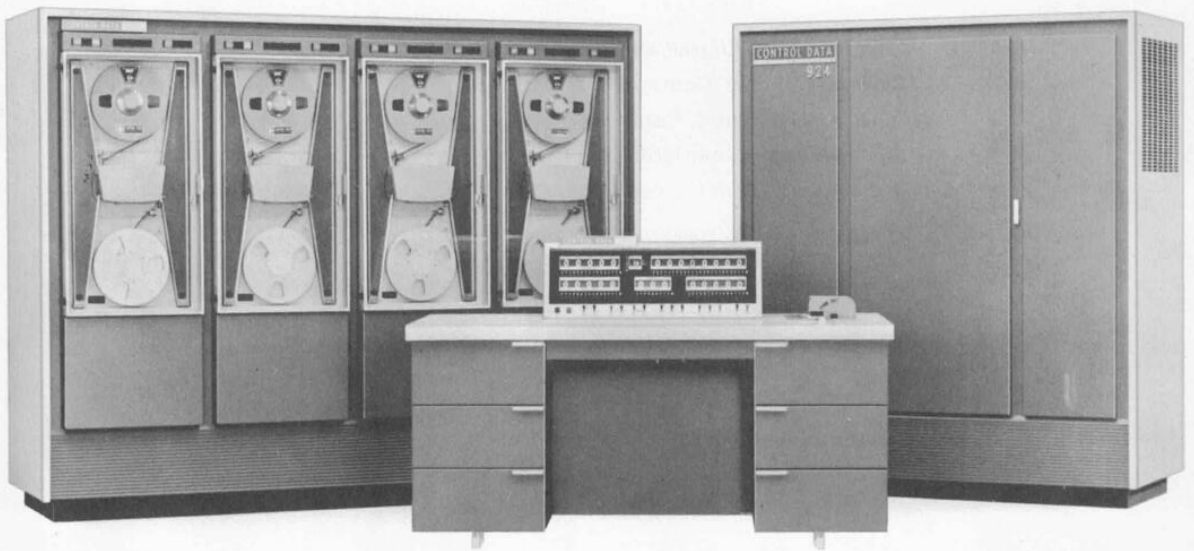
1-1	Operational Registers of the Computer	1-5
1-2	Arithmetic Properties of Registers	1-5

Chapter 3. Input-Output

3-1	Typical Interrupt Routine	3-8
-----	---------------------------	-----

Chapter 4. Operation

4-1	Conditions Indicated by Console Background Lights	4-3
4-2	Main Computer Controls	4-4
4-3	Reader and Punch Controls	4-6
4-4	1607 Controls and Indicators	4-7
4-5	Emergency Procedures	4-13



1

Description



The Control Data 924 Computer is a general purpose digital computer designed to solve both business and scientific problems. The reliability of high-speed transistor amplifier circuits and the efficiency of parallel operations are combined in the 924 to produce exceedingly fast computation and transfer speeds. Greater speed, reliability, and efficiency coupled with modular construction and large storage capacity results in an extremely versatile and powerful computer having highly flexible systems applications.

PHYSICAL DESCRIPTION

The basic 924 installation consists of two units; the console and main computer cabinet. An installation may also include, as optional equipment, any input-output devices capable of communication with the 160 and/or 1604 computer such as the 1607 Magnetic Tape System, the 1610 Punched Card Control Unit, the 1612 High Speed Line Printer and the 161 Typewriter.

The console contains the operator's display panel, controls, paper tape punch and paper tape reader.

The main cabinet holds four chassis hinged for easy access (figure 1-1). Each chassis holds part of the core storage assembly and the pluggable printed circuit cards. Cabinet dimensions, to the nearest inch, are: Height 5 feet 8 inches, Depth 2 feet 2 inches, Length 5 feet 2 inches, and Weight approximately 1430 pounds.

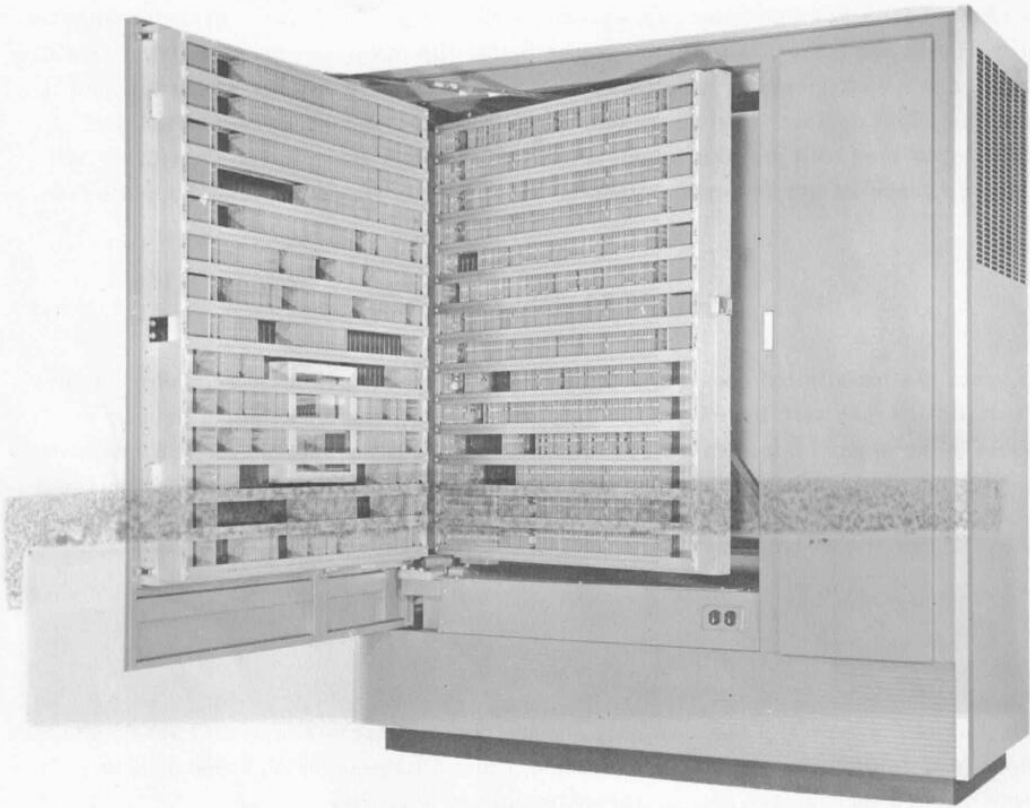


Figure 1-1. Main Cabinet Interior

924 CHARACTERISTICS

Stored-program, general-purpose digital computer

Parallel mode

24-bit word, 1 instruction per word

Single-address logic

Operation code 6 bits
Index designator 3 bits
Execution address 15 bits

Six index registers

Indirect addressing

Magnetic core storage

8,192 24-bit words , 16K, 32K

Two independent 4096 word banks alternately phased *< odd even*

5.3 μ sec effective cycle time (representative program)

6.4 μ sec total cycle time

Input-output

Three 48-bit buffer input channels, compatible with 1604 peripheral equipment, 12 lower-order bits compatible with 160 peripheral equipment.

Three 48-bit buffer output registers, compatible with 1604 peripheral equipment, 12 lower-order bits compatible with 160 peripheral equipment.

Program interrupt

Console, includes:

Photoelectric paper tape reader
Paper tape punch
Register contents displayed in octal

Flexible 64 instructions

Fixed point arithmetic
Logical and masking operations
Indexing
Input-output
Conditional and unconditional
Jumps and stops
Multiple precision capability (accumulator and auxiliary register operate as a single double-length register)
Storage searching

Binary arithmetic

Modulus $2^{24}-1$ (one's complement)
Parallel addition in 1.2 μ sec without access

Real-time clock

Completely solid-state

Diode logic
Transistor amplifiers
Magnetic core storage

Small size

Less than 400 square feet of floor space required

Low power consumption

Optional features

Increase standard 8,192 words of magnetic core storage to 16,384 or 32,768 words

Price	New Price	Size	Rent
\$ 265K	\$ 180K	8K	\$ 8K/month
300	216	16	9
370	288	32	11

LOGICAL DESCRIPTION

The computer 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, computer operation may be divided into four major sections. INPUT-OUTPUT provides communication between the computer and the external equipment; ARITHMETIC performs the arithmetic and logical operations required for executing instructions; STORAGE provides internal storage for data and instructions; and 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.

The registers in the computer are identified by letters. The operational registers (table 1-1) usually hold the end result of an operation. Their contents are displayed on the console and may be manually changed by the operator. The arithmetic properties of these registers are detailed in table 1-2. The transient registers used in formulating the result are secondary registers. They are not displayed and cannot be manually changed.

STORAGE SECTION

The magnetic core storage section of the 924 Computer provides high-speed, random access storage for 8,192 words. It consists of two independent storage units each with a capacity of 4096 words. These units operate together during the execution of a stored program and thus are considered as one 8,192 word storage system.

A word is 24 bits in length and is used as a 24-bit instruction or a 24-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 microseconds. 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 5.3 microseconds.

TABLE 1-1. OPERATIONAL REGISTERS OF THE COMPUTER

Register	Function
A	Arithmetic
Q	Auxiliary Arithmetic
B ¹ through B ⁶	Index registers (six)
P	Program Address
U	Program Control

TABLE 1-2. ARITHMETIC PROPERTIES OF REGISTERS

Register	No. of stages	Modulus	Complement Notation*	Arithmetic	Result
A	24	$2^{24}-1$	one's	subtractive	signed**
Q	24	$2^{24}-1$	one's		signed
P	15	2^{15}	two's	additive	unsigned
U	15	2^{15}	two's	subtractive	unsigned

B¹-B⁶ 15 Non-Arithmetic Registers

* Refer to Appendix

**The result of an arithmetic operation in A satisfies $A \leq 2^{23}-1$ since A always is treated as a signed quantity. When the result in A is zero, it is always represented by 000...00 except when 111...11 is added to 111...11. In this case, the result is 111...11 (negative zero).

INPUT-OUTPUT

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 in 24-bit words.

The computer communicates with external equipment through six independent buffer channels which provide for the normal exchange of data (figure 1-2).

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 pairs. It is possible to connect as many as five different equipments to any given pair of channels. All six buffer channels may be concurrently transmitting information. However, only one external equipment can use any one buffer channel at any given instant.

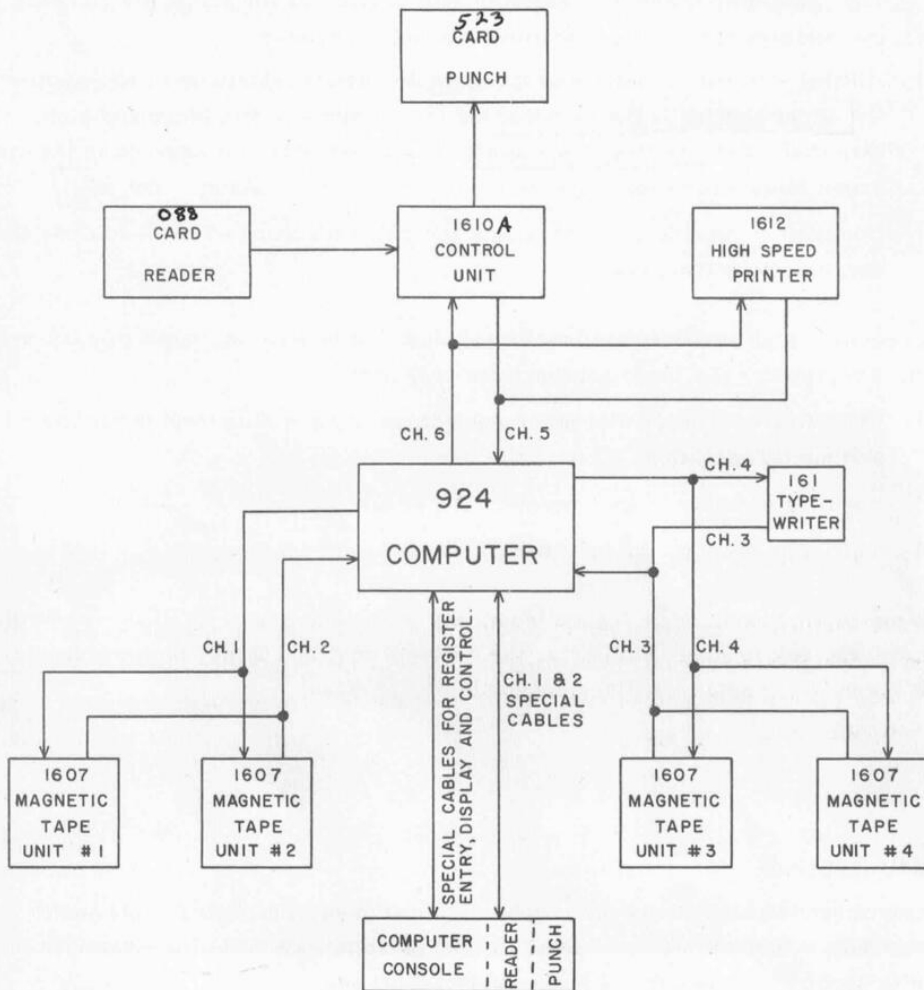
In the 924 computer, input-output operations are independent of the main computer program. When data is transmitted, the main computer program initiates an automatic cycle which buffers data to and from computer storage. The main computer program then continues while the actual buffering of data is carried out independently and automatically.

This process of asynchronous input-output operations will be termed a buffer. Buffer transmissions employ independent access to computer storage so that computation continues while the external equipment is loading or unloading information from computer storage. The rate of exchange is, in most cases, dictated by the external equipment.

ARITHMETIC SECTION

The arithmetic section of the 924 computer consists of two operational registers, A and Q, and several secondary registers.

The A register (accumulator) is the principal arithmetic register. Some of the more important functions of A are:



Sample
 Figure 1-2. 924 Input-Output System
 A

- 1) All arithmetic and logical operations use the A register in formulating a result. The A register is the only register with provisions for adding its contents and the contents of a storage location or another register.
- 2) Shifting - A may be shifted to the right or left separately or in conjunction with Q. Right shifting is open-ended; the lowest bits are discarded and sign extended. Left shifting is circular; the highest order bit appears in the lowest order stage after each shift; all other bits move one place to the left.
- 3) Control for conditional instructions - A holds the word which conditions jump and search instructions.

The Q register is an auxiliary arithmetic register and is generally used in conjunction with the A register. The principal functions of Q are:

- 1) Providing temporary storage of contents of A while A is used for another arithmetic operation.
- 2) Forming a double-length register, AQ or QA.
- 3) Shifting to the right or left, separately or in conjunction with A.

Both A and Q may load, or be loaded from, any of the six index registers without the use of storage references. Similarly, the contents of A and Q may be interchanged to permit the use of Q as an auxiliary accumulator.

CONTROL SECTION

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

The control section acquires an instruction from storage, interprets it, and sends the necessary commands to other sections. The composition of a 924 instruction is shown below.

Instruction Format

f (6 bits)	b (3 bits)	m, y, or k or unused (15 bits)
<u>operation</u> <u>code</u>	<u>index</u> <u>designator</u>	<u>base execution</u> <u>address</u>

Each of the 64 instructions has an operation code which specifies the operation to be performed. This code is usually written as a 2 octal digit numeric code or as a mnemonic code (see appendix).

The index designator generally is used for address modification; it specifies one of the six index registers whose contents are to be added to the base execution address. The index designator may also condition jump and stop instructions, specify indirect addressing, or further distinguish between two unlike instructions having the same operation code.

The execution address may be used as an address of an operand, m; as an operand, y; or as a shift count, k.

The eight operational registers in the control section are P, U and B¹ through B⁶.

The program address counter (P) is a two's complement additive register. It provides program continuity by generating in sequence the storage addresses which contain the individual instructions. Usually at the completion of each instruction, the count in P is advanced by one to specify the address of the next instruction.

The program control register, U, holds an instruction while it is executed. After executing an instruction, an exit, jump exit or skip exit is performed. After executing an instruction, an exit, jump exit or skip exit is performed. An exit advances the count in P by one and executes the next instruction specified by the contents of P. A jump exit executes the instruction at the storage location specified by the execution address of the jump instruction. The execution address is, in this case, entered into P and specifies the starting location of a new sequence of instructions. A skip exit advances the count in P by two, bypassing the next sequential instruction and executing the following one.

what part of inst?

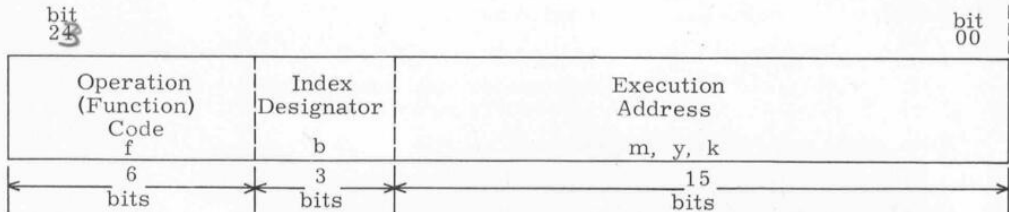
ans mod by 6

Each of the six index registers, B¹ through B⁶, provides storage for quantities which are used in a variety of ways, depending on the instruction. The B registers have no provisions for arithmetic operations. In the majority of instructions the B registers hold quantities to be added to the execution address. All address modifications are performed in A. For search instructions, the contents of a B register indicate the number of words to be search. This quantity is transmitted to U, and reduced one count for each word searched.

Description of Instructions

WORD FORMAT

A computer word consists of 24 bits and may be interpreted as a 24-bit instruction or data word. Each instruction is composed of three parts or codes: operation code, index designator, and execution address.



Code	Range	Description
Operation f	01 - 76 ₈	Specifies the operation to be performed. Codes 00, 52.3-52.6 and 77 are interpreted as faults which stop computer operation.
Execution Address m, y, k	00000 through 77777 ₈	Used in one of three ways: <ol style="list-style-type: none"> 1) as a shift count k, or 2) as an operand address, m, or 3) as an operand, y.
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 (Refer to 22, 23, 74-76 instructions for exceptions)

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 left 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 a storage reference is made to the location designated by the execution address. The lower-order 18 bits of the 24-bit word are read from storage and 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:

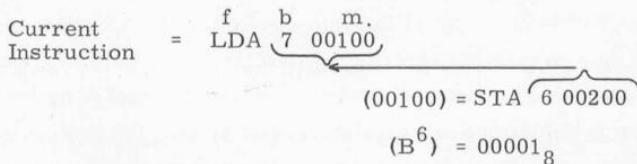
- 1) No Address Modification f b m
LDA 0 address

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



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 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. $M = 00200_8 + (00001_8) = 00201_8$

Sequential Execution of Instructions

Example:

	f	b	m
(00300) = LDA	0		00310
(00301) = ADD	1		00210
		(B^1)	$= 00101_8$
(00302) =		

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 24-bit word is read from address 00300 and entered into U. Computer operation is now dependent upon the interpretation of the 24-bit instruction in U.

The operation code, LDA, and the index designator, 0, are translated. The function of the 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 contents of P are increased by one and the next instruction is read from storage address 00301. This instruction is now translated in U. 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 again increased by one and the instruction at address 00302 is read from storage and executed. This sequence is continued unless a jump or skip instruction is executed (see page 1-9).

INSTRUCTIONS

The 64 computer instructions are described on the following pages, (EXF instructions are discussed in detail in chapter three). The title line contains the mnemonic code and format, name, and average execution time of the instruction. Abbreviations and symbols are defined as follows:

A	A register (accumulator)
A_n	The binary digit in position n of the A register
→	Transmit to
b	Index designator
B^b	Designated index register
Exit	Proceed to next instruction
j	The condition designator for jump and stop instructions
k	Unmodified shift count
K	Modified shift count. $K = k + (B^b)$
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
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 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 the base execution address is to be modified, the instruction execution time will be increased if consecutive instructions and operand addresses are used. There will be no increase in execution time if non-consecutive addresses are used for instructions and operand words.

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 an input-output request exists, the request will, in most cases, be processed before the next instruction is executed. (Refer to chapter three.)

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

All times in error. See
correction sheet of Rob
Burreason, 4/2/62.

Octal Code	Mnemonic Code	Name	Timing*
FULL-WORD TRANSMISSION			
12	LDA	LOAD A	9.9
13	LAC	LOAD A COMPLEMENT	9.9
16	LDQ	LOAD Q	9.9
17	LQC	LOAD Q COMPLEMENT	9.9
20	STA	STORE A	9.8
21	STQ	STORE Q	9.8
520	XAQ	INTERCHANGE A AND Q	6.2
ADDRESS TRANSMISSION			
53	LIL	LOAD INDEX	6.2 9.2
57	SIL	STORE INDEX	6.2 9.2
61	SAL	SUBSTITUTE ADDRESS	6.0 9.8
10	ENA	ENTER A	6.0
04	ENQ	ENTER Q	6.0
26	ATI	A TO INDEX	6.0
27	QTI	Q TO INDEX	6.0
50	ENI	ENTER INDEX	6.2
FULL-WORD ARITHMETIC			
14	ADD	ADD	6.0 9.9 12.3
15	SUB	SUBTRACT	9.9
24	MUI	MULTIPLY	12.2 27.9 + .8n*
25	DVI	DIVIDE	38.0
62	TAL	TALLY	14.8

* Timing is average execution time in usec

* n = Number of ones in multiplier

ORDER OF INSTRUCTIONS

ADDRESS ARITHMETIC

11	INA	INCREASE A	7.0
51	INI	INCREASE INDEX	7.0
56	INQ	INCREASE Q	7.0

LOGICAL

40	SST	SELECTIVE SET	9.9
42	SCM	SELECTIVE COMPLEMENT	9.9
41	SCL	SELECTIVE CLEAR	9.9
43	SSU	SELECTIVE SUBSTITUTE	9.9
44	LDL	LOAD LOGICAL	9.9
45	ADL	ADD LOGICAL	9.9
46	SBL	SUBTRACT LOGICAL	9.9
47	STL	STORE LOGICAL	9.8
52 1	CMA	COMPLEMENT A	6.2
52 2	CMQ	COMPLEMENT Q	6.2

SHIFTING

01	ARS	A RIGHT SHIFT	$7.2 + .4s^*$
02	QRS	Q RIGHT SHIFT	$7.2 + .4s^*$
03	LRS	AQ RIGHT SHIFT	$7.2 + .4s^*$
07	LLS	AQ LEFT SHIFT	$7.2 + .4s^*$
06	QLS	Q LEFT SHIFT	$7.2 + .4s^*$
05	ALS	A LEFT SHIFT	$7.2 + .4s^*$
34	SCA	SCALE A	$3.8 + .4s^*$
35	SCQ	SCALE AQ	$3.8 + .4s^*$

* s = Number of places shifted

ORDER OF INSTRUCTIONS

REPLACE

70	RAD	REPLACE ADD	15.9
71	RSB	REPLACE SUBTRACT	15.9
72	RAO	REPLACE ADD ONE	15.9
73	RSO	REPLACE SUBTRACT ONE	15.9

STORAGE SEARCH

64	EQS	EQUALITY SEARCH	6.9 + 5.2r*
65	THS	THRESHOLD SEARCH	6.9 + 6.0r*
66	MEQ	MASKED EQUALITY	6.9 + 6.0r*
67	MTH	MASKED THRESHOLD	6.9 + 6.0r*
63	PTS	PATTERN SEARCH	6.9 + 6.0r*

} 7.2 +
4.8r

STORAGE TEST

36	SSK	STORAGE SKIP	9.2
37	SSH	STORAGE SHIFT	15.9

SKIP

54	ISK	INDEX SKIP	7.0
30	SKH	SKIP HIGH	6.4
31	SKL	SKIP LOW	6.4

JUMPS AND STOPS (Normal)

22	AJP	A JUMP	7.7
23	QJP	Q JUMP	7.7
75	SLJ	SELECTIVE JUMP	7.9
76	SLS	SELECTIVE STOP	7.9
32	UJP	UNCONDITIONAL JUMP	4.8
55	IJP	INDEX JUMP	7.0

r = Number of additional words searched

ORDER OF INSTRUCTIONS

JUMPS AND STOPS (Return)

AJP	A JUMP	7.7	
QJP	Q JUMP	7.7	
SLJ	SELECTIVE JUMP	7.9	
SLS	SELECTIVE STOP	7.9	
33	URJ	UNCONDITIONAL RETURN JUMP	4.8
60	XEC	EXECUTE	8.2 + x usec for instruction at M

74 EXP EXTERNAL FUNCTION

FULL-WORD TRANSMISSION

- 1) In Full-Word Transmission instructions, a 24-bit operand or data word is used in executing the instruction.
- 2) All modes of address modification apply to the full-word transmission instructions (except XAQ).
- 3) With the exception of XAQ, one storage reference is made during the execution of full-word transmission instructions. If indirect addressing is designated, at least two storage references are made. No storage reference is made during the execution of the XAQ instruction.

LDA b m Load A 9.9 us
Replaces the contents of A with a 24-bit operand contained in storage location M. The contents of M remain unchanged.

LAC b m Load A complement 9.9 us
Replaces the contents of A with the complement of a 24-bit operand contained in storage location M. The contents of M remain unchanged.

LDQ b m Load Q 9.9 us
Replaces the contents of Q with a 24-bit operand contained in the storage location M. The contents of M remain unchanged.

LQC b m Load Q Complement 9.9 us
Replaces the contents of Q with the complement of a 24-bit operand contained in storage location M. The contents of M remain unchanged.

STA b m Store A 9.8 us
Replaces the contents of the designated storage location, M, with the contents of A. The initial contents of A remain unchanged.

STQ b m

Store Q

9.8 us

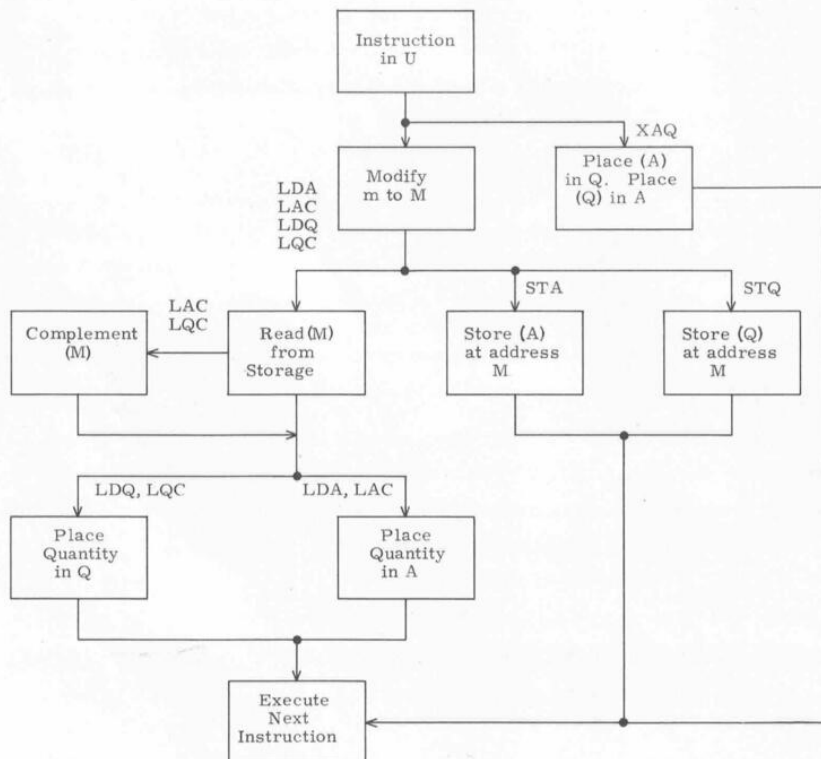
Replaces the contents of the designated storage location, M, with the contents of Q. The initial contents of Q remain unchanged.

XAQ 0 m

Interchange A and Q*

6.2 us

Places the contents of the A register into the Q register and at the same time places the contents of the Q register into the A register. The result of this instruction is that the contents of the A and Q registers are interchanged. The m portion of this instruction is not used.



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

* If the index designator does not have the indicated value, the instruction will not be executed.

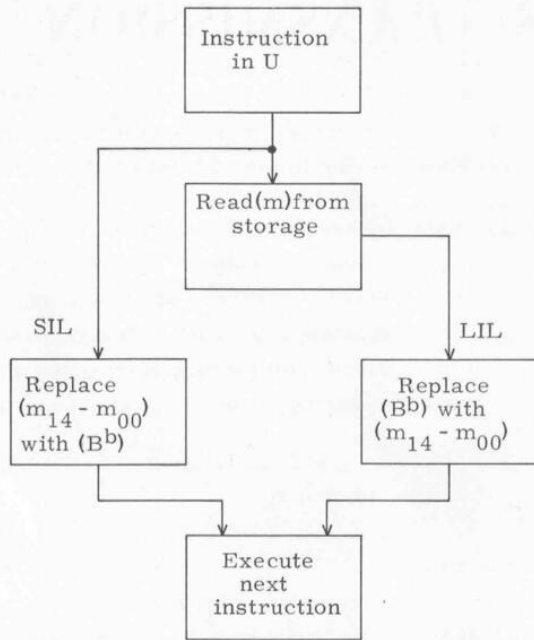
ADDRESS TRANSMISSION

- 1) In the Address Transmission instructions, only the address portion (the lower 15 bits) of the word is used.
- 2) One storage reference is made during LIL, SIL, and SAL instructions. If indirect addressing is designated, at least two storage references are required. If indirect addressing is designated in the ENI instruction, one storage reference is required. No storage references are made for the remaining instructions.
- 3) Address modification applies to the SAL, ENQ and ENA instructions only.

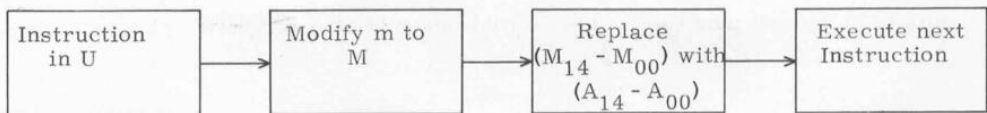
LIL b m Load Index 9.3 us
Replaces the contents of the designated index register with the address portion of storage location m. If b = 0 this instruction becomes a pass (do nothing) instruction. Initial contents of m remain unchanged.

SIL b m Store Index 9.8 us
Replaces the 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, the address portion of m is set to all 1's. Initial contents of B^b remain unchanged; initial contents of m are changed.

SAL b m Substitute Address 9.8 us
Replaces the 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.



LIL and SIL



SAL

ENA b y

Enter A

7.0 us

The 15-bit operand, Y, is entered into the A register and its highest order bit (sign bit) is extended in the remaining 9 bits. The largest positive 15-bit operand that can be entered into A is 37777_8 ($2^{14}-1$) and the "0" sign bit will be duplicated in each of the upper 9 bits. Negative zero will be formed in A if:

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

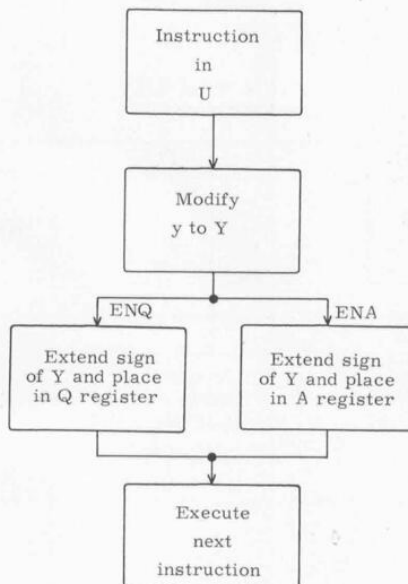
ENQ b y

Enter Q

7.0 us

The 15-bit operand, Y, is entered into Q and its highest order bit (sign bit) is extended in the remaining 9 bits. The largest positive 15-bit operand that can be entered into Q is 37777_8 ($2^{14}-1$) and its "0" sign bit will be duplicated in each of the upper 9 bits. 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 and ENQ**

ATI b m

A to Index

7.0 us

This instruction transmits the lower order 15 bits of the A register to the index register specified by the index designator. The address portion of the instruction, m, is not used in this instruction. The contents of the A register are unchanged by this instruction.

QTI b m

Q to Index

7.0 us

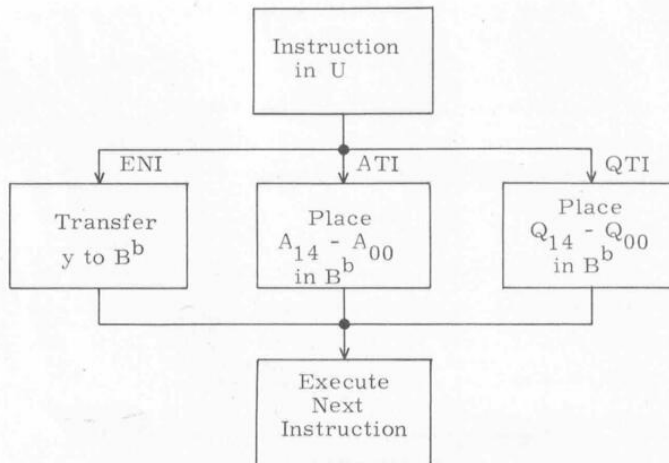
This instruction transmits the lower order 15 bits of the Q register to the index register specified by the index designator. The address portion of the instruction, m, is not used in this instruction. The contents of the Q register are unchanged as a result of the instruction.

ENI b y

Enter Index

6.2 us

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

**ATI, QTI, and ENI**

FULL-WORD ARITHMETIC

- 1) In Full-Word Arithmetic instructions, a 24-bit operand is used in executing each instruction. The TAL instruction is unique in this category in that an operand, as such, is not used. The TAL instruction examines a 24-bit quantity in the A register.
- 2) All modes of address modification apply to these instructions except TAL.
- 3) One storage reference is made for each instruction except TAL unless indirect addressing is designated. In this case, at least two references are made.
- 4) If the capacity of the A register $\pm (2^{23}-1)$ is exceeded during the execution of the instructions an arithmetic overflow fault is produced. When executing the DVI instruction, if the result exceeds the capacity of the A register $\pm (2^{23}-1)$ a divide fault is produced. (Refer to appendix.)

ADD b m

Add

9.9 us

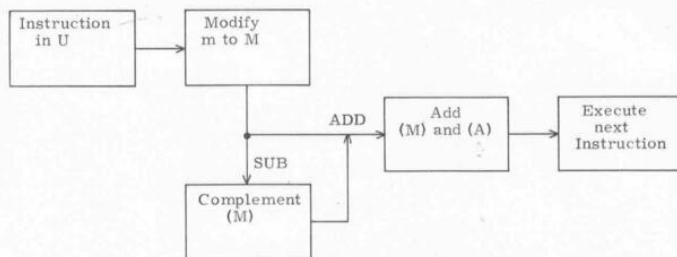
Adds a 24-bit operand obtained from storage location M to (A). A negative zero may be produced by this instruction if (A) and (M) are initially negative zero. The contents of storage address M remain unchanged.

SUB b m

Subtract

9.9 us

Obtains a 24-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.



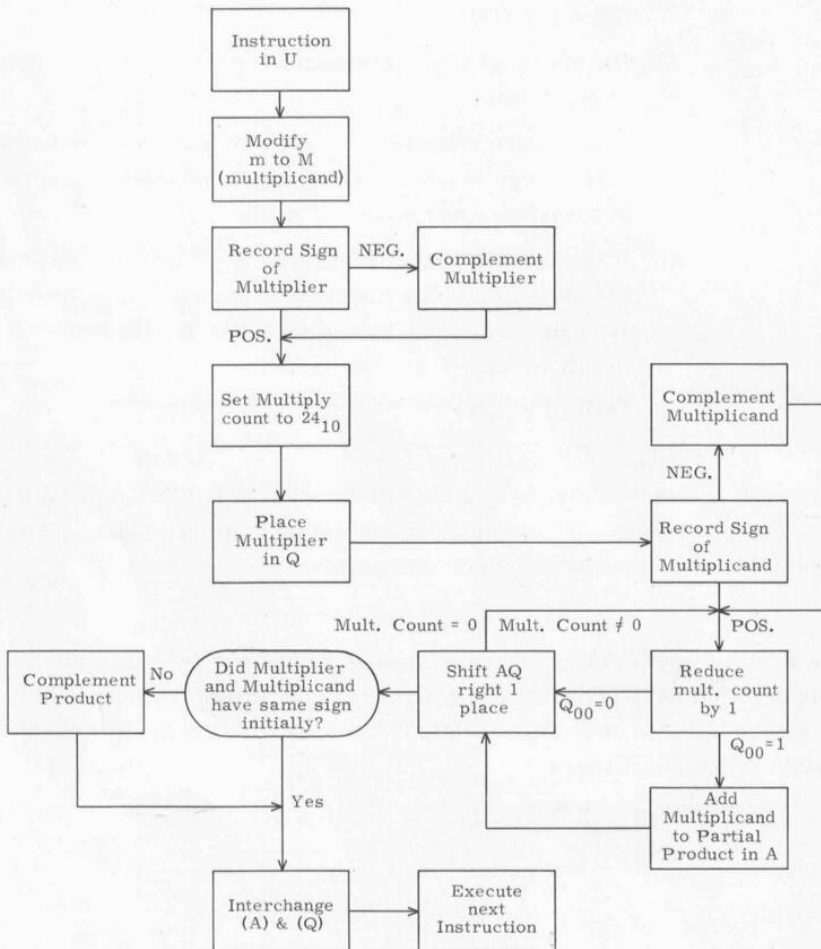
ADD and SUB

MUI b m

Multiply

27.9 + .8n*

Forms a 48-bit product from two 24-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 48-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**

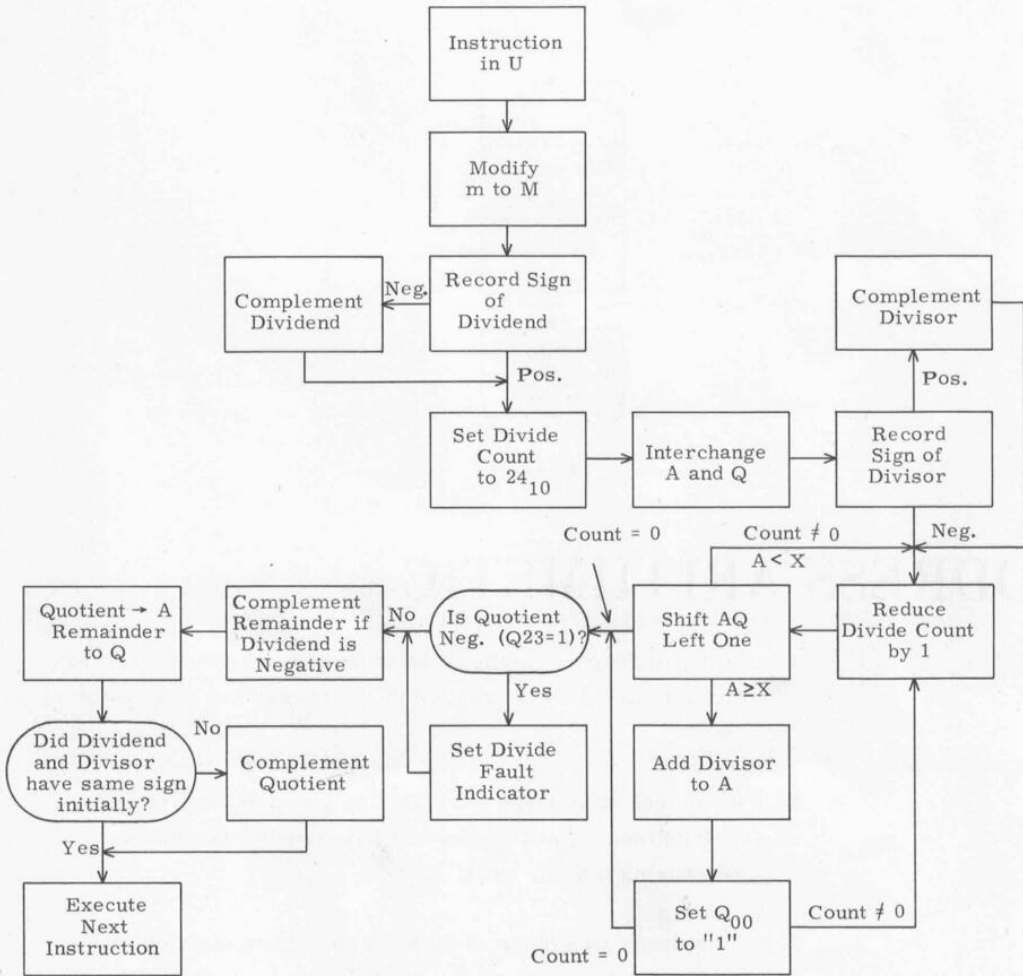
* n = Number of ones in multiplier

DVI b m

Divide

38.0 us

Divides a 48-bit integer dividend by a 24-bit integer divisor. The 48-bit dividend must be formed in the QA register prior to executing the instruction. If a 24-bit dividend is loaded into A, the sign of the dividend in A must be extended throughout Q. The 24-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.



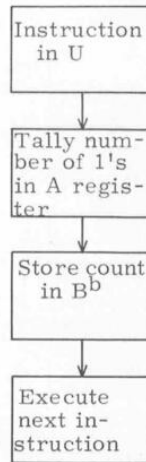
DVI

TAL b m

Tally

14.8 us

This instruction counts the number of ones in the A register and stores the count in the index register specified by the designator b; the contents of the A register remain unchanged. The m portion of the instruction is not used.



TALLY

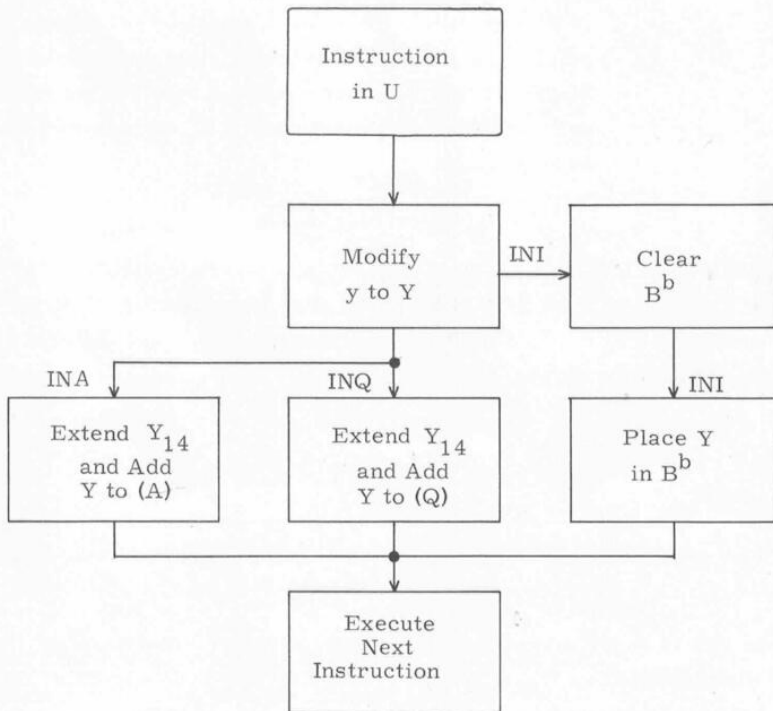
ADDRESS ARITHMETIC

- 1) In the Address Arithmetic instructions, only the lower 15-bits (or the address portion) of the operand or data word are used.
- 2) All modes of address modification apply to the INA and INQ instructions. In executing the INI instruction, indirect addressing may be 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.

INA b y Increase A 7.0 us
 Adds Y to A. The 15-bit operand Y with its highest order bit is extended in the remaining 9 bits and added to A. Y is thus considered a 15-bit signed operand, with sign extended.

INI b y Increase Index 7.0 us
 Increases (B^b) by the operand y. If the b designator is zero, this instruction becomes a pass or do nothing instruction.

INQ b y Increase Q 7.0 us
 Adds Y to Q. The 15-bit operand Y, with its highest order bit extended in the remaining 9 bits, is added to Q. Y is thus considered as a 15-bit signed operand, with sign extended.



INA, INI, and INQ

LOGICAL

- 1) All modes of address modification apply to these instructions except CMA and CMQ.
- 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.

$$\begin{array}{ll} 0 \times 0 = 0 & 1 \times 0 = 0 \\ 0 \times 1 = 0 & 1 \times 1 = 1 \end{array}$$

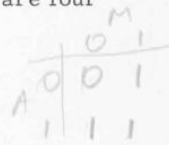
- 3) A logical product is used, in many cases, to select specific portions of an operand for entry into another operation. For example, if only a specific portion of an operand in storage is to be added to (A), as the operand passes through the exchange register (X, a secondary register) it is subjected 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.

SST b m

Selective Set = *Logical Sum* = + = 9.9 us

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

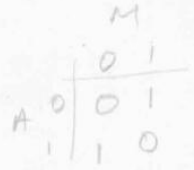


SCM b m

Selective Complement = *Mod 2 Sum* = ⊕ = 9.9 us

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



SCL b m

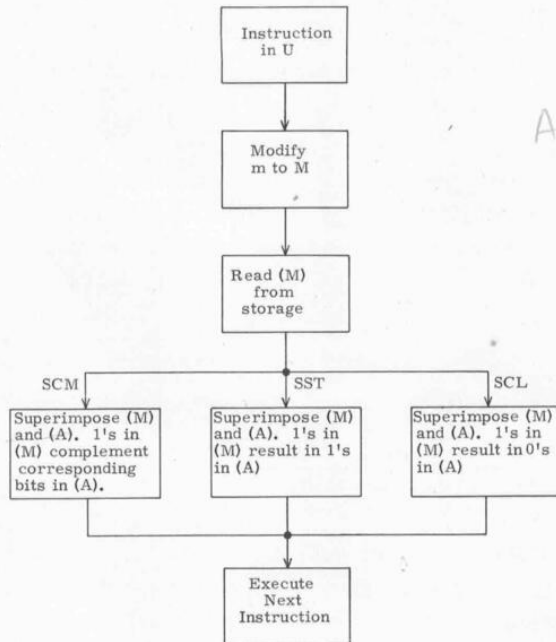
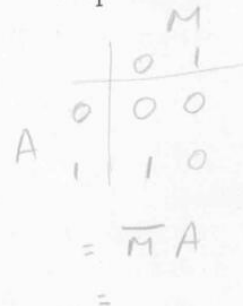
Selective Clear

9.9 us

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



SCM, SST, and SCL

$$(Q \cdot M \oplus A) \rightarrow A$$

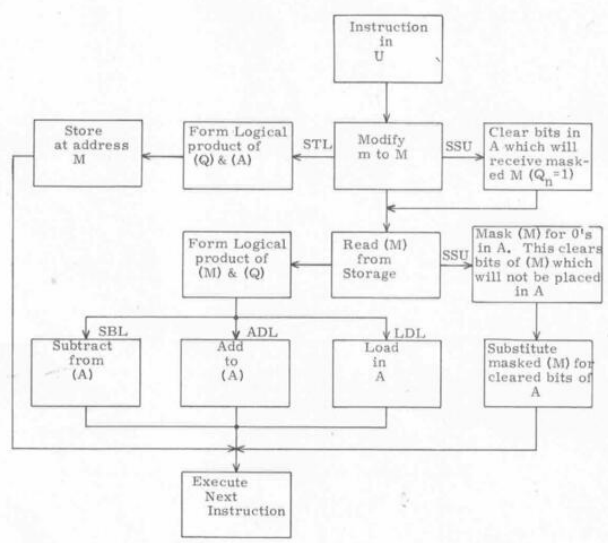
SSU b m Selective Substitute 9.9 us
 Substitutes selected portions of an operand at storage address M into the A register where there are corresponding "1's" in the Q register (mask). The portions of A not masked by "1's" in Q are left unmodified.

LDL b m Load Logical 9.9 us
 Loads A with the logical product of Q and the designated storage location, M.

ADL b m Add Logical 9.9 us
 Adds to A the logical product of Q and the quantity in location M. Once the logical product is formed addition follows normal rules.

SBL b m Subtract Logical 9.9 us
 Subtracts from A the logical product of the Q register and the quantity in storage location M. When the logical product is formed, the subtraction proceeds in the normal manner.

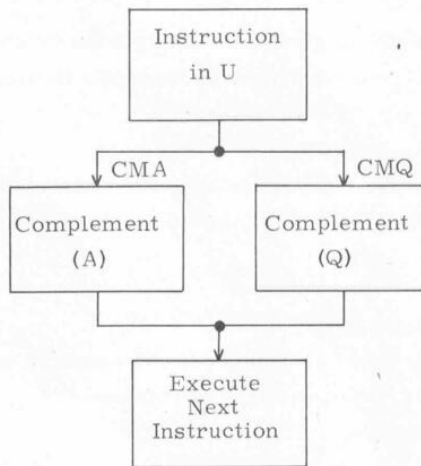
STL b m Store Logical 9.8 us
 Replaces the bits in location M with the logical product of Q and A registers. Neither (A) or (Q) are modified.



ADL, LDL, SBL, SSU, and STL

CMA 1 m Complement A* 6.2 us
This instruction complements the contents of the A register. The address portion of this instruction, m, is not used.

CMQ 2 m Complement Q* 6.2 us
This instruction complements the contents of the Q register. The address portion of this instruction, m, is not used.



CMA and CMQ

* If the index designator does not have the indicated value, the instruction will not be executed.

SHIFTING

- 1) All modes of address modification apply to all these instructions, except SCA and SCQ.
- 2) If the modified shift count, K, is greater than 63_{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 will be completed before an input, output or interrupt request will be processed. (See chapter three.)

ARS b k A Right Shift $6.2 + .4s^*$
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 23_{10} since the register is now an extension of the sign bit.

QRS b k Q Right Shift $6.2 + .4s^*$
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 23_{10} since the register is now an extension of the sign bit.

LRS b k Long Right Shift $6.2 + .4s^*$
Shifts contents of AQ to the right K places as one 48-bit register. The A register is considered as the leftmost 24 bits and the Q register as the rightmost 24 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 47_{10} since AQ is now an extension of the sign of A.

LLS b k Long Left Shift $6.2 + .4s^*$
Shifts contents of AQ to the left K places, left circular, as one 48-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 48_{10} returns AQ to its original state.

* s = Number of positions shifted

QLS b k

Q Left Shift

6.2 + .4s*

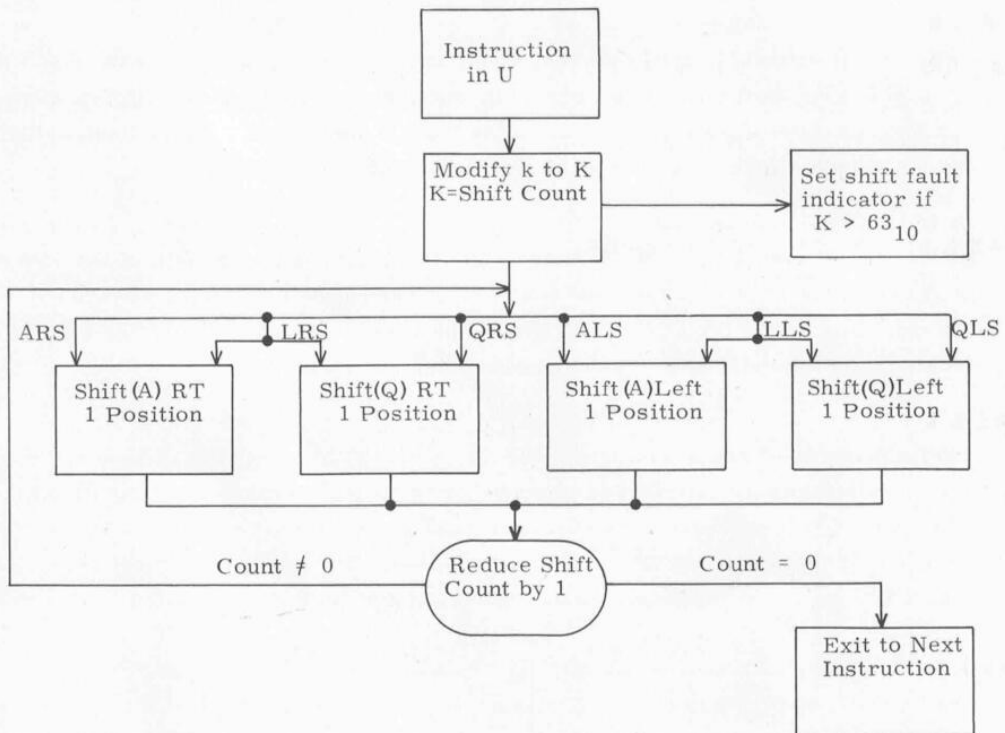
Shifts contents of Q to the left K places, left circular. The higher order bits of Q replace the lower order bits. The largest practical shift count 24_{10} returns the register to its original state.

ALS b k

A Left Shift

6.2 + .4s*

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 24_{10} returns the register to its original state.

**Shift Instructions**

In the SCA and SCQ instructions:

- 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 the $(A)_i$ 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.
- 6) The shift fault indicator is not affected by these instructions.

SCA b k

Scale A

$$3.8 + .4s^*$$

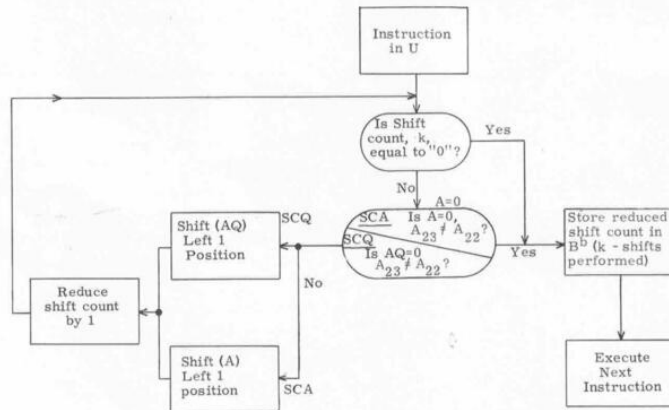
Shifts A left circularly until the most significant digit is to the right of the sign bit or until $k = 0$. Shift count k is reduced by one for each shift and terminates when $k = 0$ or the most significant digit is to the right of the sign bit. Upon termination the count (scale factor) is entered in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

SCQ b k

Scale AQ

$$3.8 + .4s^*$$

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

REPLACE

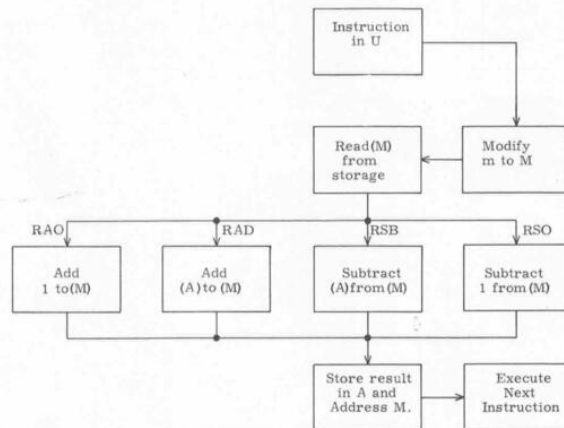
- 1) All modes of address modification apply to these instructions.
- 2) During the execution of the replace instructions, two storage references are made. If indirect addressing is designated, at least three references are made.
- 3) If the capacity of the A register $\pm (2^{23}-1)$ is exceeded during the execution of the following instructions, an arithmetic overflow fault is produced. (Refer to appendix.)

RAD b m Replace Add 15.9 us
 Obtains a 24-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 b m Replace Subtract 15.9 us
 Subtracts (A) from (M) and places the result in both the A register and location M.

RAO b m Replace Add One 15.9 us
 Replaces the operand in storage location M with its original value plus one. The result is also placed in A.

RSO b m Replace Subtract One 15.9 us
 Replaces the operand in storage location M with its original contents minus one. The difference is also left in A.



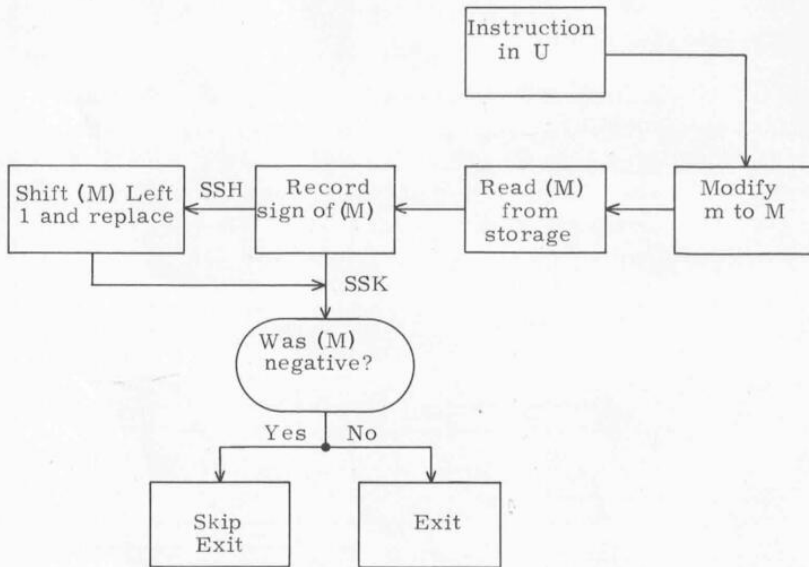
Replace

STORAGE TEST

- 1) All modes of address modification may be used in 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 b m Storage Skip 9.2 us
 Senses the sign bit of the operand in storage location M. If the sign is negative, a skip exit is taken. If the sign is positive, an exit is taken. The contents of the operational registers are left unmodified.

SSH b m Storage Shift 15.9 us
 Senses the sign bit of the quantity in storage location M. If the sign bit is negative a skip exit is taken, and if the quantity is positive an exit is taken. In either case the quantity is shifted left circularly one bit before the exit. The contents of the operational registers are left unmodified.



SSH and SSK

SKIP

- 1) Indirect addressing is the only mode of address modification recognized by these instructions.
- 2) No storage reference is made unless indirect addressing is specified in which case at least one reference will be made.

ISK b y

Index Skip

7.0 us

Compares (B^b) with y . If the two quantities are equal, B^b is cleared and a skip exit is performed. If the quantities are unequal, (B^b) is increased one count in the U register and an exit is performed. Counting is performed in a two's complement subtractive counter, thus, it is possible to count through negative zero and positive zero.

SKH b y

Skip High

6.4 us

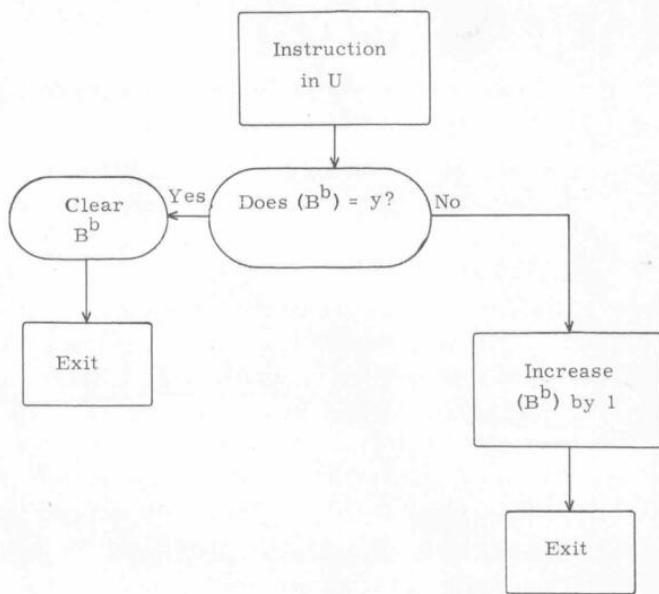
Compares the quantity in the designated index register with the operand, y . If the quantity in the index register is greater than or equal to the quantity y , an exit is performed. If the quantity in the index register is less than the quantity y , a skip exit is performed. The contents of the index register are unchanged by this instruction.

SKL b y

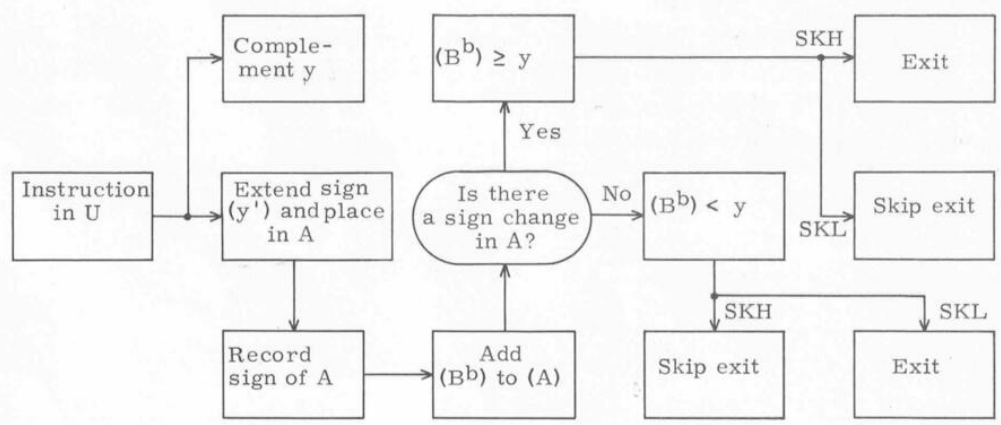
Skip Low

6.4 us

Compares the quantity in the designated index register with the operand, y . If the quantity in the index register is less than the quantity y , an exit is performed. If the quantity is greater than or equal to y , a skip exit is performed. The contents of the index register are unchanged by this instruction.



ISK



SKH and SKL

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 $b m$

Equality Search

6.9 + 5.2r*

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 location 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 skip 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), an exit is taken. Positive zero and minus zero are recognized as the same quantity.

THS $b m$

Threshold Search

6.9 + 6.0r*

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 location 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 skip 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, an exit is performed. In the comparison made here positive zero is considered as greater than minus zero.

* r = Number of additional words searched

MEQ b m Masked Equality 6.9 + 6.0r*
 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 in Q, operates in the same manner as an equality search.

MTH b m Masked Threshold 6.9 + 6.0r*
 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 in Q, this instruction operates in the same manner as the threshold search.

PTS b m Pattern Search 6.9 + 6.0r*
 Searches a list of operands to find one such that the inclusive or of (A) and (M) is equal to (A). 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 satisfies the criteria or until (B^b) equals zero. If the search is terminated by finding an operand which satisfies the condition, a skip exit is performed. The address of the operand satisfying the inclusive or condition is given by the sum of m and the final contents of B^b . If no operand in the list is found such that the inclusive or of (A) and (M) = (A), an exit is performed.

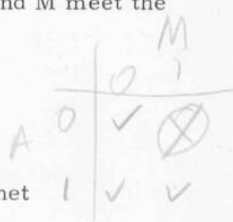
On a bit-by-bit basis, three combinations of the initial values of A and M meet the search criterion; one combination does not. The combinations are:

$A_i = 0$	$A_i = 1$	$A_i = 1$
$M_i = 0$	$M_i = 0$	$M_i = 1$

Search criterion met

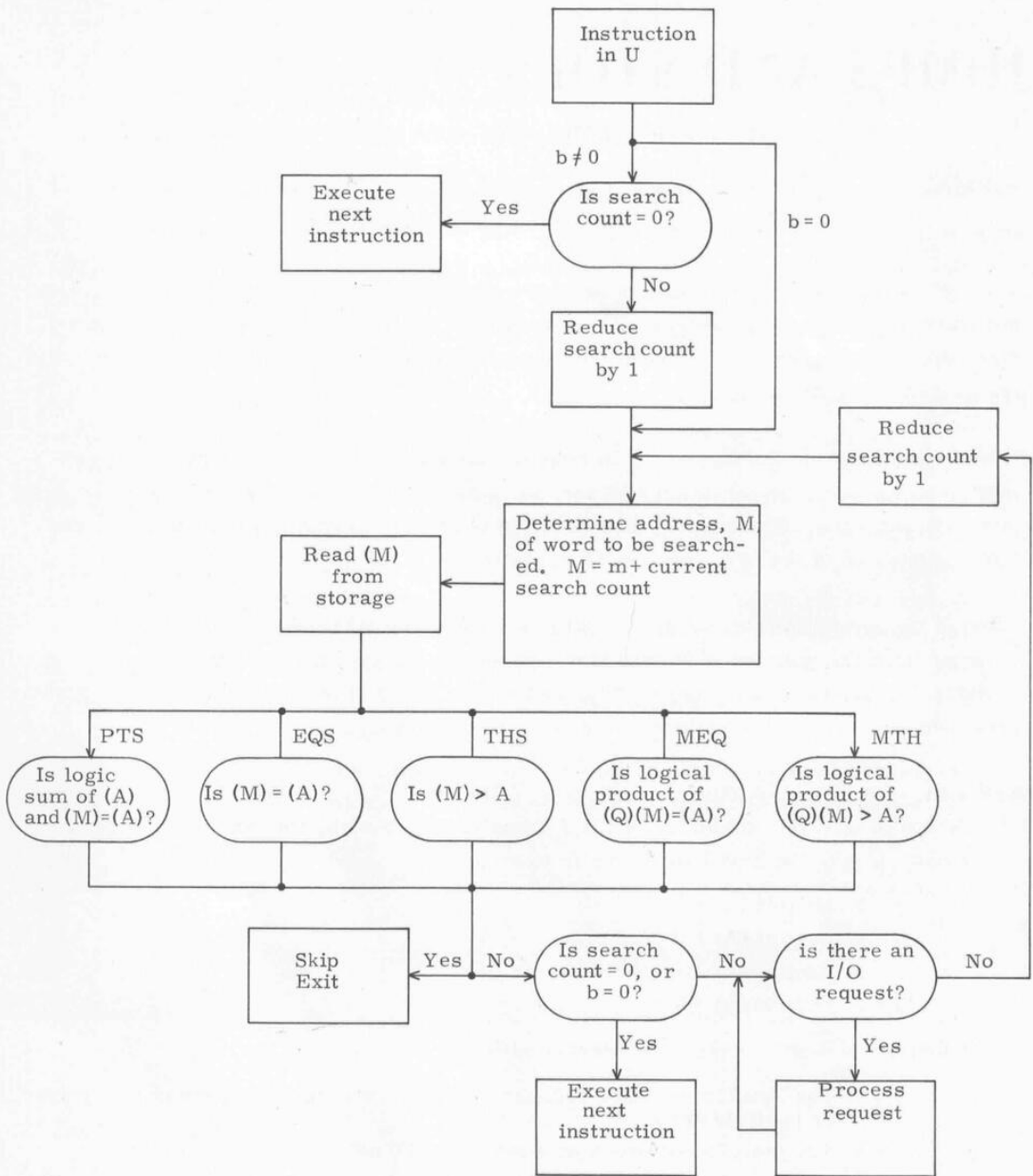
$A_i = 0$
$M_i = 1$

Search criterion not met



$A \pm M = A$
 $\Rightarrow A$ includes M
 $A \oplus M \sim M \oplus A$

* r = Number of additional words searched



Search

JUMPS AND STOPS

1) Address modification is used in the UJP instruction.

NORMAL

2) If indirect addressing is designated for the IJP or UJP instruction, at least one storage reference is required.

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 most 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 and the 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.

AJP j m

A Jump

7.7 us

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 when (A) = +0 or -0.

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".

QJP j m Q Jump 7.7 us

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.

SLJ j m Selective Jump 7.9 us

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. (Does not reference jump key setting.)
- 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.

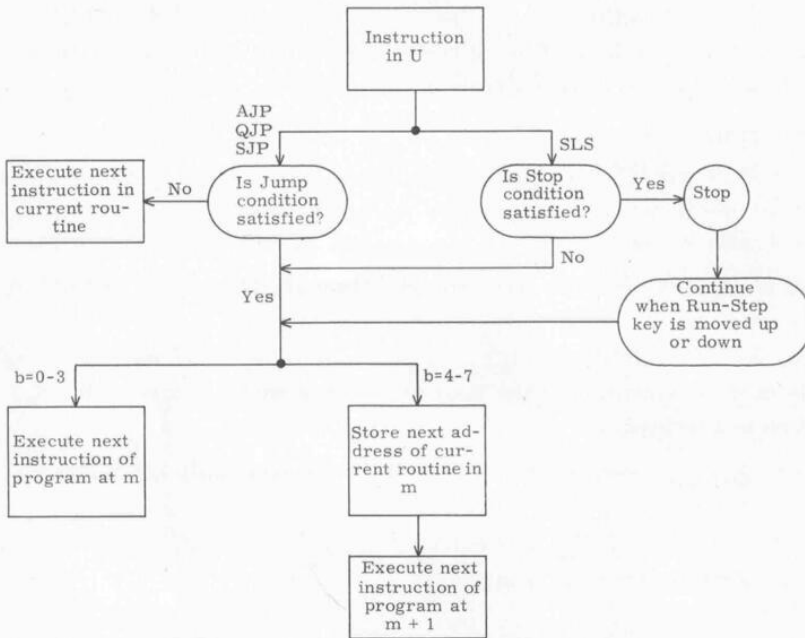
SLS j m Selective Stop 7.9 us

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. (Does not reference stop key setting.)
- 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.

UJP b m Unconditional Jump 4.8 us

Jumps unconditionally to M. $M = m + (B^b)$. If b = 0, the jump is to m. If b = 7, indirect addressing will be used.



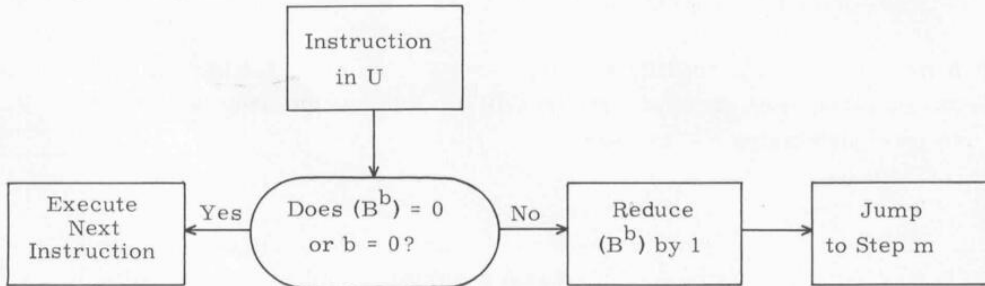
AJP, QJP, SJP, and SLS

IJP b m

Index Jump

7.0 us

Examines (B^b) . If this quantity is not zero, the quantity is reduced one count and a jump is executed to program step m . The counting operation is performed in a two's complement subtractive counter but negative zero is not generated because IJP terminates at positive zero. (See appendix.) If (B^b) is zero, the present program sequence is continued.

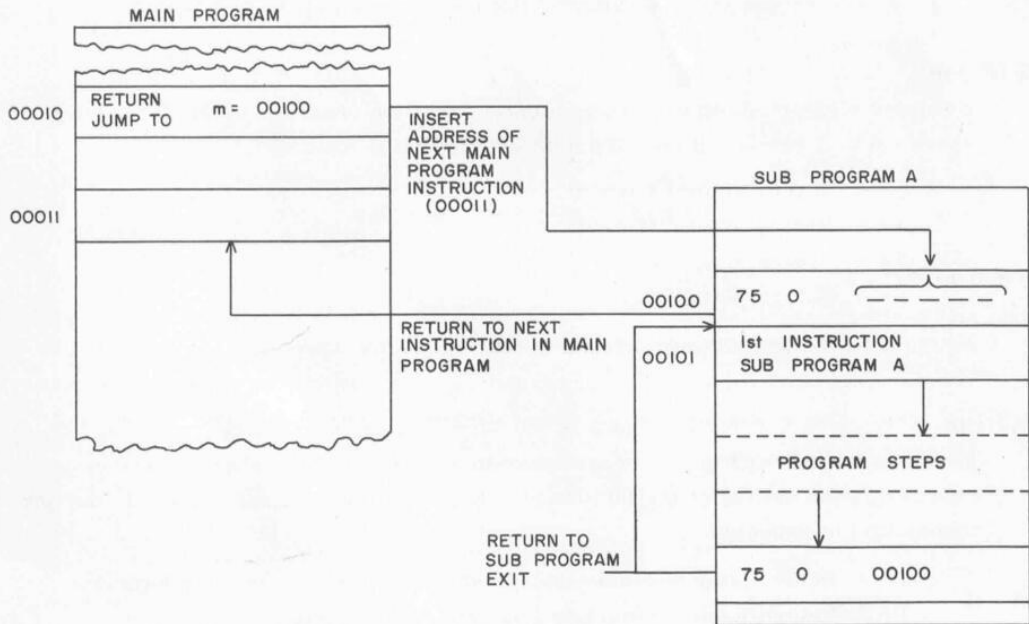


IJP

RETURN

- 1) Address modification applies only to URJ and XEC.
- 2) One storage reference is made.

A return jump begins a new program sequence at the address specified by $m + 1$, e.g., 00101. At the same time, the execution address of the first instruction in the new program sequence, 00100, is replaced with the address of the next program step in the main program (00011). The first instruction in a new program sequence is usually an unconditional jump which allows a return to the main program after completing the subprogram sequence.



Return Jump

AJP j m

A Jump

7.7 us

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) = -

When (A) is negative zero the interpretation is:

- 4 j = 0 The return jump is executed because, in this case, negative zero is recognized as positive zero.
- 5 j = 1 The return jump is not executed when (A) = + 0 or - 0.
- 6 j = 2 The return jump is not executed because the sign bit is a "1".
- 7 j = 3 The return jump is executed because the sign bit is a "1".

QJP j m

Q Jump

7.7 us

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) ≠ 0
- j = 6 Return jump if (Q) = +
- j = 7 Return jump if (Q) = -

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

SLJ j m

Selective Jump

7.9 us

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. (Does not reference jump keys.)
- 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.

SLS j m

Selective Stop

7.9 us

Stops on condition j and executes a return jump if the Run-Step key is moved in the run or step position. If the stop condition is not satisfied, the stop is not executed and the return jump is executed unconditionally.

- j = 4 Stop unconditionally. The return jump is executed when the Run-Step key is moved in either position.
- j = 5 Stop if stop key 1 is set. (Return jump.)
- j = 6 Stop if stop key 2 is set. (Return jump.)
- j = 7 Stop if stop key 3 is set. (Return jump.)

URJ b m

Unconditional Return Jump

4.8 us

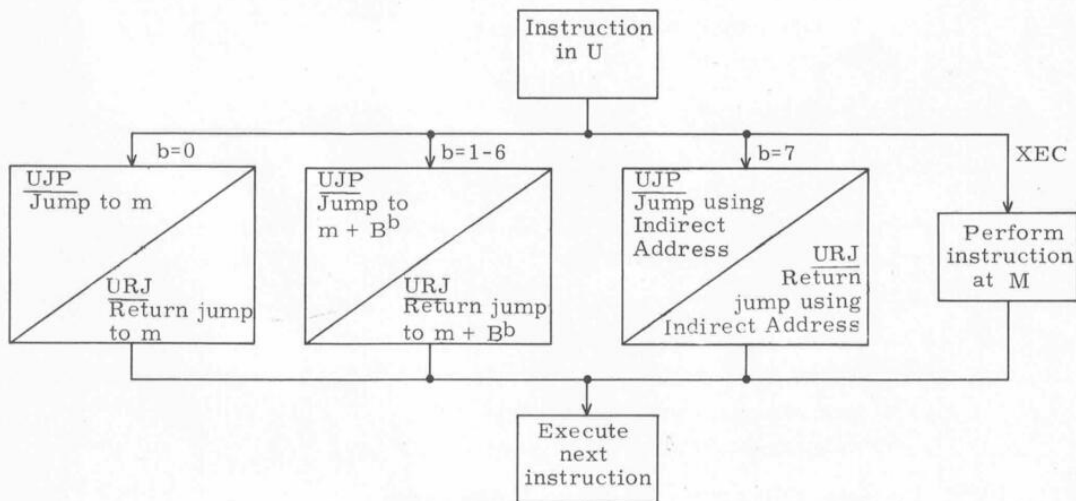
Return jumps unconditionally to M. $M = m + (B^b)$. If $b = 0$, the return jump is to m. If $b = 7$, indirect addressing will be used

XEC b m

Execute

8.2 + x us for inst. at M

Jumps to M. After executing M, the main sequence is continued unless M performs a jump. In this case, a new program is initiated at the location specified by the jump and a return is not effected to the main sequence. This instruction is effectively an indirect instruction, or a subroutine of a single instruction.

**UJP, URJ, and XEC**

Input-Output

METHODS OF DATA EXCHANGE

The computer communicates with external equipment via six buffer channels. These channels provide for the parallel transmission of binary words to and from computer storage.

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

Every external equipment is connected to one of these pairs. All six buffer channels may be concurrently transmitting information. However, only one external equipment can use any one buffer channel at any given instant. The rate of data flow on the buffer channels is usually dependent on the operating speed of the external equipment.

INITIATION AND CONTROL OF DATA EXCHANGE

The actual buffer (exchange) operation, although program initiated, proceeds under controls that are independent of the program. Information is buffered asynchronously with the execution of the program, thus, storage is permitted to serve its primary function of working with the arithmetic and control sections during the time a buffer is in progress.

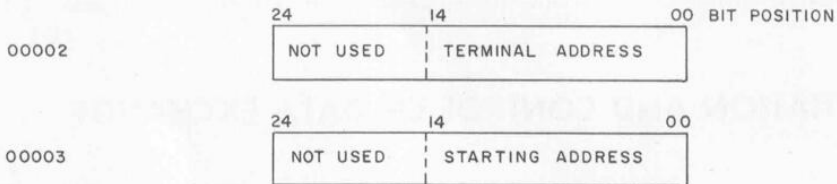
Buffer Control Word

Information is buffered in blocks one word at a time. The current and terminal storage addresses of the block are located in adjacent storage locations called buffer control words. Each of the six buffer channels has two assigned storage addresses which hold the buffer control word for the current and terminal addresses of the block.

Special Address

00000	Initial start
00002	Channel 1 control (terminal address)
00003	Channel 1 control (current address)
00004	Channel 2 control (terminal address)
00005	Channel 2 control (current address)
00006	Channel 3 control (terminal address)
00007	Channel 3 control (current address)
00010	Channel 4 control (terminal address)
00011	Channel 4 control (current address)
00012	Channel 5 control (terminal address)
00013	Channel 5 control (current address)
00014	Channel 6 control (terminal address)
00015	Channel 6 control (current address)

Buffer Control Words



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 even addressed control word. The starting address is automatically entered into the word when the buffer is initiated by an EXF instruction.

The execution address of the EXF instruction is used to designate the buffer starting address in storage. This address is automatically recorded in the lower 15 bits of the appropriate odd-number special storage location. The terminal address (plus one) of the block of data must have been previously recorded, by the program, in the lower 15-bit portion of the appropriate even-numbered special storage location prior to the execution of the external function instruction.

External Function (EXF) Instructions

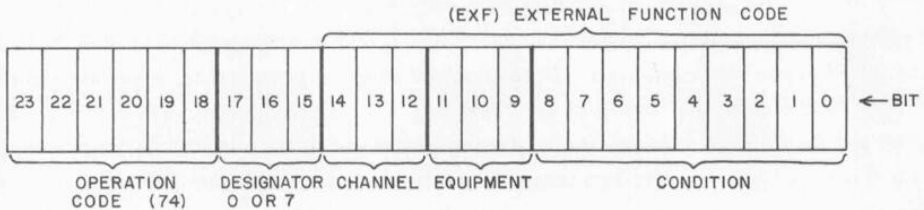
The EXF instructions initiate a buffer, sense for specified conditions, and select operations and equipment. EXF codes are listed at the end of the chapter.

There are three kinds of external instructions:

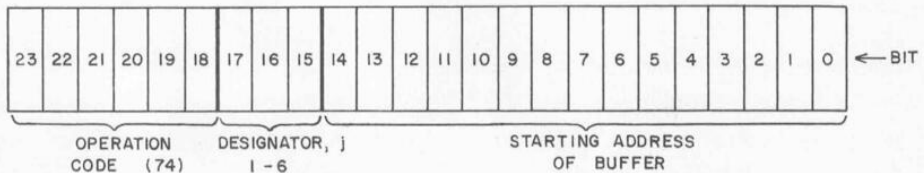
<u>Select</u>	74 0 XXXXX	
<u>Sense</u>	74 7 XXXXX	
<u>Activate</u>	74 j XXXXX	$1 \leq j \leq 6$

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.

Three basic modes of communication can be selected by an EXF instruction: 1604 (48-bit), 160 (12-bit), and 24-bit. Selecting 1604 or 160 mode permits communication with any peripheral equipment. (See p 3-13.)

No distinction is made between 24 and 48-bit transmissions on the control lines. However, when 24-bit mode is selected, a single storage word is read; two storage words are required for each 48-bit transmission.

When 160 mode is selected, the computer appears as a 160 to the peripheral equipments. The Status Request is accomplished by a one word output buffer which is a Select code because its sign bit is "1". If the sign bit is a "0", the output word is accepted as a word of information by the peripheral equipment. Only the lower 12 bits and the sign bit (bit 23) are significant during this mode. The lower 12 bits contain the Status Request code. The sign bit is a Function Ready signal to the peripheral equipment and indicates a select or Status Request operation. A one word input buffer then transmits the Status Reply to the computer. If an input or output operation is to follow, the computer program places the terminal address plus one in the appropriate control address. An Activate instruction follows and buffering proceeds in the normal manner except only the lower 12 bits of data are transmitted to or from computer storage.

The EXF 7 instructions sense the condition of an external equipment or the internal conditions (faults) of the computer. If the specified condition exists, a skip exit is performed; if not present, an exit is performed (Example 1).

Example 1	(00110)	74 7 00011
	(00111)	75 0 00007
	(00112)	53 1 00005

In this example, the translation of program step 00110 is skip on channel 1 inactive. If channel 1 is inactive the next instruction to be executed would be program step 00112, i. e., 53 1 00005. If channel 1 were active however, program step 00111 would be the next instruction executed. In either case, the sensing of a condition 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 buffer region

in storage. These instructions are the only instructions which can initiate a buffer (information flow) between the computer and an external equipment.

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

- 1) Sense for: (a) channel inactive and (b) equipment ready.
- 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 terminated (incremented starting address = terminating address) in order to inactivate the channel. This can be accomplished by setting the starting address equal to the terminating address and activating the channel (74 j instruction). This makes the channel inactive but no additional information is transmitted.

Buffer Scanner

Data exchange on each of the separate buffer channels is initiated by the program but proceeds under a control that is independent of the main program. This control is the buffer scanner.

The buffer scanner prevents one external equipment from dominating all others by sequentially monitoring the six buffer channels and the interrupt line to sense when any of these require action. When one of the positions demands action, the scanner stops. One word is processed and the scanner is released to resume its monitoring. If none of the positions demand action, a full scan is made in 1.6 usec. If, however, all seven positions demand processing at the same time, the maximum time lapse for processing successive words on the same channel will be 175 usec.

Figure 3-1 shows the relationship between executing instructions and recognizing action requests (input-output requests) from the buffer channels or interrupt line. During the time an input-output request is being processed, the scanner is released and resumes monitoring. If the scanner halts (indicating a channel demands action) this request will be processed before the main program is allowed to continue.

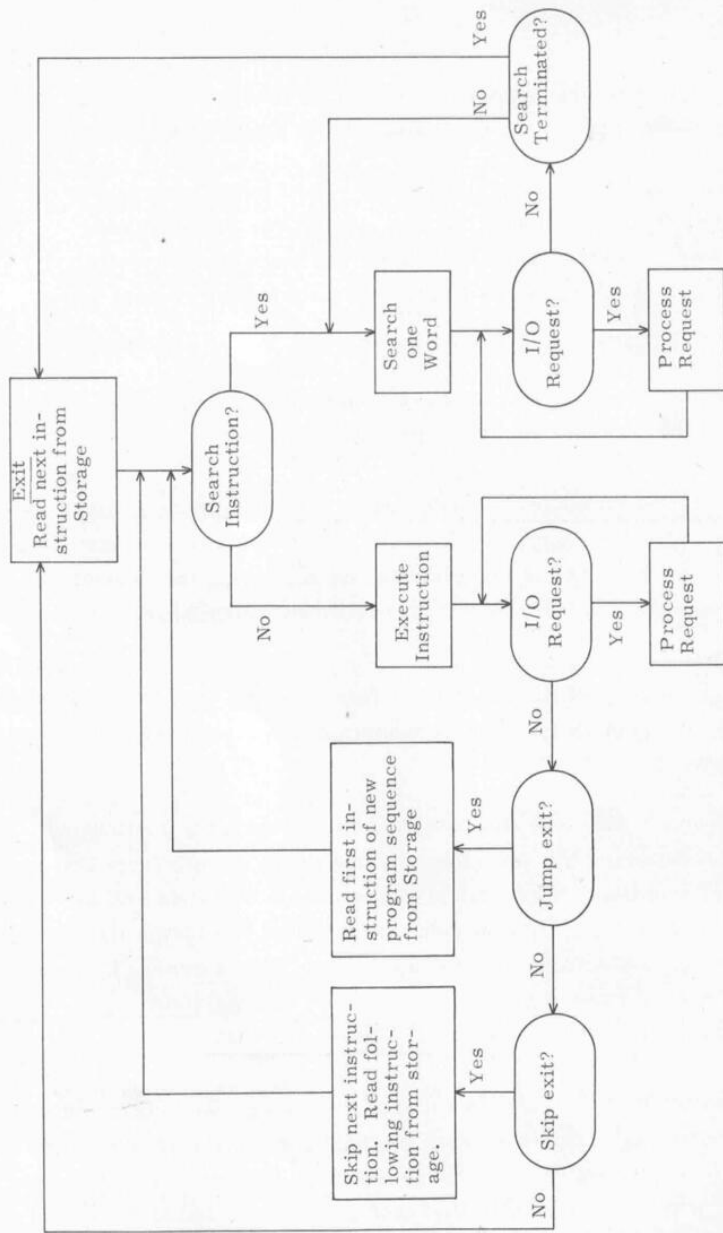


Figure 3-1. Instruction Searching

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 xxxxx) must have been previously programmed to permit an interrupt on that condition. Unless such selection is made, no interrupt is produced when the condition arises.

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 return jump to address 00016. This is a special address allocated for use as the entrance point to the interrupt routine and for the return from this routine to the main program.

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.

When an interrupt signal occurs, it remains on until turned off by a select (74.0) instruction from the computer. Sensing an interrupt does not remove the interrupt signal nor does it remove the interrupt selection. It is therefore mandatory that the routine turn off the interrupt signal so that upon exiting from the routine the signal will not immediately cause another interrupt.

In the Process Reader Not Ready (retxx) portion of the table, the interrupt signal is removed by a 74 0 10000 (clear all channel one selections). The interrupt selection is removed by a 74 0 11200 (No Interrupt on Reader Not Ready.) Note that, if desired, the interrupt selection need not be removed as evidenced by the ovfx section of the routine.

TABLE 3-1. TYPICAL INTERRUPT ROUTINE

00016	75 0 _____	Exit to Main Program
	└─┬─┘ next address of main program	
00017	75 0 int00	Entrance to Interrupt Routine
int00	74 7 00131	
int01	75 0 ovf00	Sense Overflow
int02	74 7 11201	
int03	75 0 ret00	Sense Reader End of Tape
int04	74 7 _____	
int05	75 0 _____	
ovf00	_____	
ovf01	_____	

	_____	Process Overflow

	74 0 00070	Clear Arithmetic Faults
ovf ___	75 0 00016	Exit to Main Program
ret00	_____	
ret01	_____	

	_____	Process Reader End of Tape
	74 0 10000	Remove Interrupt Signal
	74 0 11200	Remove Interrupt Selection
ret ___	75 0 00016	Exit to Main Program

Real Time Clock

Incorporated into the interrupt circuitry of the 924 computer is a real time clock source which generates a pulse once every 100 milliseconds. This pulse will produce an interrupt only if a 74 0 01000 (select real time interrupt) instruction is programmed.

CONSOLE INPUT-OUTPUT EQUIPMENT

Two input-output devices mounted on the console are standard equipment with the 924 computer. A Teletype high speed punch and a high speed tape reader provide for the processing of perforated paper tape. The console input-output units communicate with the central computer via buffer channel 1 (input) and 2 (output). Other input-output units may share these channels but console input-output units cannot use any of the other channels.

Data may be transmitted between the console equipments and the computer in either the character or the assembly mode. In the character mode a 7-bit character is buffered one character at a time. The 7-bit character occupies the lowest bit positions of a 24-bit word, the upper bits are "0's".

In the assembly mode the 24-bit word consisting of four 6-bit characters is buffered. During an input buffer in the assembly mode four successive characters assembled into a 24-bit word are 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 24-bit word from the computer is dis-assembled into four characters beginning with character 3, which is the upper 6 bits of the word.

*Auto-
mate?*

PAPER TAPE READER

The 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 milliseconds. Selecting the Reader, 74 0 112xx, automatically selects 24-bit mode. Any other mode selection for this channel after the 74 0 112xx Select, but prior to the Activate (74 1 xxxxx) is illegal. ~~A 160 mode does not affect operation.~~ *or 1604 mode select clears the reader.*

Manual controls for the reader are on the 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 seven 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 four characters in a word is indicated by a hole in the control level.

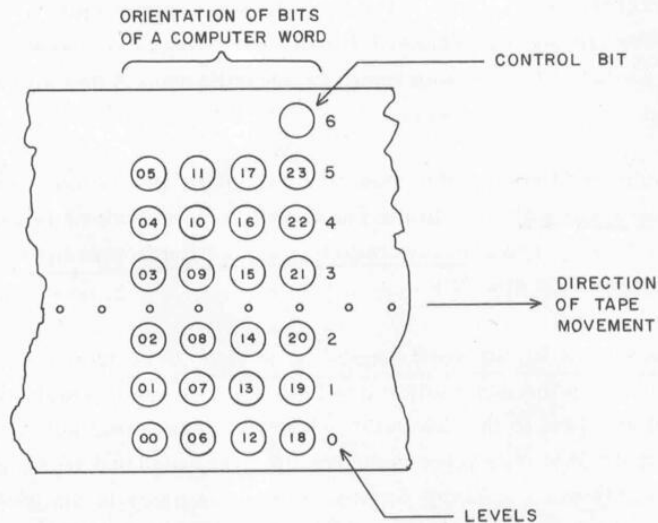


Figure 3-2. Seven Level Punched Paper Tape

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) If a 7th level punch in the assembly mode is absent 4 characters from the last 7th level punch indicating the end of information or faulty tape.

The end-of-tape indicator is set on any of three conditions:

- 1) Absence of a 7th level every 4th character in the assembly mode.
- 2) On a computer Master Clear.
- 3) By a 74 0 1121X instruction. This instruction is used to indicate the end of information in the character mode.

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, a 74 0 1121X instruction must be programmed to set the end-of-tape indicator. 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.

Tape motion stops when buffer is loaded? Yes, see previous page.

PAPER TAPE PUNCH

The punch which prepares paper tape output is always connected to buffer channel 2. Nominal operating rate is 110 characters per second. In character mode, 7-bit characters are punched; in assembly mode, 6-bit characters are punched, with a 7th level every four characters.

On the punch, the feedout lever provides for punching out leader. A micro-switch 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 by a 74 0 21200 instruction.

INTERNAL EXF SELECT INSTRUCTIONS

- 74 0 000C0 Interrupt on Channel C Inactive
 Selects interrupt when channel C is inactive. C = 1 - 6
- 000C1 Remove Selection Above
 Interrupt on channel C inactive selection removed
- 00070 Clear Arithmetic Faults
 Removes arithmetic fault indication and turns off arithmetic fault
 background lights on console
- 00100 Interrupt on Arithmetic Faults
 Selects interrupt on occurrence of any arithmetic faults
- 00101 Remove Selection Above
 Interrupt on arithmetic faults selection removed
- 01000 Select Real-Time Interrupt
 Produces an interrupt pulse every 100 milliseconds
- 01001 Remove Selection Above
 Stops process of producing interrupt pulses by removing real time
 interrupt selection

BUFFER MODE INSTRUCTIONS

- 74 0 04010 - Select 24-bit mode for channel 1 and 2
- 04011 - Select 160 (12-bit) mode for channel 1 and 2
- 04012 - Select 1604 (48-bit) mode for channel 1 and 2
- 04020 - Select 24-bit mode for channel 3 and 4
- 04021 - Select 160 (12-bit) mode for channel 3 and 4
- 04022 - Select 1604 (48-bit) mode for channel 3 and 4
- 04030 - Select 24-bit mode for channel 1 and 2, 3 and 4
- 04031 - Select 160 (12-bit) mode for channel 1 and 2, 3 and 4
- 04032 - Select 1604 (48-bit) mode for channel 1 and 2, 3 and 4
- 04040 - Select 24-bit mode for channel 5 and 6
- 04041 - Select 160 (12-bit) mode for channel 5 and 6
- 04042 - Select 1604 (48-bit) mode for channel 5 and 6
- 04050 - Select 24-bit mode for channel 1 and 2, 5 and 6
- 04051 - Select 160 (12-bit) mode for channel 1 and 2, 5 and 6
- 04052 - Select 1604 (48-bit) mode for channel 1 and 2, 5 and 6
- 04060 - Select 24-bit mode for channel 3 and 4, 5 and 6
- 04061 - Select 160 (12-bit) mode for channel 3 and 4, 5 and 6
- 04062 - Select 1604 (48-bit) mode for channel 3 and 4, 5 and 6
- 04070 - Select 24-bit mode for channel 1 and 2, 3 and 4, 5 and 6
- 04071 - Select 160 bit mode for channel 1 and 2, 3 and 4, 5 and 6
- 04072 - Select 1604 bit mode for channel 1 and 2, 3 and 4, 5 and 6
- ~~04012 - Clear channel 1 and 2 160 mode select~~
- ~~04011 - Clear channel 1 and 2 1604 mode select~~
- ~~04022 - Clear channel 3 and 4 160 mode select~~
- ~~04021 - Clear channel 3 and 4 1604 mode select~~
- ~~04042 - Clear channel 5 and 6 160 mode select~~
- ~~04041 - Clear channel 5 and 6 1604 mode select~~

CHANNEL CLEAR INSTRUCTIONS

- 74 0 C0000 Clear all channel C selections
 Clears all previous equipment selections made on channel C.

*1604 or 924 - mode only. Must be used before
going to 160-mode*

INTERNAL EXF SENSE INSTRUCTIONS

- 74 7 000C0 Skip on Channel C Active
 Skip Exit if channel C is active
- 000C1 Skip on Channel C Inactive
 Skip Exit if channel C is inactive

ARITHMETIC FAULT INSTRUCTIONS

Refer to appendix.

- 74 7 00110 Skip on Divide Fault
- 00111 Skip on No Divide Fault
- 00120 Skip on Shift Fault
- 00121 Skip on No Shift Fault
- 00130 Skip on Overflow Fault
- 00131 Skip on No Overflow Fault

CONSOLE EXF SELECT INSTRUCTIONS

PAPER TAPE READER

- 74 0 11200 Reader and No Interrupt on End-of-Tape
 Selects reader.* Reader will not move until buffer is activated.
 The interrupt on End-of-Tape selection cleared.

*End-of-tape indicator is cleared when reader mode of operation is manually selected;
Reader Ready light turns on.

11210 Set End-of-Tape Indicator
Sets end-of-tape indicator*
Clears the Interrupt on End-of-Tape Selection
Primarily used in controlling the reader in character mode

11220 Reader and Interrupt on End-of-Tape
Selects reader
Selects interrupt on end-of-tape **
An external master clear or a 74 0 11200 (P. T. Reader and no
interrupt on end-of-tape) clears interrupt on end-of-tape

PAPER TAPE PUNCH

74 0 21200 Punch Assembly Mode
Selects punch
Turns on punch motor
Selects assembly mode

21210 Punch Character Mode
Selects punch
Turns on punch motor
Selects character mode

21240 Turn Punch Motor Off
Turns punch motor off

* Reader Ready light turns off and paper tape stops moving when end-of-tape indicator is set.

** The lack of a Mod 4 and a 7th level in assembly mode or a 74 0 11210 (set end-of-tape indicator) will set end-of-tape indicator.

CONSOLE EXF SENSE INSTRUCTIONS

PAPER TAPE READER

- 74 7 11200 Skip on Reader, End-of-Tape
 Skip exit if end-of-tape indicator is set
- 11201 Skip on Reader, No End-of-Tape
 Skip exit if end-of-tape indicator is not set
- 11210 Skip on Reader, Assembly Mode
 Skip exit if the assembly mode of operation selected
- 11211 Skip on Reader, Character Mode
 Skip exit if character mode of operation selected

PAPER TAPE PUNCH

- 21200 Skip on Punch Out-of-Tape
 Skip exit if punch out-of-tape condition exists
- 21201 Skip on Punch, No Out-of-Tape
 Skip exit if punch out-of-tape condition does not exist

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.

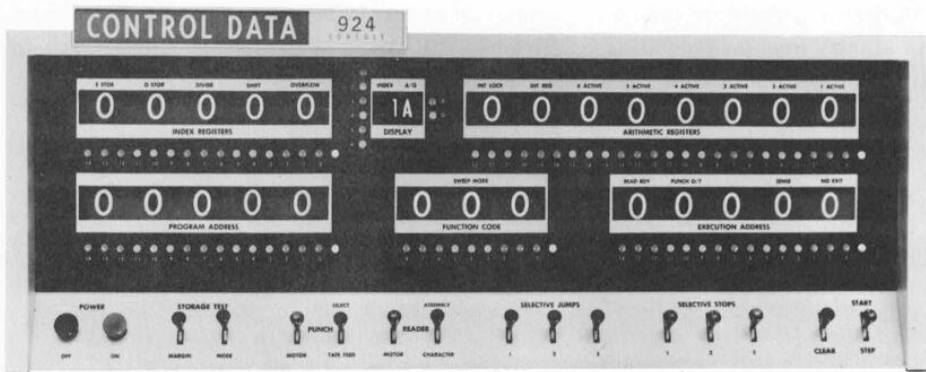


Figure 4-1. Center Panel of Console

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. Different shades of blue are used in adjacent octal groups; within an octal group the three buttons are of the same shade.

At the right end of each register is a CLEAR push button (white). This button will clear the individual 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.

STOP

- 1) Illegal function codes 00, 77, and 52.3-52.6
- 2) Selective Stops (instruction 76)
- 3) Pressing START-STEP switch
- 4) 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, A/Q-4 is fourth from the left in the A/Q register indicator panel. 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.

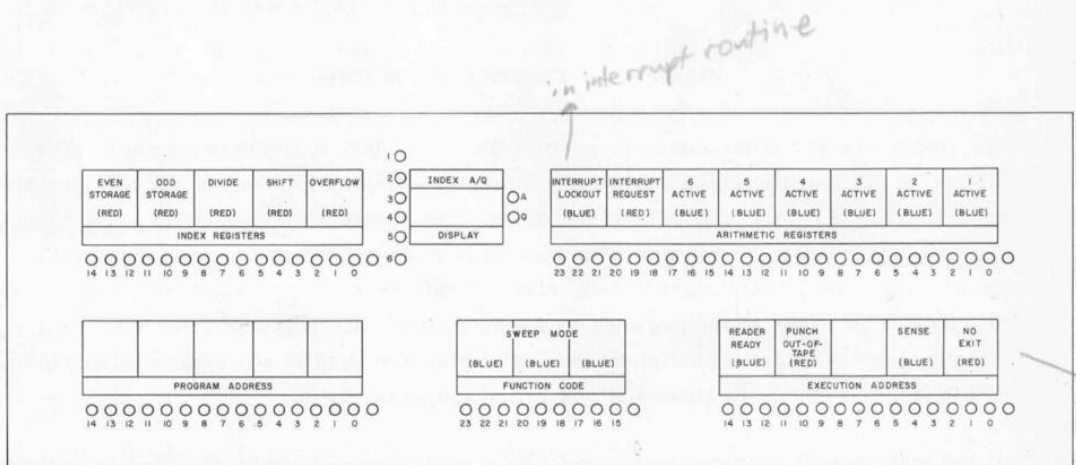


Figure 4-2. Console Display

TABLE 4-1. CONDITIONS INDICATED BY CONSOLE BACKGROUND LIGHTS

Light	Condition
A/Q-1 (blue)	<u>Interrupt Lockout</u> - Computer is in interrupt routine.
A/Q-2 (red)	<u>Interrupt Request</u> - Interrupt request signal is being received from interrupt circuit. <i>Does Arithmetic fault turn this on? or only if int desired?</i>
A/Q-3 (blue)	<u>Channel 6 Active</u> - Channel 6 is in use for output buffer.
A/Q-4 (blue)	<u>Channel 5 Active</u> - Channel 5 is in use for input buffer.
A/Q-5 (blue)	<u>Channel 4 Active</u> - Channel 4 is in use for output buffer.
A/Q-6 (blue)	<u>Channel 3 Active</u> - Channel 3 is in use for input buffer.
A/Q-7 (blue)	<u>Channel 2 Active</u> - Channel 2 is in use for output buffer.
A/Q-8 (blue)	<u>Channel 1 Active</u> - Channel 1 is in use for input buffer.
A/Q Select "A" "Q" Index Register Select "1. 2. 3. etc. "	A Register Selected Q Register Selected Index Register Selected Indicates selected register whose contents are being displayed in the appropriate indicator panel. Quantities can now be manually inserted into selected registers.
EA-1 (blue)	<u>Reader Ready</u> - (1) Paper tape is at load point, ready for an input buffer; or (2) paper tape input buffer is in progress.]
EA-2 (red)	<u>Punch Out of Tape</u> - Punch tape reel is nearly empty.
IR-2 (red)	<u>Odd Storage Fault</u> - Fault in sequence chain of odd storage unit is inoperative until master cleared.
IR-1 (red)	<u>Even Storage Fault</u> - Fault in sequence chain of even storage unit; storage unit is inoperative until master cleared. <i>call maintenance</i>
IR-3 (red)	<u>Divide Fault</u> - Improper divide instruction executed.
IR-4 (red)	<u>Shift Fault</u> - Shift instruction executed with shift count greater than 63 (decimal).
IR-5 (red)	<u>Overflow Fault</u> - Required sum or difference exceeds capacity of A register
EA-4 (blue)	<u>Sense</u> - Computer is sensing for various internal or external conditions.
EA-5 (red)	<u>No Exit</u> - <u>Restart operation</u> . If unable to proceed, master clear and restart program. If condition persists, notify maintenance.
FUNCTION CODE (blue) (3 lights)	Computer in sweep mode of operation.

TABLE 4-2. MAIN COMPUTER CONTROLS

Control		Function
POWER push button	ON - blue	Applies d-c and a-c power to computer by energizing contactor in primary power lines of motor-generator.
	OFF - red	Removes d-c and a-c power from computer by de-energizing contactor in primary power lines of motor-generator.
STORAGE TEST Lever switches lock in up, down and neutral positions	MARGIN	Varies the bias applied to storage sense amplifiers. Used for maintenance purposes only; should be in neutral position at all other times.
	MODE <i>SWEEP</i>	UP: an instruction is executed repeatedly in either the Step or Start mode. DOWN: contents of consecutive storage locations may be manually examined by depressing Step. Consecutive words are displayed in function code and execution address but are not executed.
SELECTIVE JUMPS 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 (Same function in all positions)
SELECTIVE STOPS 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 (Same function in all positions)
CLEAR Lever switch, momentary in up and down positions.		*DOWN master clears the computer, clears all operational registers and most control FFs. <i>5bps comp.</i> UP master clears external equipment, causing most of the registers and control FFs of the external equipment to be cleared. Also clears input and output channels. <u>Selects paper tape reader.</u>

TABLE 4-2. (CONT'D.)

Control	Function
<p>START-STEP</p> <p>Lever switch, <u>momentary</u> in up and down positions.</p>	<p>START (up) selects high-speed mode in which a program of instructions and auxiliary operations proceeds until completed or stopped.</p> <p>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). <u>Step selection overrides any previous selection of Start.</u> <i>i.e., used to stop comp.</i></p>
<p>SELECT A SELECT Q SELECT INDEX REGISTER</p>	<p>Selection of the designated register. Register contents are displayed on console and quantities may be inserted manually into the registers.</p>
<p>*SET Push Buttons</p> <p>Numbered in the powers of 2, beginning with zero. Each group of 3 is an octal digit.</p>	<p>Allow for manual entry of a quantity into a given register. Forces that particular stage of the register to the Set state. The register to be loaded (A, Q, or B^b) must be selected.</p>
<p>*CLEAR Push Button</p>	<p>Clears the individual FFs within the selected register.</p>

* Should be used only when the computer is stopped

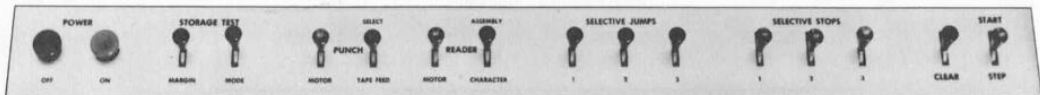


Figure 4-3. Manual Controls

READER AND PUNCH CONTROLS

TABLE 4-3. READER AND PUNCH CONTROLS

Switch	Function
PUNCH MOTOR Up: On Down: Off	Turns punch motor on or off. (Motor may also be turned on under program control.)
SELECT Up: Locks	Enables use of punch.
TAPE FEED Down: Momentary	Causes leader to be punched out.
READER MOTOR Momentary: Up: On Down: Off	Turns reader motor on or off. (Motor cannot be turned on by any other means.)
CHARACTER Momentary	Selects character mode. The reader sends each 7-bit character to computer separately.
ASSEMBLY Momentary	Selects Assembly mode. Reader sends four consecutive 6-bit characters assembled into a word to computer.

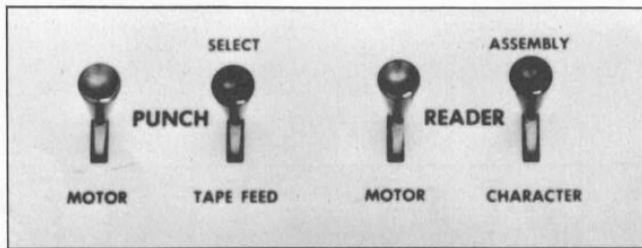


Figure 4-4. Reader and Punch Controls

1607 CONTROLS AND INDICATORS

Each tape unit is provided with push buttons for manual operation. These controls are mounted on a panel above the front door.

TABLE 4-4. 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	Indicates tape rewound under program control and interlocked at metal leader; operation prevented until Stop Manual switch operated and moved off metal leader.
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

OPERATION

The main computer and external equipment are placed in operation by procedures which include loading and unloading data and programs, making necessary manual selections, and starting a program. Steps are listed in the recommended order of performance, beginning with the computer and external equipment in a shut down condition.

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).
- 2) If punch is to be used, set Punch switch to Select and check for sufficient paper in reel. (If supply of tape is low see paragraph on additional procedures.)
- 3) Make required manual selections:
 - Selective Jumps
 - Selective Stops
- 4) Prepare paper tape baskets and empty chad box.
- 5) If paper tape is to be read, load into reader.
 - a. Turn tape release lever clockwise to raise tape guide plate.
 - b. Insert tape as shown in figure 4-5.
 - c. Turn on reader (Reader switch).
 - d. If tape is bi-octal with 7th level control holes, select Assembly mode.
 - e. If tape is in flex or other code, select Character mode.
- 6) If magnetic tape is to be used: (figure 4-6)
 - a. Open door to handler.
 - b. Check that file reel to be loaded has been file protected as necessary.
 - c. 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.

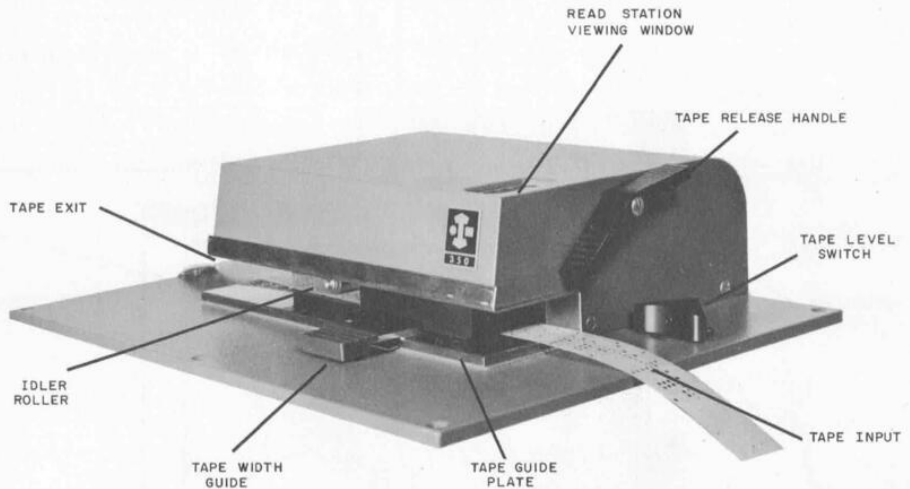


Figure 4-5. Paper Tape Reader

- d. 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.
- e. Connect file tab to permanent machine leader.
- f. Take up slack by turning file reel while pressing upper Reel Brake pushbutton.
- g. Lift leader clamp and close door.
- h. 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 2 minutes. The Stop Manual lamp should turn on, if not, call maintenance.
- i. Press Forward button; wait 10 seconds.
- j. Press Stop Manual.
- k. Press Rewind button. Unit is ready when Rewind lamp turns off. If Stop Manual lamp remains on unit is not ready; call maintenance.
- 7) Master Clear both internal and external (press Clear then raise it).
- 8) Set Program Address register to address of first instruction of program.
- 9) Begin computer operation (set Start switch).

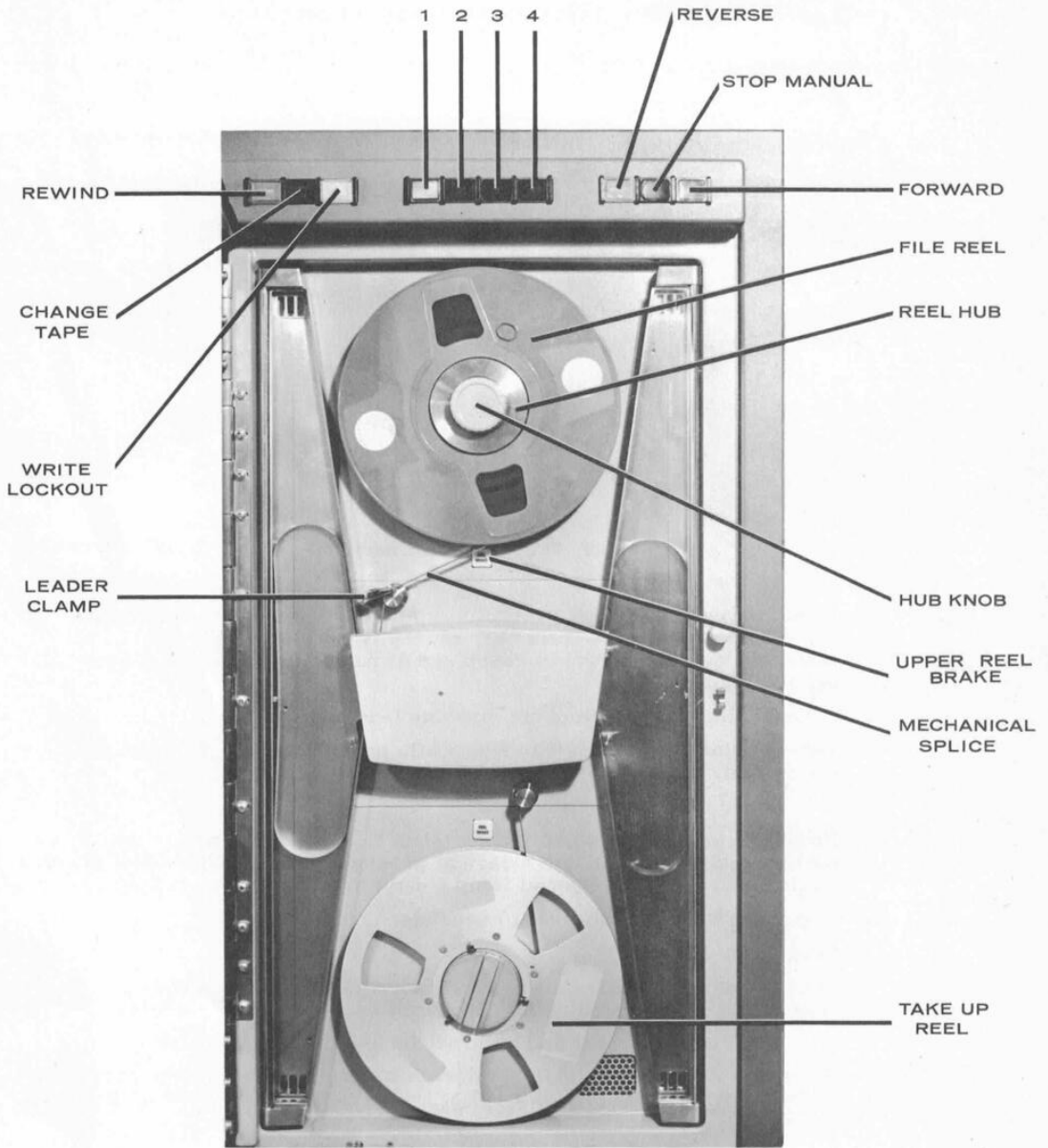


Figure 4-6. Tape Unit

STARTING OPERATION WITHOUT PRE-STORED LOAD PROGRAM

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 00002.
- 6) Set terminal address of buffer ⁺ in lowest five octal digits of A register
- 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) Steps 2 through 9 of operation with stored program may be performed.

SHUTTING DOWN EQUIPMENT

After operation has stopped, shut down the equipment.

- 1) Remove paper tape from reader and baskets; rewind tapes.
- 2) Turn off reader motor.
- 3) If punch was used, generate a foot of leader by pressing Tape Feed; remove tape and wind it up.
- 4) To unload magnetic tape:
 - a) Press Stop Manual button to select manual mode.
 - b) Press Reverse button to move tape backwards to change tape position.
 - c) Open front door of tape unit.
 - d) To secure tape, lower leader clamp.
 - e) Press the upper Reel Brake button to release the mechanical brake and pull tape from file reel to provide slack.
 - f) Unfasten mechanical splice which connects the file tab to the permanent machine leader.

- g) Loosen file reel hub knob and remove the file reel.
- h) Check if reel needs to be file protected and also if it is labelled adequately prior to storage.
- 5) Press Power Off button, which disconnects power from all equipments.

ADDITIONAL PROCEDURES

REPLACING TAPE ROLL AT PUNCH

The paper tape punch (figure 4-7) is mounted on a sliding rack on the left side of the console. To replace a roll of tape:

- 1) Pull out punch drawer.
- 2) Remove the tape reel from cradle at side of punch.
- 3) Unscrew tape hold-down assembly, remove old roll, and place new roll on reel. Replace hold-down assembly and mount reel in cradle.
- 4) Thread tape as shown in figure 4-7. Bring tape around lower roller and into guides leading to punch block.
- 5) Turn on punch motor and advance tape through the punch block by pressing the tape feed-out lever (at top).
- 6) Bring leader out through slot in compartment.
- 7) Slide punch back into compartment.

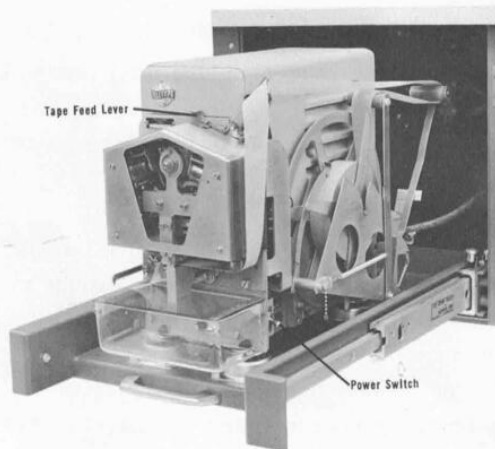


Figure 4-7. Paper Tape Punch

FILE PROTECTION RING

The back of the 1607 file reel has a slot near the hub which accepts a plastic file protection ring (figure 4-8). Writing on a tape is possible only when the reel contains a file protection ring. When the ring is in place the Write Lockout indicator goes out immediately after the reel is loaded onto the tape unit. 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.

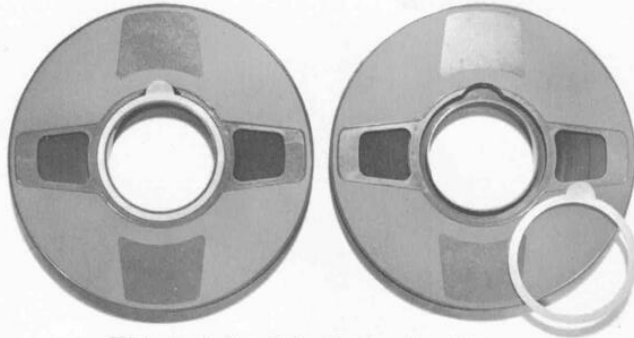


Figure 4-8. File Protection Ring

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 PROCEDURES

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.
No Exit	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 and Overflow Faults. (Refer to appendix.)

GLOSSARY

ABSOLUTE ADDRESS	A specific storage location; contrast with relative address.
ACCESS TIME	The time from request to delivery of data from storage (1.8 usec.).
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 satisfied (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 24-bit subtractive accumulator (modulus: $2^{24}-1$).
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.

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, four 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) 17 bits.
CLEAR	A command that removes a quantity from a register by placing every stage of the register in the "0" state.

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.
COMPILER	A routine which automatically produces a specific program for a particular problem. The routine determines the meaning of 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.
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.
CYCLE TIME	The time required for a complete storage reference (6.4 usec.).
EVEN STORAGE	The storage unit which contains the 4096 even addresses.
EXCLUSIVE OR	A logical function in Boolean algebra that is satisfied when any but not all of its terms are "1". It is not satisfied when all terms are "0".
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	Execute next instruction.

EXTERNAL FUNCTION	<ol style="list-style-type: none"> 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 pre-determined position.
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.
FUNCTION CODE	The upper 9 bits of a 24-bit instruction consisting of the operation and index codes.
INCLUSIVE OR	A logical function in Boolean algebra that is satisfied when any of its terms are "1". It is <u>not</u> satisfied when all terms are "0".
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 REQUEST	A signal received from an external equipment 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.
LOOP	Repetition of a group of instructions in a routine.
MASK	In the formation of the logical products of two quantities, one quantity may mask the other, i. e., determine what part of the other quantity is to be considered. If the mask is "0" that part of the other quantity is cleared; if the mask is a "1", the other quantity is left unaltered.
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.
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 4096 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).
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^2 , O^4 , O^6 REGISTERS	Output registers used for output buffer 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 <u>not</u> 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	A 24-bit instruction contained in a 24-bit storage address; such an instruction is read from storage and executed.
Q REGISTER	Auxiliary arithmetic register which assists the A register in the more complicated arithmetic operations (Modulus: $2^{24}-1$).
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.
READ	To remove a quantity from a storage location.
READY	<ol style="list-style-type: none"> 1) The 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 (24 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	That portion of the computer which automatically samples the state of the buffer channels and interrupt line, and initiates action in accordance with the information obtained.
SECONDARY REGISTERS	Transient registers not displayed on the console (U ² , S ^{1,2} , Z ^{1,2} , X, O ^{2,4,6}).
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.

SKIP EXIT	Execute instruction at (P) + 2.
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.
TOGGLE	To complement each bit of a quantity as a result of an individual condition.
TRANSLATION	An indication of the content of a group of bit registers. A complete translation gives the exact content; a partial translation indicates only that the content is within certain limits.
TRANSMISSION, FORCED	A transfer of bits into a register which has not been cleared previously.
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 REGISTER	Program Control register. A 24-bit register that holds a program step while the instruction contained in it is being executed. The lower 15 bits make up a subtractive accumulator (Modulus 2^{15}), used primarily as a counter in shift and search instructions.
WORD	A unit of information which has been coded for use in the computer as a series of bits. The normal word length is 24 bits.
WRITE	To enter a quantity into a storage location.

X REGISTER

Exchange register. Most internal transmissions between the arithmetic section and the rest of the computer are made through X. Output transmissions go through X.

Z¹ REGISTER

Storage Restoration register (even storage). Holds the word to be written into a given storage location.

Z² REGISTER

Storage Restoration register (odd storage). Holds the word to be written into a given storage location.

APPENDIX SECTION

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. The FFs set by fault conditions may be cleared by a computer Master Clear.

Shift Fault

Any attempt to shift a register more than 63_{10} (77_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 74 7 00120, 1.

Divide Fault

A divide fault occurs in fixed point divide instructions (25) when the divisor is zero or the required quotient exceeds the 23-bit capacity of the quotient register, A. The sign bit of A 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 is produced when the capacity of the A register ($>\pm 2^{23}-1$) is exceeded. The fault is sensed after the arithmetic operation causing the overflow is completed. An overflow may be sensed by a 74 7 00130, 1.

If an interrupt has been selected, the main program may or may not be halted before the next instruction can be executed because the scanner may not immediately recognize the interrupt.

A pass or sense overflow instruction could be programmed after the arithmetic instruction which causes the overflow. The interrupt would, in this case, be recognized before executing succeeding instructions which might alter the contents of the A register.

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 VI EXF AND CHARACTER CODES

SELECT

10x60x10
6000

74 0 000C0	Interrupt on Channel C inactive; C=1-6
000C1	Remove Selection above on Channel C
00070	Clear Arithmetic faults
00100	Interrupt on Arithmetic faults
00101	Remove Selection above
01000	Select interrupt from Start real-time clock
01001	Remove selection of Stop real-time clock

02000 ? already to M1180m.

SENSE

74 7 000C0	Skip exit on Channel C active; C=1-6
000C1	Skip exit on Channel C inactive; C=1-6
00110	Skip exit on Divide fault
00111	Skip exit on No Divide fault
00120	Skip exit on Shift fault
00121	Skip exit on No Shift fault
00130	Skip exit on Overflow fault
00131	Skip exit on No Overflow fault

00200 skip exit on RT Clock
00201 "No" "

CONSOLE I/O EQUIPMENT

SELECT

74 0 11200	Reader and no interrupt on end-of-tape
11210	Reader and set end-of-tape indicator
11220	Reader and interrupt on end-of-tape
21200	Punch Assembly mode
21210	Punch Character mode
21240	Turn Punch Motor off

SENSE

74 7 11200	Skip exit on Reader, end-of-tape
11201	Skip exit on Reader, no end-of-tape
11210	Skip exit on Reader, assembly mode
11211	Skip exit on Reader, character mode
21200	Skip exit on Punch out-of-tape
21201	Skip exit on Punch no out-of-tape

1607 CODES

SELECT

74 0	320n1	Select read tape n, binary
	320n2	Select read tape n, coded
	32001	Read selected read tape, binary
	32002	Read selected read tape, coded
	32004	Interrupt when selected read tape ready
	32005	Rewind selected read tape
	32006	Backspace selected read tape
	32007	Rewind selected read tape, interlock
	420n1	Select write tape n, binary
	420n2	Select write tape n, coded
	42001	Write selected write tape, binary
	42002	Write selected write tape, coded
	42003	Write end-of-file mark
	42004	Interrupt when selected write tape ready
	42005	Rewind selected write tape
	42006	Backspace selected write tape
	42007	Rewind selected write tape, interlock
	---	Status Request

SENSE

74 7	32000-1	Ready to read
	32002-3	Read parity error
	32004-5	Read length error
	32006-7	End-of-file mark
	42000-1	Ready to write
	42002-3	Write reply parity error
	42004-5	Write reply length error
	42006-7	End of tape marker

1608 CODES

SELECT

74 05 77n1	Select tape N to read binary
77n2	Select tape N to read coded
7001	Prepare selected tape to read binary
7002	Prepare selected tape to read coded
7004	Interrupt when selected tape ready
7005	Rewind selected tape
7006	Backspace selected tape
7007	Rewind and unload selected tape
7101	Turn off "Tape Indicator" on read unit
7102	Set low density on read unit
7103	Set high density on read unit
7104	Search file mark forward on read unit
7105	Search file mark backward on read unit
7106	Remove interrupt selection on read unit
74 06 77n1	Select tape N to write binary
77n2	Select tape N to write coded
7001	Prepare selected tape to write binary
7002	Prepare selected tape to write coded
7003	Write end-of-file (tape mark) on selected unit
7004	Interrupt when selected tape ready
7005	Rewind selected tape
7006	Backspace selected tape
7007	Rewind and unload selected tape
7101	Turn off "Tape Indicator" on write unit
7102	Set low density on write unit
7103	Set high density on write unit
7104	Skip bad spot on selected write unit
7106	Remove interrupt selection on write unit

Note: All codes EXCEPT those specifying a unit number "N" refer to previously selected unit.

1608 FUNCTION CODES

SENSE

74 75	7000	Skip exit on ready to read
	7001	Skip exit on not ready to read
	7002	Skip exit on read parity error
	7003	Skip exit on <u>no</u> read parity error
	7004	Skip exit on read length error
	7005	Skip exit on <u>no</u> read length error
	7006	Skip exit on end-of-file mark
	7007	Skip exit on no end-of-file mark
	7106	Skip exit when read unit is rewinding or is at Load Point
	7107	Skip exit when read unit is not rewinding or is at Load Point
	7102	Skip exit on even bus ready
	7103	Skip exit on not even bus ready
	7104	Skip exit on odd bus ready
	7105	Skip exit on not odd bus ready
74 76	7000	Skip exit on ready to write
	7001	Skip exit on not ready to write
	7002	Skip exit on write reply parity error
	7003	Skip exit on no write reply parity error
	7004	Skip exit on write reply length error
	7005	Skip exit on no write reply length error
	7006	Skip exit on end-of-tape marker
	7007	Skip exit on no end-of-tape marker
	7106	Skip exit when write unit is rewinding
	7107	Skip exit when write unit is not rewinding
	7102	Skip exit on even bus ready
	7103	Skip exit on not even bus ready
	7104	Skip exit on odd bus ready
	7105	Skip exit on not odd bus ready

1609 ADAPTOR CODES

SELECT

74 0	54001	Read Station A
	54002	Read station B
	54003	Read station A and B
	54004	Punch
	54005	Punch and Read A
	54006	Punch and Read B
	54007	Punch, Read A and B

SENSE

74 5	54002	Skip exit on 1604 Selected
	54003	Skip exit on 1604 Not Selected
	64004	Skip exit on unit ready
	64005	Skip exit on unit not ready

SELECT

74 0	54011	Read station A with Interrupt
	54012	Read station B with Interrupt
	54013	Read stations A and B with Interrupt
	54014	Punch with Interrupt
	54015	Punch and Read A with Interrupt
	54016	Punch and Read B with Interrupt
	54017	Punch, Read A and B, with Interrupt

161 TYPEWRITER CODES

SELECT

74 0	4210	Select Typewriter Output
	4220	Select Typewriter Input

Should be 4000 4210
 + 4000 4220

 4240

accdy to change. How does that work?
 Request Typewriter Status

Status, Response
 0000
 0010

26
 Typewriter Ready
 " Not Ready

1610 INSTRUCTIONS

SELECT

74 05 4001	Primary sequence read
4002	Secondary sequence read
4003	Primary and secondary sequence read
4005	Primary sequence read with interrupt
4006	Secondary sequence read with interrupt
4007	Primary and secondary sequence read with interrupt
74 06 4001	Print
4002	Punch
4005	Print with Interrupt
4006	Punch with Interrupt

SENSE

74 76 4002	Skip exit in read ready
4003	Skip exit on read not ready
4002	Skip exit on print ready
4003	Skip exit on print not ready
4004	Skip exit on punch ready
4005	Skip exit on punch not ready

1612 EXF CODES

SELECT

74 0	* y6000	Select Printer
	y6001	Advance Paper One Line
	y6002	Advance Paper Two Lines
	y6003	Skip on Channel 7 (space to selected area of form)
	y6004	Skip on Channel 8 (space to top of form)
	y6006	Suppress Paper Advance
	y6007	Interrupt on Printer Ready
	y6010	Clear Monitor Channels 1-6
	y6011	Select Monitor Channel 1
	y6012	Select Monitor Channel 2
	y6013	Select Monitor Channel 3
	y6014	Select Monitor Channel 4
	y6015	Select Monitor Channel 5
	y6016	Select Monitor Channel 6

SENSE

74 7	y6000	Skip exit on Printer Ready
	y6001	Skip exit on Printer not Ready

* y = Channel designator

BUFFER MODE INSTRUCTIONS

- 74 0 04010 - Select 24-bit mode for channel 1 and 2
- 04011 - Select 160 (12-bit) mode for channel 1 and 2
- 04012 - Select 1604 (48-bit) mode for channel 1 and 2
- 04020 - Select 24-bit mode for channel 3 and 4
- 04021 - Select 160 (12-bit) mode for channel 3 and 4
- 04022 - Select 1604 (48-bit) mode for channel 3 and 4
- 04030 - Select 24-bit mode for channel 1 and 2, 3 and 4
- 04031 - Select 160 (12-bit) mode for channel 1 and 2, 3 and 4
- 04032 - Select 1604 (48-bit) mode for channel 1 and 2, 3 and 4
- 04040 - Select 24-bit mode for channel 5 and 6
- 04041 - Select 160 (12-bit) mode for channel 5 and 6
- 04042 - Select 1604 (48-bit) mode for channel 5 and 6
- 04050 - Select 24-bit mode for channel 1 and 2, 5 and 6
- 04051 - Select 160 (12-bit) mode for channel 1 and 2, 5 and 6
- 04052 - Select 1604 (48-bit) mode for channel 1 and 2, 5 and 6
- 04060 - Select 24-bit mode for channel 3 and 4, 5 and 6
- 04061 - Select 160 (12-bit) mode for channel 3 and 4, 5 and 6
- 04062 - Select 1604 (48-bit) mode for channel 3 and 4, 5 and 6
- 04070 - Select 24-bit mode for channel 1 and 2, 3 and 4, 5 and 6
- 04071 - Select 160 bit mode for channel 1 and 2, 3 and 4, 5 and 6
- 04072 - Select 1604 bit mode for channel 1 and 2, 3 and 4, 5 and 6
- ~~04012 - Clear channel 1 and 2 160 mode select~~
- ~~04011 - Clear channel 1 and 2 1604 mode select~~
- ~~04022 - Clear channel 3 and 4 160 mode select~~
- ~~04021 - Clear channel 3 and 4 1604 mode select~~
- ~~04042 - Clear channel 5 and 6 160 mode select~~
- ~~04041 - Clear channel 5 and 6 1604 mode select~~

Input - Output Typewriter Codes

CHARACTERS		CODE	CHARACTERS		CODE
UC	LC		UC	LC	
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	7	/	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

→ TO COMP

50 characters
14 "spaces"

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		
V	25	record mark	32
W	26	minus zero	52 ?
X	27	plus zero	72 ?
Y	30	group mark	77
Z	31	tape mark	17
0	12		
1	01		

from old manual?

Flexowriter Codes

UC	LC	CODE	UC	LC	CODE
A	a	30	Y	y	25
B	b	23	Z	a	21
C	c	16	o	0	38
D	d	22	1	1	74
E	e	20	2	2	70
F	f	26	3	3	64
G	g	13	4	4	62
H	h	05	5	5	68
I	i	14	6	6	72
J	j	32	7	7	60
K	k	36	8	8	33
L	l	11	9	9	37
M	m	07	Ⓜ	-	52
N	n	06	'	/	44
O	o	03	()*	54 ←
P	p	15	+	, †	46
Q	q	35	=	.	42
R	r	12	:	;	50
S	s	24	CR		45
T	t	01	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
X	x	27			

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

To read tape, hold with forward motion away from you, and with 4-hole side to the left. The left-most hole is the 7th level.

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									
=	^{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

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.

APPENDIX VII

924 REPERTOIRE

FULL-WORD TRANSMISSION

12	LDA	LOAD A	(M) → A
13	LAC	LOAD A, COMPLEMENT	(M) ^f → A
16	LDQ	LOAD Q	(M) → Q
17	LQC	LOAD Q COMPLEMENT	(M) ^f → Q
20	STA	STORE A	(A) → M
21	STQ	STORE Q	(Q) → M
520	XAQ	INTERCHANGE A AND Q	(A) → Q; (Q) → A

ADDRESS TRANSMISSION

53	LIL	LOAD INDEX	(M) → B ^b
57	SIL	STORE INDEX	(B ^b) → m
61	SAL	SUBSTITUTE ADDRESS	(A ₁₄ -A ₀₀) → M
10	ENA	ENTER A	Y → A, Extend sign Y
04	ENQ	ENTER Q	Y → Q, Extend sign Y
26	ATI	A TO INDEX	(A) → B ^b
27	QTI	Q TO INDEX	(Q) → B ^b
50	ENI	ENTER INDEX	y → B ^b ; b = 0 : pass

FULL-WORD ARITHMETIC

14	ADD	ADD	[(A) + (M)] → A
15	SUB	SUBTRACT	[(A) - (M)] → A
24	MUI	MULTIPLY	(M) (A) → QA
25	DVI	DIVIDE	(QA)/(M) → A; Remainder = Q _r
62	TAL	TALLY	Count "1's" in A

ADDRESS ARITHMETIC

11	INA	INCREASE A	[Y + (A)] → A, Extend sign Y
51	INI	INCREASE INDEX	y + (B ^b) → B ^b
56	INQ	INCREASE Q	Y + (Q) → Q

LOGICAL

40	SST	SELECTIVE SET	Set (A_n) for $(M_n) = 1$
42	SCM	SELECTIVE COMPLEMENT	Complement (A_n) for $(M_n) = 1$
41	SCL	SELECTIVE CLEAR	Clear (A_n) for $(M_n) = 1$
43	SSU	SELECTIVE SUBSTITUTE	$(M_n) \rightarrow (A_n)$ for $(Q_n) = 1$
44	LDL	LOAD LOGICAL	$L(Q)(M) \rightarrow A$
45	ADL	ADD LOGICAL	$[(A) + L(Q)(M)] \rightarrow A$
46	SBL	SUBTRACT LOGICAL	$[(A) - L(Q)(M)] \rightarrow A$
47	STL	STORE LOGICAL	$L(Q)(A) \rightarrow M$
52 1	CMA	COMPLEMENT A	$(A)^Y \rightarrow A$
52 2	CMQ	COMPLEMENT Q	$(Q)^Y \rightarrow Q$

SHIFTING

01	ARS	A RIGHT SHIFT	Shift (A) right by K
02	QRS	Q RIGHT SHIFT	Shift (Q) right by K
03	LRS	AQ RIGHT SHIFT	Shift (AQ) right by K
07	LLS	AQ LEFT SHIFT	Shift (AQ) left by K
06	QLS	Q LEFT SHIFT	Shift (Q) left by K
05	ALS	A LEFT SHIFT	Shift (A) left by K
34	SCA	SCALE A	Shift (A) left until $ (A) \geq .5$ or $k = 0$; k - No. of shifts $\rightarrow B^b$
35	SCQ	SCALE AQ	Shift (AQ) left until $ (AQ) \geq .5$ or $k = 0$; k - No. of shifts $\rightarrow B^b$

REPLACE

70	RAD	REPLACE ADD	$[(M) + (A)] \rightarrow M$ and A
71	RSB	REPLACE SUBTRACT	$[(M) - (A)] \rightarrow M$ and A
72	RAO	REPLACE ADD ONE	$[(M) + 1] \rightarrow M$ and A
73	RSO	REPLACE SUBTRACT ONE	$[(M) - 1] \rightarrow M$ and A

STORAGE SEARCH

64	EQS	EQUALITY SEARCH	Search (B^b) words, if $(M-1)$, $(M-2)$, etc. = (A) : Skip Exit
65	THS	THRESHOLD SEARCH	Search (B^b) words, if $(M-1)$, $(M-2)$, etc. > (A) : Skip Exit

66	MEQ	MASKED EQUALITY	Search (B^b) words, if $L(Q)(M-1)$, ($M-2$), etc. = (A) : Skip Exit
67	MTH	MASKED THRESHOLD	Search (B^b) words, if $L(Q)(M-1)$, ($M-2$), etc. > (A) : Skip Exit
63	PTS	PATTERN SEARCH	Search (B^b) words, if $[(A) \vee (M-1),$ ($M-2$)] , etc. = (A) : Skip Exit

STORAGE TEST

36	SSK	STORAGE SKIP	(M_{23}) Neg: Skip Exit; (M_{23}) Pos: Exit
37	SSH	STORAGE SHIFT	(M_{23}) Neg: Skip Exit, left 1; (M_{23}) Pos: Exit, left 1

SKIP

54	ISK	INDEX SKIP	(B^b) = y : Skip NI; (B^b) \neq y : (B^b) + 1 \rightarrow B^b
30	SKH	SKIP HIGH	(B^b) \geq y : Exit (B^b) < y : Skip Exit
31	SKL	SKIP LOW	(B^b) < y : Exit (B^b) \geq y : Skip Exit

JUMPS AND STOPS

22	AJP	A JUMP	Jump to m on condition j
23	QJP	Q JUMP	Jump to m on condition j
75	SLJ	SELECTIVE JUMP	Jump to m on condition j
76	SLS	SELECTIVE STOP	Stop on j, and jump to m
32	UJP	UNCONDITIONAL JUMP	Jump to M; b = 0, 7, jump to m
55	IJP	INDEX JUMP	(B^b) \neq 0 : $B^b - 1 \rightarrow B^b$, jump to m; (B^b) = 0, NI
60	XEC	EXECUTE	Execute instruction at M
33	URJ	UNCONDITIONAL RETURN JUMP	Return jump to M

EXTERNAL FUNCTION

74	EXF	EXTERNAL FUNCTION	j = 1-6 Activate Channel; j = 0 Select conditional y; j = 7 on condition y, exit or skip exit
----	-----	-------------------	---

924 INSTRUCTIONS

00	<i>Fault</i>	Page			Page
01	ARS A Right Shift	2-26		41	SCL Selective Clear 2-23
02	QRS Q Right Shift	2-26		42	SCM Selective Complement 2-22
03	LRS AQ Right Shift	2-26		43	SSU Selective Substitute 2-24
04	ENQ Enter Q	2-15		44	LDL Load Logical 2-24
05	ALS A Left Shift	2-27		45	ADL Add Logical 2-24
06	QLS Q Left Shift	2-27		46	SBL Subtract Logical 2-24
07	LLS AQ Left Shift	2-26		47	STL Store Logical 2-24
10	ENA Enter A	2-15		50	ENI Enter Index 2-16
11	INA Increase A	2-21		51	INI Increase Index 2-21
12	LDA Load A	2-11		52 0	XAQ Interchange A and Q 2-12
13	LAC Load A Complement	2-11		52 1	CMA Complement A 2-25
14	ADD Add	2-17		52 2	CMQ Complement Q 2-25
15	SUB Subtract	2-17		53	LIL Load Index 2-13
16	LDQ Load Q	2-11		54	ISK Index Skip 2-31
17	LQC Load Q Complement	2-11		55	IJP Index Jump 2-38
20	STA Store A	2-12		56	INQ Increase Q 2-21
21	STQ Store Q	2-12		57	SIL Store Index 2-13
22	AJP A Jump	2-36, 40		60	XEC Execute 2-41
23	QJP Q Jump	2-37, 40		61	SAL Substitute Address 2-13
24	MUI Multiply	2-18		62	TAL Tally 2-20
25	DVI Divide	2-19		63	PTS Pattern Search 2-34
26	ATI A to Index	2-16		64	EQS Equality Search 2-33
27	QTI Q to Index	2-16		65	THS Threshold Search 2-33
30	SKH Skip High	2-31		66	MEQ Masked Equality 2-34
31	SKL Skip Low	2-31		67	MTH Masked Threshold 2-34
32	UJP Unconditional Jump	2-37		70	RAD Replace Add 2-29
33	URJ Unconditional Return Jump	2-41		71	RSB Replace Subtract 2-29
34	SCA Scale A	2-28		72	RAO Replace Add One 2-29
35	SCQ Scale AQ	2-28		73	RSO Replace Subtract One 2-29
36	SSK Storage Skip	2-30		74	EXF External Function 3-3
37	SSH Storage Shift	2-30		75	SLJ Selective Jump 2-37, 40
40	SST Selective Set	2-22		76	SLS Selective Stop 2-37, 40
				77	<i>Fault</i> 2-1

CONTROL DATA

CORPORATION

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