

3400

**Control Data[®] 3400 Computer System
Reference Manual**

INDEX TO INSTRUCTIONS (MNEMONIC CODES)

<u>Mnemonic Code</u>	<u>Operation</u>	<u>Page</u>	<u>Mnemonic Code</u>	<u>Operation</u>	<u>Page</u>
ADD	Add	3-11	LIBJP	Unconditional Jump	
ADL	Add Logical	3-14	LIL	Load Index (lower)	3-11
AJP	A Jump	3-17, 3-18	LIU	Load Index (upper)	3-10
ALS	A Left Shift	3-15	LLS	Long Left Shift (AQ)	3-15
ARS	A Right Shift	3-15	LQC	Load Q, Complement	3-10
ATI	Transmit A to Index	3-11	LRS	Long Right Shift (AQ)	3-15
AUG	Augment	3-20	MEQ	Masked Equality Search	3-16
BEGR	Read	5-4	MTH	Masked Threshold Search	3-16
BEGW	Write	5-4	MUI	Multiply Integer	3-12
BJPL	Unconditional Jump to Lower	3-19	QJP	Q Jump	3-17, 3-18
BRTJ	Unconditional Return Jump	3-19	QLS	Q Left Shift	3-15
CCWD	Change Control Word	5-5	QRS	Q Right Shift	3-15
CIS	Copy Interrupt Status	4-14	RAD	Replace Add	3-15
CLCH	Clear Channel	5-7	RAO	Replace Add One	3-15
CMS	Copy Mask Status	4-15	RSB	Replace Subtract	3-15
CONN	Connect	5-2	RSO	Replace Subtract One	3-16
COPY	Copy Status	5-6	SAL	Substitute Address (Lower)	3-11
CPR	Copy Product Register	4-15	SAU	Substitute Address (Upper)	3-11
DVI	Divide Integer	3-12	SBL	Subtract Logical	3-14
ELB	Enter Lower Bound	3-22	SCA	Scale A	3-15
ENA	Enter A	3-11	SCL	Selective Clear	3-14
ENI	Enter Index	3-11	SCM	Selective Complement	3-14
ENQ	Enter Q	3-11	SCQ	Scale AQ	3-15
EQS	Equality Search	3-16	SEN	Internal Sense	3-14
EUB	Enter Upper Bound	3-22	SIL	Store Index (Lower)	3-11
EXTF	Function	5-3	SIU	Store Index (Upper)	3-11
FAD	Floating Add	3-12	SLJ	Selective Jump	3-17, 3-19
FDV	Floating Divide	3-13	SLS	Selective Stop	3-17, 3-19
FMU	Floating Multiply	3-13	SSH	Storage Shift	3-16
FSB	Floating Subtract	3-12	SSK	Storage Skip	3-16
IAQ	Interchange A & Q	3-10	SST	Selective Set	3-14
IJP	Index Jump	3-17	SSU	Selective Substitute	3-14
INA	Increase A	3-13	STA	Store A	3-10
INF	Internal Function	4-11	STCH	Store Character	3-20
INI	Increase Index	3-13	STL	Store Logical	3-14
ISK	Index Skip	3-13	STQ	Store Q	3-10
LAC	Load A, Complement	3-10	SUB	Subtract	3-11
LDA	Load A	3-10	THS	Threshold Search	3-16
LDCH	Load Character	3-20	UBJP	Unconditional Jump	3-19
LDL	Load Logical	3-14			
LDQ	Load Q	3-10			

CONTENTS

Chapter I – Basic System Description		Jumps and Stops	3-17
3400 System Characteristics	1-1	Normal Jump	3-17
Basic 3400 System	1-2	Return Jump	3-18
Options	1-2	Bank Jumps	3-19
Chapter II – Storage Section		Character Handling	3-20
Storage Word	2-1	Chapter IV – Interrupt System	
Storage Addressing	2-1	Logical Description of Interrupt System	4-1
Chapter III – Computation Section		Interrupt Register	4-1
Logical Description	3-1	Interrupt Mask Register	4-1
Arithmetic Section	3-1	Product Register	4-1
Control Section	3-2	Categories of Interrupts	4-1
Description of Instructions	3-2	Category I	4-2
Class I	3-3	Category II	4-2
Class II	3-3	Category III	4-3
Class III	3-3	Programming Cautions	4-3
Class IV Miscellaneous	3-3	Chapter V – Input/Output	
Description of Designators	3-4	Input/Output Instructions	5-2
Address Modification	3-4	Control Word	5-5
Address Modification Modes	3-5	Auto-Load	5-7
Execution of Instructions	3-5	Chapter VI – Parity	
Symbols	3-6	Parity Generation	6-1
Order of Instructions in Basic Computer	3-6	Data Parity Generation	6-1
Instructions Added by Floating Point Option	3-10	Parity Generation for I/O Channel Transmissions	6-1
Inter-register Transmission	3-10	Parity Checking	6-1
Full-Word Transmission	3-10	Parity Checking on Storage or Transmission of Data	6-1
Address Transmission	3-10	Parity Checking on I/O Channel Transmissions	6-2
Fixed Point Arithmetic	3-11	Parity Errors	6-2
Single Precision Floating Point Arithmetic	3-12	Operand Parity Error	6-2
Address Arithmetic	3-13	Instruction Parity Error	6-2
Logical	3-13	I/O Channel Transmission Parity Error	6-2
Shifting	3-14	Interrupt Selection on I/O Parity Error	6-3
Scale	3-15	External Equipment Parity Error	6-3
Replace	3-15		
Storage Test	3-16		
Storage Search	3-16		

Chapter VII – 3401 Console

Switches	7-1
Indicators	7-2
Typewriter	7-5
Connect	7-5
Function	7-5
Status	7-5
Programming	7-7
Set Tabs, Margins, and Spacing	7-7
Clear	7-7
Connect	7-8
Check Status	7-8
Function	7-8
Write	7-8
Read	7-8

3404 Maintenance Panel and Power Control Panel	7-9
System Status Display Panel	7-10
Switches and Indicators	7-10
Circuit Breakers	7-17
Meters and Dials	7-17

Glossary

Appendices

- A. Interruptible Conditions and Faults
- B. Control Data 3400 Computer Instructions
- C. Instruction Execution Times
- D. Number Systems
- E. Table of Powers of 2
- F. Octal-Decimal Conversion Table
- G. Octal-Decimal Fraction Conversion Table

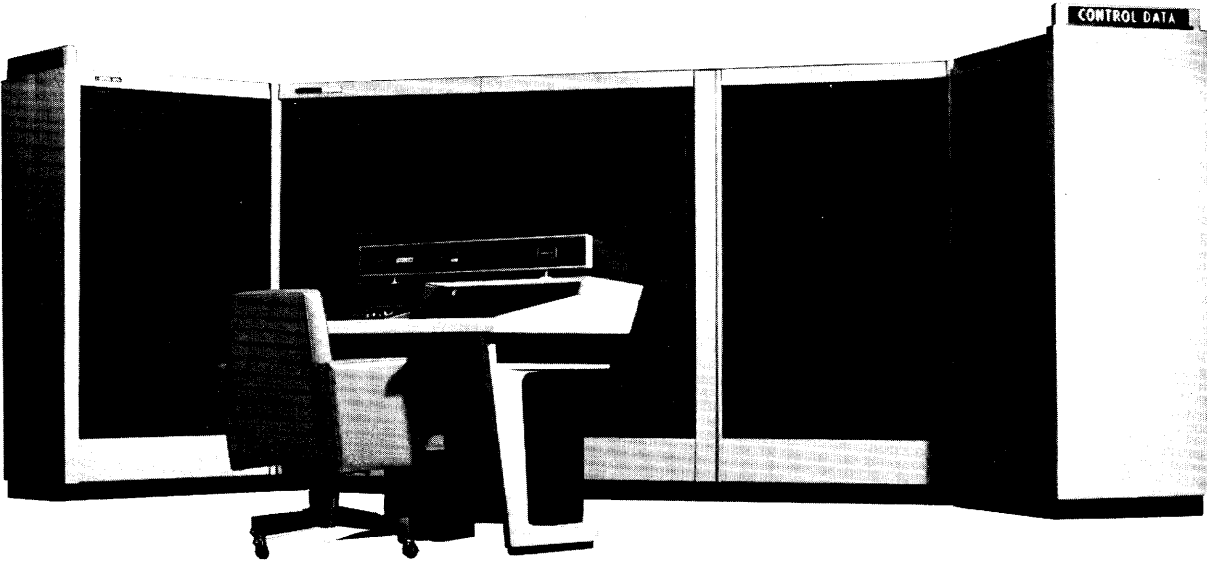
FIGURES

1-1	3400 Computer System
3-1	Return Jump
4-1	Interrupt Example 1
4-2	Interrupt Example 2
4-3	Interrupt Example 3
4-4	Interrupt Example 4
4-5	Interrupt Example 5
5-1	3400 System
7-1	System Status Display Panel
7-2	3404 Maintenance Panel
7-3	3400 Power Control Panel

TABLES

1-2	3-1	Arithmetic Properties of Registers	3-1
3-18	4-1	Assignment of Bits	4-2
4-4	4-2	Interrupt Example 1	4-5
4-6	4-3	Interrupt Example 3	4-7
4-7	4-4	Interrupt Example 4	4-9
4-8	4-5	Interrupt Example 5	4-10
4-10	7-1	Console Switches	7-1
5-1	7-2	Display Panel Indicators	7-3
7-2	7-3	731 Typewriter Codes	7-6
7-9	7-4	Connect, Function, and Status Codes	7-7
7-10	7-5	3404 Switches and Indicators	7-10

**CONTROL DATA® 3400 COMPUTER SYSTEM
REFERENCE MANUAL**



3400 COMPUTER

PREFACE

This manual provides information for the machine-language use of the 3400 computer. Its intention is to describe the capabilities of the hardware. Options and constraints for programming are noted. Some programming examples are given to illustrate how instructions perform.

Other than using COMPASS mnemonics to abbreviate titles of instructions, no software systems are used in describing instructions.

CHAPTER I

BASIC SYSTEM DESCRIPTION

The CONTROL DATA* 3400 is a solid-state, stored-program, general-purpose digital computing system, with large storage capacity and fast data transmission and computation speeds.

The 3400 system incorporates features of the CONTROL DATA 3600 Computer to provide program compatibility with this machine. With the several available options, a variety of configurations is possible.

3400 SYSTEM CHARACTERISTICS

Stored-program, general-purpose computer	Character handling instructions
Parallel mode of operation	Indexing
Single address logic	Storage searching
51-bit storage word (48 bits of data, 3 parity bits)	Binary arithmetic
Six 15-bit index registers	Modulus $2^{48} - 1$ (one's complement) for single precision operations
Indirect addressing	Completely solid-state
Magnetic core storage	Diode logic
32,768 51-bit words	Transistor amplifiers
16,384 51-bit words	Ready access to circuits
Input/Output	Console includes:
Transmission of 48-bit words (12-bit bytes)	I/O Typewriter and display panel
Up to four separate bidirectional input/output channels	Inter-computer communication
System interrupt	3400 ↔ 3400/3600/3800 } Via 3682 Satellite
Flexible repertoire of instructions	3400 ↔ 3100/3200/3300 } Coupler
Fixed point arithmetic (integer)	3400 ↔ 160/160-A } Via 3682 Satellite
Single-precision, floating-point arithmetic (optional)	Via 3681 Data Channel Converter
Logical and masking operations	

* Registered trademark of Control Data Corporation.

BASIC 3400 SYSTEM

The basic 3400 system consists of a central computer, an input/output section, magnetic core storage, and a console. Over-all system operation depends on the integral operation of these elements. (See figure 1-1.)

Included in the basic computing system are operator and maintenance consoles. The consoles contain all the controls and indicators necessary to operate the system.

A set of basic programs is provided with the basic 3400 system consisting of an elemental assembler and operating system.

OPTIONS

For greater systems capability, the 3400 computing

system may be expanded with several available options. These options are:

1) Floating point option (3410)

This adds four single-precision, floating-point instructions.

2) 16,384 word storage

The standard 3400 system includes 32,768 words of core storage. An optional system is available with only 16,384 words of core storage. Standard programming systems for the 3400 require the use of the 3409 storage option; basic programs do not require the 3409.

3) Additional Input/Output channels (3406)

The basic system may be expanded to a maximum of four bidirectional I/O channels.

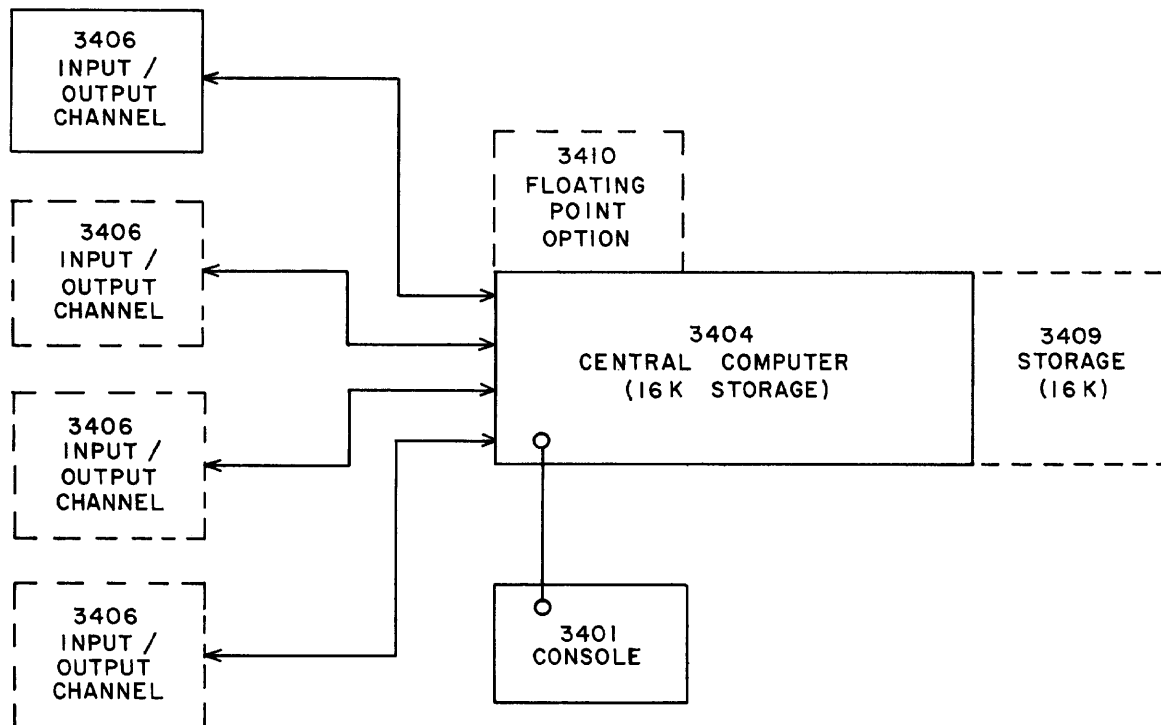


Figure 1-1. 3400 Computer System
(Dotted Lines Indicate Optional Additions)

CHAPTER II

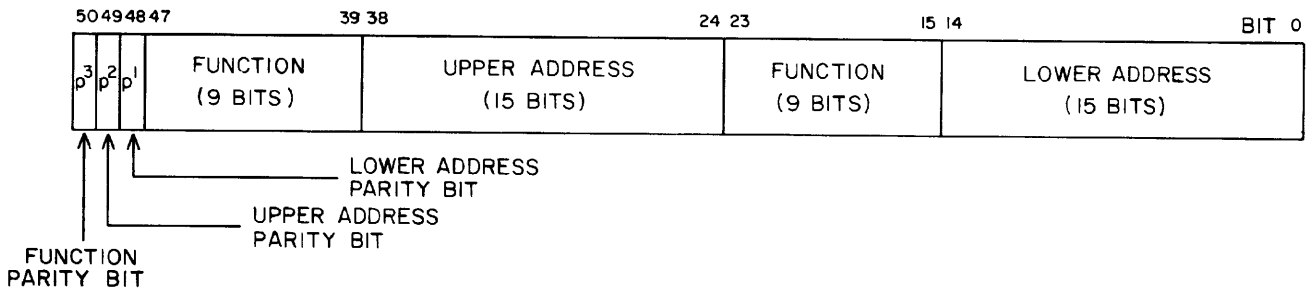
STORAGE SECTION

The magnetic core storage section provides high-speed, random-access storage for 32,768 words. The storage section consists of the storage elements themselves, and the circuitry for addressing the storage elements.

STORAGE WORD

A storage word may be two 24-bit instructions, a single 48-bit instruction, or a 48-bit data word.

Three parity bits are appended to each 48-bit word; thus a storage word is 51 bits in length. The format of a typical storage word is diagrammed below.



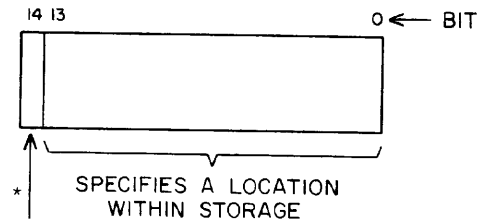
The storage word is divided into three portions:

- 1) a 15-bit lower address,
- 2) a 15-bit upper address, and
- 3) an 18-bit function portion, distributed in the storage word as diagrammed. A parity bit accompanies each of these portions when the word is stored. The parity bit (P1) associated with the lower address portion is placed in bit 48 of the storage word, parity bit P2 (upper address) is placed in bit 49, and parity bit P3 (function) is placed in bit 50.

When part of the word or the entire 51 bits is read from storage, the appropriate parity bit(s) accompanies the word and the computer checks the word for parity.

STORAGE ADDRESSING

The location of each word in storage is identified by an assigned number (address). An address consists of 15 bits interpreted as shown below:



The following storage locations are used with return jump instructions for interrupt processing (unless the

* 1) In a 3400 system with 32,768 words of core storage: If this bit is a "0", it specifies the lower section of storage which contains addresses 00000₈--37777₈; if this bit is a "1", it specifies the upper section of storage which contains addresses 40000₈--77777₈.

2) In a 3400 system with 16,384 words of core storage, this bit is always "0". It specifies the one storage section which contains addresses 00000₈--37777₈.

interrupt category is removed by adding an option):

00007 } Category I Interrupt

00030 }
00031 } Category II Interrupt
00032 }
00033 }

00020 } Category III Interrupt

See Chapter 4 for additional information on the uses of these storage locations.

Address 00000g is reserved for use with a return jump instruction when the Restart switch is depressed. (See chapter 7.)

CHAPTER III

COMPUTATION SECTION

The computation section performs calculations and processes data in a parallel binary mode through the step-by-step execution of individual instructions. The instructions and data are stored in the storage section(s).

LOGICAL DESCRIPTION

The computation section may be functionally divided into an arithmetic section and a control section.

Arithmetic Section

The arithmetic section performs the arithmetic and logical operations necessary for executing instructions. It consists primarily of several operational registers. The operational registers are described below. Table 3-1 lists the arithmetic properties of the registers.

A Register

Nearly all arithmetic and logical operations use the 48-bit A register (Arithmetic register). The contents of this register may be shifted to the right or to the left, separately or in conjunction with the Q register. In certain conditional instructions, the A register is used to hold control quantities which govern operations.

Q Register

The 48-bit Q register (Auxiliary Arithmetic register) assists the A register in performing arithmetic and logical operations. The contents of the Q register may be shifted right or left, separately or in conjunction with the A register. Q may also be used with the A register to form a double length register, AQ or QA. In addition to assisting the A register, certain instructions reference the Q register directly.

P Register

The 15-bit P register functions as a program address counter. The P register holds the address of each program step. After executing the instruction (or instructions) contained in the program step, the quantity in P is advanced by one to the address of the next instruction.

When a jump condition is met, the P register is set to the quantity specified by the execution address of

Table 3-1. Arithmetic Properties of Registers

Register	No. of Stages	Modulus	Complement Notation	Arithmetic	Result
A Register	48	$2^{48}-1$	one's	subtractive	signed*
Q Register	48	$2^{48}-1$	one's	----	signed
P Register	15	2^{15}	two's	additive	unsigned
Index Registers	15	$2^{15}-1$	one's**	----	----

* The result of an arithmetic operation in A satisfies $A \leq 2^{47}-1$ since A is always treated as a signed quantity. When the result in A is zero, it is always represented as 000...000 except when 111...111 is added to 111...111. In this case, the result is 111...111 (negative zero).

**Though the index registers have no arithmetic capabilities themselves, address modification using the index registers is performed modulus $2^{15}-1$ (one's complement).

the jump instruction. If the instruction is a return jump, the contents of P are stored before executing the jump, permitting a return to the program sequence after the jump is made.

Since the P register is a two's complement additive register, it can generate storage addresses in sequence from 00000 to 77777₈. When a count of 77777₈ is reached, the next count in P reduces its value to 00000. (Note that in generating storage addresses by adding the contents of an index register to a base quantity, address 77777₈ usually cannot be reached. Refer to the section on Address Modification Modes, (p. 3-5).

U Register

The 48-bit Program Control register (U) holds the program step while it is being executed. All operations necessary to execute an instruction are governed by the contents of this register.

B¹ - B⁶ (Index Registers)

Six 15-bit index registers may be used to:

- 1) Hold quantities used as address modifiers.
- 2) Hold control quantities for certain instructions.

The index registers may also be explicitly referenced by certain instructions (refer to Repertoire of Instructions section).

Interrupt Register

Each interruptible condition in the system is connected to a particular bit position of the Interrupt register. The lower bit positions detect internal interrupt conditions such as overflow, divide fault, and exponent fault. The upper bit positions are interrupt lines coming from each of the four possible communication channels.

Interrupt Mask Register

This register enables testing of external interrupt lines and internal conditions. The bit positions of this register match the Interrupt register. Interrupt occurs on a condition if the bit of the Mask register is set to "1". The Internal Function instruction may be used to set or clear bits in this register.

Product Register

The product register contains the bit-by-bit logical product of the Interrupt register and the Interrupt Mask register (refer to the Interrupt section).

Bounds Registers (Upper and Lower)

Two bounds registers serve as a memory and jump lock-out. The Lower bounds register holds an 8-bit Lower bound address. The Upper bounds register holds an 8-bit upper bound address. The upper 8 bits of the 15-bit storage address S are compared with the contents of the bounds registers when bounds checking is in effect.

Control Section

The control section of the computer directs the operations required to execute instructions and establishes the timing relationships needed to perform these operations in the proper sequence. It also sends the preliminary commands necessary to begin the processing of input/output data.

The control section acquires an instruction from storage, interprets it, and sends the necessary commands to other sections. A program step may be a single 48-bit instruction or a pair of 24-bit instructions which together occupy a single storage location as a 48-bit word.

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 program steps. Usually, at the completion of each program step, the count in P is advanced by one to specify the address of the next program step.

The Program Control register, U, holds a program step while it is being executed. If the program step is a pair of 24-bit instructions, the upper instruction is executed first, followed by the lower instruction.

DESCRIPTION OF INSTRUCTIONS

A computer word consists of 48 bits and may be interpreted as one 48-bit data word, a 48-bit instruction, or two 24-bit instructions.

Most instructions designated by three-letter mnemonic codes are 24-bit instructions common to the 1604 and 3600 computers. These instructions are arranged in a 48-bit word; the higher order 24 bits are called the upper instruction and the lower order 24 bits are called the lower instruction.

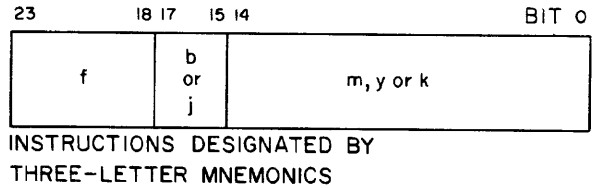
Instructions which are not common to the 1604 computer and designated by mnemonic codes of three or four letters, differ in format and in word length (some are 24 bits; others are 48 bits).

Instruction formats are arranged in four major classes, according to differences in word length and the

position of the function code within the format. A typical format from each class is outlined below. Designators used within these formats are explained at the end of this section. For a comprehensive description of instructions, refer to the Repertoire of Instructions section.

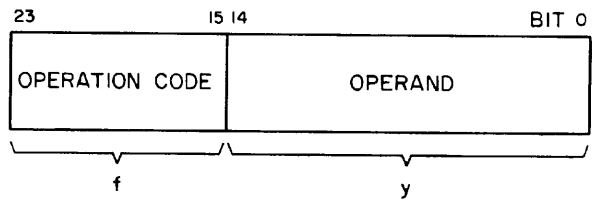
Class I

Class I instruction formats are 24 bits in length and have 6-bit function codes, 'f'. All instructions common to the 1604 and 3600 computers and designated by three-letter mnemonic codes are included in this category.



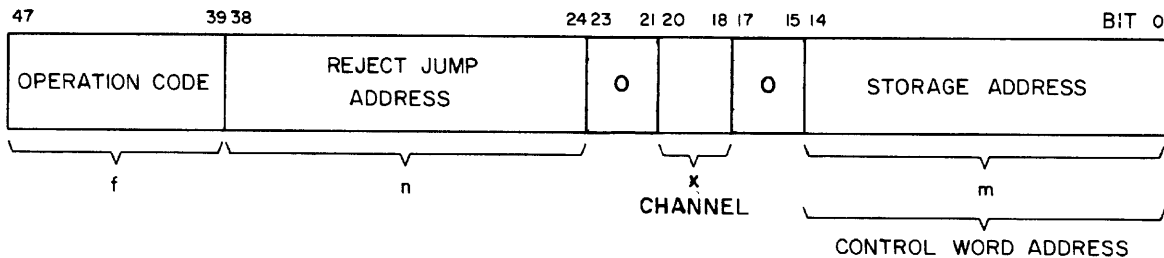
Class II

Class II instruction formats are 24 bits in length and have 9-bit function codes. All instructions in this category are designated by mnemonic codes of three letters.



Class III

Class III instruction formats are 48 bits in length and have 9-bit function codes. All instructions in this category are designated by mnemonic codes of four letters.



Class IV Miscellaneous

See pages 3-20, 3-21, and 3-22.

DESCRIPTION OF DESIGNATORS

Designators used throughout the Description of Instructions section and in instruction formats are explained below. For specific interpretations of designators, refer to the individual instructions.

<u>Designator</u>		<u>Use</u>
b	Index	Specifies index register (B) used, or whose contents are used in the operation.
c	Connect and Function	Specifies codes used in Connect and Function instructions.
f	Function Code	A 6 or 9-bit code (depending on the operation) which specifies the operation to be performed.
j	Condition	Conditions operations in jumps and stops.
k	Unmodified Shift Count	Number of shifts to be executed.
K	Modified Shift Count	$[K = k + (B^b)]$
m	Unmodified Execution Address	Address of operand.
M	Modified Execution Address	$[M = m + (B^b)]$
n	Reject Jump Address	Used with I/O instructions.
v	Second Index	Specifies second index register (V) used, or whose contents are used in the operation.
w	Word Count	A 15-bit quantity which specifies the number of words to be processed in a data transmission operation.
x	Channel Number	Specifies I/O channel; also used to specify channel whose status will be read or sensed.
y	Unmodified Operand	Used in execution address portion of instruction; specifies this address will be used as the operand.
Y	Modified Operand	$[Y = y + (B^b)]$

ADDRESS MODIFICATION

The portion of the instruction word designated by 'm', 'y', or 'k', is often termed the base 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 be modified or left unmodified depending on the index designator. The execution address is modified by adding the contents of the designated index register to the execution address. If left unmodified, the lower-case symbols 'k', 'y', or 'm', are used. If the address is modified, the symbols are capitalized.

The modified shift count 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 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 $Y = y$.

SYMBOLS

The following symbols are used in the Order of Instructions section.

A	The A register
A_n	The binary digit in position 'n' of the A register
B^b	Designated index register
LA	Lower address; execution address portion of lower instruction of a program step
Q	Auxiliary Arithmetic register
UA	Upper address
()	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
#	A flag to denote the instruction must be located in the upper instruction position of an instruction word
∨	The logical inclusive OR function
⊖	The logical exclusive OR function
∧	The logical AND function

ORDER OF INSTRUCTIONS IN BASIC COMPUTER

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage * References	Address Modification	Number of Instruction Bits
<u>Inter-Register Transmission</u>						
	IAQ	Interchange A and Q (00700554)	No	0	No	24
<u>Full-Word Transmission</u>						
12	LDA	Load A	Yes	1	Yes	24
16	LDQ	Load Q	Yes	1	Yes	24
20	STA	Store A	Yes	1	Yes	24
21	STQ	Store Q	Yes	1	Yes	24
13	LAC	Load A, Complement	Yes	1	Yes	24
17	LQC	Load Q, Complement	Yes	1	Yes	24

* If indirect addressing is designated, at least one additional storage reference is required.

ORDER OF INSTRUCTIONS (Cont'd)


Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
<u>Address Transmission</u>						
61	SAL	Substitute Address (lower)	Yes	1	Yes	24
60	SAU	Substitute Address (upper)	Yes	1	Yes	24
04	ENQ	Enter Q	Yes	0	Yes	24
10	ENA	Enter A	Yes	0	Yes	24
53	LIL	Load Index (lower)	Yes	1	No	24
52	LIU	Load Index (upper)	Yes	1	No	24
57	SIL	Store Index (lower)	Yes	1	No	24
56	SIU	Store Index (upper)	Yes	1	No	24
50	ENI	Enter Index	Yes	0	No	24
	ATI	Transmit A to Index (0074054b)	No	0	No	24
<u>Fixed Point Arithmetic</u>						
14	ADD	Add	Yes	1	Yes	24
15	SUB	Subtract	Yes	1	Yes	24
24	MUI	Multiply Integer	Yes	1	Yes	24
25	DVI	Divide Integer	Yes	1	Yes	24
<u>Address Arithmetic</u>						
11	INA	Increase A	Yes	0	Yes	24
51	INI	Increase Index	Yes	0	No	24
54	ISK	Index Skip	Yes	0	No	24
<u>Logical</u>						
40	SST	Selective Set	Yes	1	Yes	24
41	SCL	Selective Clear	Yes	1	Yes	24
42	SCM	Selective Complement	Yes	1	Yes	24
43	SSU	Selective Substitute	Yes	1	Yes	24
44	LDL	Load Logical	Yes	1	Yes	24
45	ADL	Add Logical	Yes	1	Yes	24
46	SBL	Subtract Logical	Yes	1	Yes	24
47	STL	Store Logical	Yes	1	Yes	24

ORDER OF INSTRUCTIONS (Cont'd)

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
<u>Shifting</u>						
01	ARS	A Right Shift	Yes	0	Yes	24
02	QRS	Q Right Shift	Yes	0	Yes	24
03	LRS	Long Right Shift (AQ)	Yes	0	Yes	24
05	ALS	A Left Shift	Yes	0	Yes	24
06	QLS	Q Left Shift	Yes	0	Yes	24
07	LLS	Long Left Shift (AQ)	Yes	0	Yes	24
34	SCA	Scale A	Yes	0	No	24
35	SCQ	Scale AQ	Yes	0	No	24
<u>Replace</u>						
70	RAD	Replace Add	Yes	2	Yes	24
71	RSB	Replace Subtract	Yes	2	Yes	24
72	RAO	Replace Add One	Yes	2	Yes	24
73	RSO	Replace Subtract One	Yes	2	Yes	24
<u>Storage Test</u>						
36	SSK	Storage Skip	Yes	1	Yes	24
37	SSH	Storage Shift	Yes	2	Yes	24
<u>Search</u>						
64	EQS	Equality Search	Yes	N	Yes	24
65	THS	Threshold Search	Yes	N	Yes	24
66	MEQ	Masked Equality Search	Yes	N	Yes	24
67	MTH	Masked Threshold Search	Yes	N	Yes	24
<u>Jumps and Stops</u>						
22	AJP	A Jump	No	1*	No	24
23	QJP	Q Jump	No	1*	No	24
55	IJP	Index Jump	Yes	0	No	24
75	SLJ	Selective Jump	No	1*	No	24
76	SLS	Selective Stop	No	1*	No	24
63.0	UBJP	Unconditional Jump	Yes	0	Yes	48
63.0	BRTJ	Unconditional Return Jump	Yes	1	Yes	48
63.1	BJPL	Unconditional Jump to Lower	Yes	0	Yes	48

* Return jump only

ORDER OF INSTRUCTIONS (Cont'd)

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
<u>Input/Output</u>						
74.0	CONN	Connect	No	0	No	48
74.1	EXTF	Function	No	0	No	48
74.2	BEGR	Read	No	1	No	48
74.3	BEGW	Write	No	1	No	48
74.4	COPY	Copy Status	No	0	No	48
74.5	CLCH	Clear Channel	No	0	No	48
74.6	CCWD	Change Control Word	No	0	No	48
77.2	CIS	Copy Interrupt Status	No	0	No	24
77.2	CMS	Copy Mask Status	No	0	No	24
77.3	SEN	Internal Sense	No	0	No	24
77.4	CPR	Copy Product Register	No	0	No	24
77.0	INF	Internal Function	No	0	No	24
<u>Miscellaneous</u>						
77.1	AUG	Augment	Yes	0	No	24
77.5	EUB	Enter Upper Bound	No	0	No	24
77.6	ELB	Enter Lower Bound	No	0	No	24
	LDCH	Load Character (63bv 0006500m)	Yes	1	Yes	48
	STCH	Store Character (63bv 0006505m)	Yes	2	Yes	48
<u>Illegal Codes</u>						
00*		Return Jump to Address 00020				
26						
27						
62						
63*						
74.0* → 74.6*						
74.7						
77.0*						
77.5*						
77.6*						
77.7						

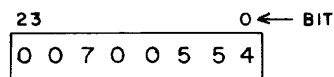
* See instruction description

INSTRUCTIONS ADDED BY FLOATING POINT OPTION

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
<u>Single Precision Floating Point Arithmetic</u>						
30	FAD	Floating Add	Yes	1	Yes	24
31	FSB	Floating Subtract	Yes	1	Yes	24
32	FMU	Floating Multiply	Yes	1	Yes	24
33	FDV	Floating Divide	Yes	1	Yes	24

INTER-REGISTER TRANSMISSION

IAQ Interchange A and Q



The 24-bit Interchange A and Q instruction interchanges the contents of the A and Q registers. The 00 instruction is illegal if bits 00 → 17 contain other than the exact code shown (with the exception of the transmit A to Index instruction).

FULL-WORD TRANSMISSION

In Full-Word Transmission instructions, a 48-bit operand or data word is used in executing the instruction.

LDA Load A Op. Code 12

Replaces the contents of A with a 48-bit operand contained in the storage location specified by M. Negative zero is formed in A if the operand at M is equal to negative zero.

LAC Load A Complement Op. Code 13

Replaces the contents of A with the complement of a 48-bit operand contained in the storage location specified by M. Negative zero is formed in A if the operand at M is equal to positive zero.

LDQ Load Q Op. Code 16

Replaces the contents of Q with a 48-bit operand contained in the storage location specified by M. Negative zero is formed in Q if the operand at M is equal to negative zero.

LQC Load Q Complement Op. Code 17

Replaces the contents of Q with the complement of a 48-bit operand contained in the storage location specified by M. Negative zero is formed in Q if the operand at M is equal to positive zero.

STA Store A Op. Code 20

Replaces the contents of the designated storage location, M, with the contents of A.

STQ Store Q Op. Code 21

Replaces the contents of the designated storage location, M, with the contents of Q.

ADDRESS TRANSMISSION

- 1) In the Address Transmission instructions, only the lower 15 bits of a 24-bit word instruction or data word are used.
- 2) In the LIU and LIL instructions, an index designation of "0" has no meaning and should not be used. If used, these instructions become pass instructions, but use some time in storage reference. The next instruction is then executed. Using "0" as an index designation does not constitute a fault.

LIU Load Index Upper Op. Code 52

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

LIL Load Index Lower Op. Code 53

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

SIU Store Index Upper Op. Code 56

Replaces the upper address portion of storage location 'm' with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If $b = 0$, (m_{10}) is cleared.

SIL Store Index Lower Op. Code 57

Replaces the lower address portion of storage location 'm' with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If $b = 0$, (m_{10}) is cleared.

SAU Substitute Address Upper Op. Code 60

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

SAL Substitute Address Lower Op. Code 61

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

ENQ Enter Q Op. Code 04

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

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

ENA Enter A Op. Code 10

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

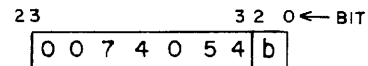
the remaining 33 bits. Negative zero will be formed in A if:

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

ENI Enter Index Op. Code 50

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

ATI Transmit A to Index



Transmits the lower 15 bits of A to the index register specified by b. The quantity designated by b may have values 1--6; ($b = 0, 7$ is an illegal instruction.) The contents of the A register are not modified. The 00 instruction is illegal if bits 00-17 contain other than the exact code shown (with the exception of the IAQ instruction).

FIXED POINT ARITHMETIC

- 1) If the capacity of the A register, $\pm(2^{47} - 1)$, is exceeded during the execution of the Fixed Point Arithmetic instructions (ADD, SUB, RAD, RSB, RAO, and RSO), an arithmetic overflow fault is produced. When executing the DVI instruction, if the result exceeds the capacity of the A register, $\pm(2^{47} - 1)$, a divide fault is produced (refer to appendix A).
- 2) The Multiply Integer instruction (MUI) uses the double register configuration QA. The least significant bit of the product is left in bit position A_{00} . The most significant bit may be in either A or Q, depending upon the magnitude of the product.

ADD Add Op. Code 14

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

SUB Subtract Op. Code 15

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

MUI Multiply Integer Op. Code 24

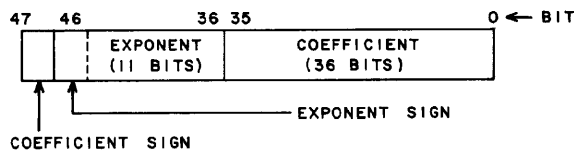
Forms a 96-bit product from two 48-bit operands. The multiplier must be loaded into A prior to execution of the instruction. The execution address specifies the storage location of the multiplicand. The product is contained in QA as a 96-bit quantity. The operands are considered as integers and the binary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

DVI Divide Integer Op. Code 25

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

SINGLE PRECISION FLOATING POINT ARITHMETIC (Floating Point Option)

All 3400 floating point numbers have the following format:



- 1) Operands do not have to be normalized. However, in a FDV operation, the coefficient of the dividend must be less than two times the coefficient of the divisor. If not, a divide fault will result.
- 2) When a right shift is required to normalize, it is performed whether the augment bit t^1 ($t^1 = "1"$ for un-normalized arithmetic) equals "0" or "1".
- 3) When a number is shifted left to normalize, sign bits are entered into the least significant positions.

- 4) The residue in the Q register is not altered during during normalize or round operations.
- 5) On an exponent underflow, A and Q are cleared after the end sign correction. On an exponent overflow, the result is left as formed.
- 6) When the coefficient of the result is ± 0 , the exponent portion of A is cleared before the end sign correction.
- 7) Rounding is accomplished by adding ± 1 to the A register.
- 8) Floating point range faults (overflow/underflow) occur if the exponent exceeds $\pm (2^{10}-1)$.

FAD Floating Add (FPO) Op. Code 30

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

If the exponent difference $\leq 36_{10}$, the coefficient with the smallest exponent is placed in A. It is then long right shifted into AQ until the exponents are equal. The bits (residue) in Q are not altered during the remainder of the instruction.

If the difference in exponents is greater than 36_{10} , no shift occurs. The coefficient of the greatest exponent is used as the basis for the answer. For these cases, the residue in Q is always positive 0.

The round decision (to add -1, +1, or 0) is made after the shift to equalize exponents and before the addition of the coefficients.

Add +1 to A if	$A_{47} = 0$	$Q_{47} = 1$
Add -1 to A if	$A_{47} = 1$	$Q_{47} = 0$
Add 0 to A if	$A_{47} = Q_{47}$	

There is never a round when the exponents are equal if the difference in exponents is greater than 36_{10} .

FSB Floating Subtract (FPO) Op. Code 31

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

The same rules mentioned previously in exponent alignment and rounding apply to FSB. FSB is identical to FAD with the exception that the operand read from memory is complemented before the execution of the instruction.

FMU Floating Multiply (FPO)
Op. Code 32

Forms the product of an operand in floating point format with the contents of A also in floating point format. The operand is read from storage location M. The product is rounded and normalized if necessary and retained in A.

The magnitudes of the coefficients are multiplied to obtain a 72-bit product in AQ (lower 36 bits of A and upper 36 bits of Q). Q is not changed after the iterative multiply operation, and thus the residue is an unsigned magnitude quantity (significant bits are "1's").

The round decision (to add 0 or +1) is made after the iterative multiply operation. The result at this time is always positive. Add +1 if $Q_{47} = 1$ and the augment bit t^0 ($t^0 = 1$ for unrounded arithmetic) = 0.

If the coefficient of $A_i = \pm 0$, the answer is ± 0 according to normal arithmetic rules. (+ 0 times + 0 = + 0; + 0 times - 0 = -0). In these cases, Q_f is always positive 0. No exponent or divide faults occur.

If the coefficient of the operand in memory is ± 0 , the result is the same as if A_i were equal to ± 0 .

FDV Floating Divide (FPO)
Op. Code 33

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

The magnitudes of the coefficients are divided to obtain a 36-bit coefficient in A (lower 36 bits) and a 36 bit remainder in the lower part of Q. After the iterative divide operation, two times the remainder is compared with the divisor. If two times the remainder is greater than or equal to the divisor, the resulting quotient in A is increased by + 1 (provided the augment bit $t^0 = 0$).

If the coefficient of A_i is equal to ± 0 , the answer is ± 0 according to normal arithmetic rules. In these cases, Q_f is always positive 0. No exponent or divide faults occur.

If the coefficient of M_i (divisor) is ± 0 , this constitutes a Divide Fault. No exponent faults occur. If the coefficients of both A_i and M_i are ± 0 , the answer is ± 0 according to normal arithmetic rules and no faults occur. Note that arithmetically this result is undefined.

ADDRESS ARITHMETIC

In the Address Arithmetic instructions, only the lower 15 bits of the operand or data words are used.

INA Increase A Op. Code 11

Adds Y to A. The 15-bit operand, Y, with its highest order bit (sign bit) extended, is added to A.

INI Increase Index Op. Code 51

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

ISK Index Skip Op. Code 54

Compares (B^b) with 'y'. If the two quantities are equal (positive zero \neq negative zero), B^b is cleared and the lower instruction is skipped. If the quantities are unequal, (B^b) is increased by one and the lower instruction is performed. Counting in this instruction is performed in two's complement notation. If $b = 0$, the (B^b) are taken to be zero. ISK should be restricted to an upper instruction. If used as a lower instruction, the following occurs:

- 1) If $(B^b) = y$, B^b is cleared and the next instruction is executed.
- 2) If $(B^b) \neq y$, (B^b) is increased by one and the next instruction is executed.

LOGICAL

- 1) 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:

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

2) A logical product is used, in many cases, to extract 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), the operand is compared to a mask composed of a predetermined pattern of "0's" and "1's". Forming the logical product of the operand and the mask causes the operand to retain its original contents only in those stages which have corresponding "1's" in the mask. When only the selected bits remain, the instruction proceeds to conclusion.

SST Selective Set Op. Code 40

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 storage word do not modify the corresponding bits in A. In a bit-by-bit comparison of (A) and (M), four possible combinations of bits are possible.

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 Selective Complement Op. Code 42

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 Selective Clear Op. Code 41

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), four possible combinations of bits are possible.

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

SSU Selective Substitute Op. Code 43

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 Load Logical Op. Code 44

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

ADL Add Logical Op. Code 45

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

SBL Subtract Logical Op. Code 46

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

STL Store Logical Op. Code 47

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

SHIFTING

- 1) The largest shift count for a 48-bit register is 48₁₀, for a 96-bit register, 96₁₀. If a shift greater than the largest shift count is attempted, the Shift Fault indicator will be set and the shift will not occur.
- 2) Shift instructions can vary in execution time; that is, performing a shift of 46₁₀ places, for example, takes longer than a shift of 1 place.
- 3) All left shifts are end-around, all right shifts are end-off with sign extension.

ARS A Right Shift Op. Code 01

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

QRS Q Right Shift Op. Code 02

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

LRS Long Right Shift Op. Code 03

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

ALS A Left Shift Op. Code 05

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

QLS Q Left Shift Op. Code 06

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

LLS Long Left Shift Op. Code 07

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

SCALE

- 1) Address modification does not apply. The index register is used to preserve the scale factor.
- 2) If $b = 0$, scaling is executed but the scale factor is lost.
- 3) If $b = 7$, indirect addressing is used and at least one storage reference is made.

- 4) If (A) or (AQ)_i is already scaled or equal to positive or negative zero, $k \rightarrow B^b$, and scaling is not executed.
- 5) If the lower 7 bits of the shift count are initially equal to 0, $k \rightarrow B^b$, and scaling is not executed.
- 6) The Shift Fault indicator is not affected by this instruction.

SCA Scale A Op. Code 34

Shifts A left circularly until the most significant digit is to the right of the sign bit or until the lower 7 bits of $k = 0$. (Maximum shift = 177_8 , although k may contain up to and including 77777_8 .) The shift count (lower 7 bits of k) is reduced by one after each shift. The shift operation terminates when the lower 7 bits of $k = 0$ or the most significant bit is to the right of the sign bit. Upon termination, bits 00 -- 14 of k are entered in the designated index register.

SCQ Scale AQ Op. Code 35

Shifts AQ left circularly until the most significant digit is to the right of the sign bit or until the lower 7 bits of $k = 0$. (Maximum shift = 177_8 , although k may contain up to and including 77777_8 .) The shift count (lower 7 bits of k) is reduced by one after each shift. The shift operation terminates when the lower 7 bits of $k = 0$ or the most significant bit is to the right of the sign bit. Upon termination, bits 00 -- 14 of k are entered in the designated index register.

REPLACE

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

RAD Replace Add Op. Code 70

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

RSB Replace Subtract Op. Code 71

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

RAO Replace Add One Op. Code 72

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

RSO Replace Subtract One Op. Code 73

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

STORAGE TEST

SSK Storage Skip Op. Code 36

Senses the sign bit of the operand in storage location M. If the sign is negative, the lower instruction is skipped. The contents of the operational registers are left unmodified. SSK is usually restricted to an upper instruction. If used as a lower instruction, this becomes a pass instruction.

SSH Storage Shift Op. Code 37

Senses the sign bit of the quantity in storage location M. If the sign bit is negative, the lower instruction is skipped. In either case the quantity is shifted left circular one bit. This instruction is usually restricted to the upper position. If used as a lower instruction the quantity at storage location M is shifted as above and the next instruction is executed. The contents of the operational registers are left unmodified.

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 $b = 1 - 6$, the first address searched is at $m + (B^b - 1)$.
- 4) If $(B^b) = 0$, no search is made.
- 5) The operands searched by these instructions may be in either fixed or floating point format.
- 6) Normally this is an upper instruction. If used as a lower instruction, no skip is possible.

EQS Equality Search Op. Code 64

Searches a list of operands to find one that is equal to A. The number of items to be searched is specified by B^b . These items are in sequential addresses

beginning at the location specified by 'm'. The search begins with the last address, $m + (B^b - 1)$. B^b is reduced one count for each word that is searched until an operand is found that equals A or until B^b equals zero. If the search is terminated by finding an operand that equals A, the lower instruction is skipped. The address of the operand satisfying this condition is given by the sum of 'm' and the final contents of B^b . Positive zero and minus zero are recognized as unequal quantities. When EQS is used as a lower instruction, the next instruction is executed when $(B^b) = 0$, or when the condition is met.

Note:

- 1) If $m = 00000_8$, $(B^b) = 77777_8$, 77777 quantities may be searched starting at address 77776₈.
- 2) To search location 77777₈, let $b = 0$ and $m = 77777_8$.

THS Threshold Search Op. Code 65

Searches a list of operands to find one that is greater than A. The number of items to be searched is specified by B^b . These items are located in sequential addresses beginning at the location specified by 'm'. The search begins with the last address, $m + (B^b - 1)$. The contents of the index register are 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, the lower instruction is skipped. 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, no skip is possible. If THS is used as a lower instruction, the next instruction will be executed when search terminates. In the comparison made here, positive zero is considered as greater than minus zero.

MEQ Masked Equality Search Op. Code 66

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 Masked Threshold Search Op. Code 67

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.

JUMPS AND STOPS

Normal Jump

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

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

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

A jump instruction may appear in either position in a program step. If the jump instruction appears in the first (upper) part of the program step and the jump is taken, the second (lower) part of the program step is never executed. If the instruction appears in the lower part, the upper part is executed in the normal manner.

AJP A Jump Op. Code 22

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".

For conditions specified by 'j' when its value is 4 - 7, refer to the Return Jump section.

QJP Q Jump Op. Code 23

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.

IJP Index Jump Op. Code 55

Examines (B^b). If this quantity is not zero, (zero ≠ 77777g) the quantity is reduced one count and a jump is executed to program step 'm'. The index jump can be used in the upper or lower instruction without reservation; it executes a normal jump upon satisfaction the jump condition. If b = 0, execution continues with the next program step.

SLJ Selective Jump Op. Code 75

Jumps to 'm' if the condition of the Jump switches specified by 'j' exists. If not, the next instruction is executed.

SLJ	j = 0	Jump unconditionally (does not reference Jump switch setting).
SJ1	j = 1	Jump if Jump switch 1 is set
SJ2	j = 2	Jump if Jump switch 2 is set
SJ3	j = 3	Jump if Jump switch 3 is set

SLS Selective Stop Op. Code 76

Stops at present step in the sequence if the condition of the Stop switch specified by 'j' exists. (Stop switches are located on the maintenance panel of the computer. These are only active when the computer is in maintenance mode.) If the stop condition exists, the stop is executed, and the jump is executed unconditionally when the Go switch is pressed. If the stop condition is not satisfied, the jump is executed unconditionally.

- SLS j = 0 Stop unconditionally (does not reference Stop switch setting).
- SS1 j = 1 Stop if Stop switch 1 is set
- SS2 j = 2 Stop if Stop switch 2 is set
- SS3 j = 3 Stop if Stop switch 3 is set

Return Jump

A return jump begins a new program sequence at the lower instruction portion of the program step to which the jump is made. At the same time, the execution address of the upper instruction of that program step is replaced with the address of the next program step in the main program. This instruction is usually an Unconditional Jump instruction and allows a return to the main program after completing the subprogram sequence (figure 3-1).

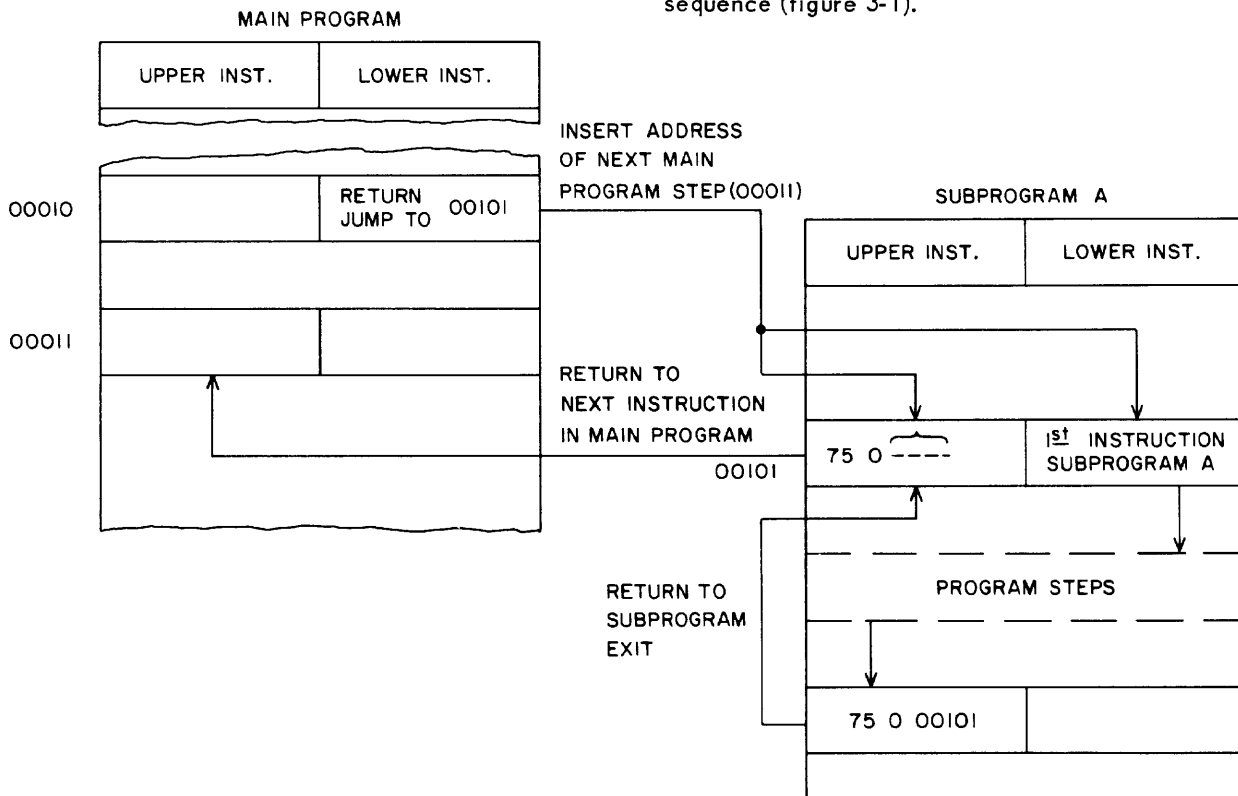


Figure 3-1. Return Jump

ARJ A Jump Op. Code 22

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) = -

For conditions specified by 'j' when its value is 0 - 3, refer to the Normal Jump section.

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

QRJ Q Jump Op. Code 23

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 Selective Jump Op. Code 75

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

- RTJ j = 4 Return jump unconditionally (does not reference Jump switches).
- RJ1 j = 5 Return jump if Jump switch 1 is set
- RJ2 j = 6 Return jump if Jump switch 2 is set
- RJ3 j = 7 Return jump if Jump switch 3 is set

Note: The Jump switch is illuminated when it is in the Set position.

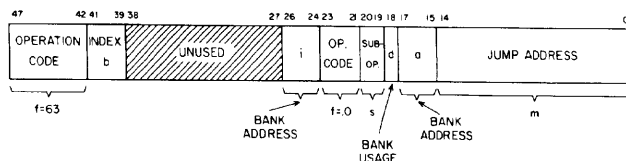
SLS Selective Stop Op. Code 76

Stops on condition 'j' and executes a return jump when the Go switch is pressed. If the stop condition is not satisfied, the stop is not executed and the return jump is executed unconditionally.

- SRJ j = 4 Stop unconditionally; return jump on Go (does not reference Stop switches).
- SR1 j = 5 Stop if Stop switch 1 is set; return jump on Go.
- SR2 j = 6 Stop if Stop switch 2 is set; return jump on Go.
- SR3 j = 7 Stop if Stop switch 3 is set; return jump on Go.

BANK JUMPS

The Bank Jump instructions (63.0, 63.1) are similar in operation to the Normal and Return Jumps. These instructions are included in the 3400 for 3600 compatibility purposes. When used in the 3400 computer, the a, i and d designators are meaningless (i.e., not interpreted). See the 3600 Reference Manual for a detailed description of the Bank Jump instructions using the a, i, and d designators.



Two jump instructions, using the format above, are designated by the suboperation designator 's'. Interpretations of 's' and the operations are outlined below.

UBJP Unconditional Jump Op. Code 63.0, s=0

Jumps unconditionally to the modified jump address $M [M = m + (B^b)]$. If $b = 0$, the jump is to 'm'. If $b = 7$, the jump address is obtained by indirect addressing.

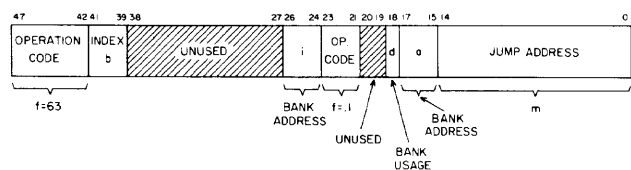
BRTJ Unconditional Return Jump Op. Code 63.0, s=1

Stores an Unconditional Jump instruction (63.0, s = 0) in the address designated by 'M' [$M = m + (B^b)$]. If $b = 0$, $M = m$; if $b = 7$, the Jump address is obtained by indirect addressing. Designator values in this stored (UBJP) instruction are selected to enable return to the next main program step. Values for these designators are:

- 1) $m = P + 1$ Enables return to the next main program step.
- 2) $b = 0$ Index designator set to zero so that $P + 1$ (in 'm' portion) is not modified when UBJP instruction is executed.

After storing this instruction, execution continues at $M + 1$ (address immediately following store address).

BJPL Unconditional Jump to Lower Op. Code 63.1



This instruction, specified by operation code 63.1, performs an unconditional jump to the lower instruction contained in the storage address designated by $M [M = m + (B^b)]$. If $b = 0$, the jump is to the lower instruction in 'm'; if $b = 7$, the jump address is obtained by indirect addressing.

CHARACTER HANDLING

These instructions transfer 6 bits of information to/ from a 6-bit portion of storage location 'M' and the lower six bits of the A register. On a load operation, the contents of 'M' remain unaltered. On a store operation, only the 6-bit portion of 'M' specified by the contents of v is altered.

LDCH Load Character Op. Code 63bv0006500m

Transmit the 6-bit portion of the contents of M specified by the contents of index register v to the lower 6 bits of A. The remainder of A is cleared. (M) is not altered.

Contents of index register v equals:

00052 ₈ = 00042 ₁₀	bits 47 - 42 of (M)
00044 ₈ = 00036 ₁₀	bits 41 - 36 of (M)
00036 ₈ = 00030 ₁₀	bits 35 - 30 of (M)
00030 ₈ = 00024 ₁₀	bits 29 - 24 of (M)
00022 ₈ = 00018 ₁₀	bits 23 - 18 of (M)
00014 ₈ = 00012 ₁₀	bits 17 - 12 of (M)
00006 ₈ = 00006 ₁₀	bits 11 - 06 of (M)
00000 ₈ = 00000 ₁₀	bits 05 - 00 of (M)

Index register b is used for normal address modification. If b = 0, m = M; if b = 7, indirect addressing applies.

If the contents of index register v are unequal to one of the eight quantities specified, or the octal content of bits 15 → 35 is not exactly as shown, the instruction is illegal. An unconditional return jump to address 00020 is executed. The initial contents of A and M are not altered.

STCH Store Character Op. Code 63bv0006505m

Store the lower 6-bit portion of the contents of A in the 6-bit portion of address M specified by the contents of index register v. A remains unchanged. The portion of (M) not affected by the byte is left in its original state.

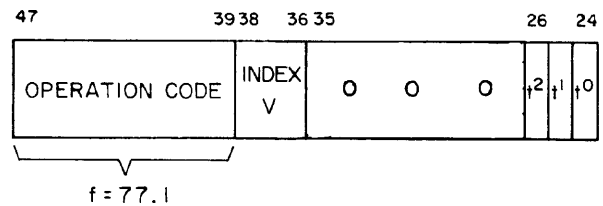
Contents of index register v equals:

00052 ₈ = 00042 ₁₀	bits 47 - 42 of M
00044 ₈ = 00036 ₁₀	bits 41 - 36 of M
00036 ₈ = 00030 ₁₀	bits 35 - 30 of M
00030 ₈ = 00024 ₁₀	bits 29 - 24 of M
00022 ₈ = 00018 ₁₀	bits 23 - 18 of M
00014 ₈ = 00012 ₁₀	bits 17 - 12 of M
00006 ₈ = 00006 ₁₀	bits 11 - 06 of M
00000 ₈ = 00000 ₁₀	bits 05 - 00 of M

Index register b is used for normal address modification. If b = 0, m = M, if b = 7, indirect addressing applies.

If the contents of index register v are unequal to one of the eight quantities specified, or the octal content of bits 15 → 35 is not exactly as shown, the instruction is illegal. An unconditional return jump to address 00020 is executed. The initial contents of A and M are not altered.

AUG Augment Op. Code 77.1



The 24-bit Augment instruction may be used to perform one or more of the following operations:

- 1) Increase the capabilities of certain instructions by specifying additional operations to be performed.
- 2) Provide additional modification of the address portion of the lower instruction.

When this instruction is used in the lower position of a program step, no meaningful operation is performed. The instruction becomes a pass instruction and the program continues at P + 1.

When this command is used in the upper position of a program step (cases 1 and 2 above) the following operations occur:

- 1) Operations using index designator 'v' are performed.
- 2) The augment operation designators 't' are stored to condition the operation of the lower instruction being augmented.

Values for 'v' are assigned as follows:

Value	Operation
v = 0	If v = 0, this designator has no significance in the operation.
v = 1-6	If v = 1-6, address modification applies. The contents of the index register specified by 'v' are added to the address portion (bits 00-14) of the lower instruction to form m', y', or k', whichever the case.
v = 7	If v = 7, indirect addressing rules apply. The quantity held in the address portion (bits 00-14) of the lower instruction is treated as a storage address (whether 'm' or 'y' or 'k'). The lower 18 bits at this storage address are read from storage and: a) The upper 3 bits (new 'v') are placed in the 'v' designator position of the augment instruction. b) The lower 15 bits are placed in the address portion of the lower instruction. If new v = 7, indirect addressing continues until completed; if new v = 1-6, address modification is performed.

Values for 't' are assigned as follows:

Designator	Value	
	If a "0"	If a "1"
t ₀	Rounded arithmetic	Unrounded arithmetic
t ₁	Normalized arithmetic	Un-normalized arithmetic***
t ₂	Use signed operand	Use magnitude of operand* (positive value)

Instructions which may be augmented are listed below. Designators which may be used when augmenting a given instruction are checked opposite that instruction.

Instruction to be Augmented	t ₂ *	t ₁	t ₀	v**
LDA	X			X
LDQ	X			X
STA	X			X
STQ	X			X
ADD	X			X
SUB	X			X
MUI	X			X
DVI	X			X
FAD	X	X	X	X
FSB	X	X	X	X
FMU	X	X	X	X
FDV	X	X	X	X

Upon completing the operations specified by the upper (Augment) instruction, the address portion of the lower instruction now contains a modified value (if 'v' specifies address modification). When the instruction being augmented (the lower instruction) is executed, its index designator is interpreted in the normal manner, i.e., M, Y, or K = [(m,y, or k) + (Bb) + (Vv)]. Indirect addressing or address modification is performed on the address modified by the Augment operation.

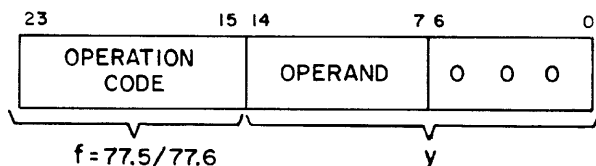
*t₂ applies to both fixed and floating point arithmetic.

** May be meaningfully used to augment most 24-bit instructions.

*** If a right shift is required to normalize, the computer normalizes regardless if the t bit = "1".

EUB Enter Upper Bound
Op. Code 77.5

ELB Enter Lower Bound
Op. Code 77.6



These 24-bit bounds instructions load 8-bit quantities into the upper and lower bounds registers. (When a bounds register contains all "1's", its content is interpreted as 77600g.)

Bounds checking on jump and memory addresses occurs only if the interrupt system is active and the bounds bit (bit 42) in the Interrupt Mask register is set. (Bit 42 is set with an Internal Function instruction.) Setting the Mask bit or activating interrupt may occur in any order. If the Mask bit is set and interrupt is active, any one of the following operations will cause a Category I interrupt:

- 1) A jump to an address outside the bounds defined by the upper and lower bound addresses. (If a jump is attempted out of bounds, it will not be executed and interrupt will occur. Upon return to the main program after processing the interrupt, the next instruction will be executed.)

- 2) Writing in an address out of bounds.
- 3) Reading an instruction from out of bounds.

The only permissible use of an address out of bounds (when bounds checking is in effect) is to read its content as an operand. (An 18-bit indirect address is considered an operand and can be read from out of bounds.)

Addresses which may be referenced are such that $B_L \leq S$ (upper 8 bits) $< B_U$ (where S is the storage address). Only bits 7 → 14 of S are used in the comparison for bounds checking. When the upper bounds register is set to + 0 (every bit = "0") the upper bound is disregarded.

Note that:

- 1) There is no MC for the bounds registers. They are program cleared when both registers are set to all 0's by 77.5 and 77.6 instructions.
- 2) If interrupt is active and bit 42 in the Interrupt Mask register is set (Out of Bounds Interrupt selected), then instructions 77.5 and 77.6 are illegal and cause a category III interrupt. This prevents changing the bounds when checking is in effect.
- 3) Bounds checking does not occur when the Category I, II or III Interrupt Mode FF is set. This permits the interrupt routines to be out of bounds.

CHAPTER IV

INTERRUPT SYSTEM

The interrupt system provides for testing whether or not certain conditions (internal or external) exist without having these tests in the main program. (See the three categories of interrupts mentioned later in this chapter.) After executing each main program instruction, these conditions are sampled. If one of the conditions exists, execution of the main program halts. The contents of the Program Address register, P, are stored and an interrupt routine is initiated. This interrupt routine takes the necessary action for the condition and then jumps back to the main program (see programming cautions at the end of this chapter).

For most conditions that can cause an interrupt, the program has two alternatives. It may select an interruptible condition, such that interrupt occurs when that condition occurs or it may choose to have the interrupt system ignore the condition. This is accomplished by not selecting interrupt on the condition. The program also has the choice of whether or not the interrupt system is to be used. The Internal Function instruction activates or deactivates the interrupt system and selects the desired interrupts.

LOGICAL DESCRIPTION OF INTERRUPT SYSTEM

Three registers are directly involved in the interrupt system:

- 1) an Interrupt register
- 2) an Interrupt Mask register
- 3) a Product register

The bits in these registers are numbered from right to left in ascending order. The rightmost bit is numbered zero, and the leftmost bit (in a 48-bit register) is numbered 47.

Each of the bits of the register is associated with a particular internal or external condition. This bit association is identical in all three registers. See the Internal Function instruction for codes that set and clear bits in the Interrupt and Interrupt Mask registers. Bits are assigned as shown in table 4-1.

Brief statements concerning the registers are given below:

Interrupt Register

Each of the internal conditions which can cause an interrupt is wired to a particular bit position of this register. Each bit position associated with external conditions receives an interrupt line from one of the four possible data channels.

Interrupt Mask Register

The programmer selects to have a given interrupt condition (internal/external) tested by setting the appropriate bit of the Interrupt Mask register.

Product Register

This register is the result of the AND of corresponding bits in the Interrupt and Interrupt Mask registers. Bits in the Product register are therefore the logical product of the Interrupt and Interrupt Mask registers. The Product register is fictitious in that it is not composed of flip-flops. Hence there are no instructions for clearing bits in this register. The "bits" in the Product register are "cleared" when one or both corresponding bits in the Interrupt and Interrupt Mask register = 0.

An interrupt results from a "1" in a bit of the Product register. To set to "1" any bit in the Product register requires that the Mask register bit must have been set to "1".

CATEGORIES OF INTERRUPTS

There are three categories of interrupts in this system. Those interrupts which set bits in the Interrupt Register fall into Category I (see table 4-1). These include all internal and external interrupts. Categories II and III interrupts may be considered traps. On all occasions, when a Category II or III interrupt occurs, it will be processed immediately. Category II is for floating point instructions when the floating point option is not part of the computer. Category III is for illegal in-

structions. These include instructions with the following operation codes: 00*, 26, 27, 62, 63*, 74, 0* → 74.6*, 74.7, 77.0, 77.7, 77.5 and 77.6 when bounds checking is in effect. (The asterisk denotes that the reader should refer to the specific instruction description in chapter 3.)

Category I

This category includes all I/O interrupts and interrupts on internal conditions such as overflow and divide fault.

A Category I interrupt causes a return jump to P = 00007 provided: a) the proper bit in the Interrupt Mask register is set; and b) interrupt is active; and c) the Category I Interrupt Mode is clear. This is accomplished as follows:

- 1) Stops execution of the "main" program at the end of the instruction currently being executed or at the completion of the current iteration of a Search instruction.
- 2) Sets the Category I Interrupt, (this locks out all other Category I Interrupts).
- 3) Sets the Category I Interrupt Mode.
- 4) Performs a return jump to address 00007.
 - a) If the next instruction would have been a lower instruction, sets the Category I Interrupt Exit to lower. (P) are placed in the upper address portion of location 00007.
 - b) If the next instruction would have been an upper instruction, the Category I Interrupt Exit is set to upper. (P) + 1 are stored in the upper address portion of location 00007.
- 5) Clears the Category I Interrupt.

The next time P is equal to fixed address 00007, the Category I Interrupt Exit is examined, and if it is set to lower, the Lower Instruction at the Interrupt Exit Address is executed. If the Category I Interrupt Exit is set to upper, the Upper instruction at the Interrupt Exit Address is executed. The Category I Interrupt mode is cleared.

Exception to Category I

If the Category I interrupt was caused by a bounds fault, the instruction step which caused the bounds fault will be repeated.

Table 4-1, Assignment of Bits

Bit	Assignment
01	Shift fault
02	Divide fault
03	Arithmetic overflow
04	Exponent overflow
05	Exponent underflow
10	Interrupt on Channel 0 becoming inactive*
11	Interrupt on Channel 1 becoming inactive
12	Interrupt on Channel 2 becoming inactive
13	Interrupt on Channel 3 becoming inactive
20	Channel 0 I/O Transmission parity error
21	Channel 1 I/O Transmission parity error
22	Channel 2 I/O Transmission parity error
23	Channel 3 I/O Transmission parity error
30	Channel 0 External interrupt
31	Channel 1 External interrupt
32	Channel 2 External interrupt
33	Channel 3 External interrupt
42	Out of Bounds
43	Manual interrupt
44	Time interrupt
45	Operand parity error
46	Instruction parity error

* The channel will go inactive before any interrupts from external equipments occur.

Category II

This category of interrupt occurs just before the computer attempts to execute one of the floating point instructions which are not available in the basic machine (30,31,32,33). A return jump is made to the fixed address specified by the operation

code (i.e., if the operation is FSB - op. code 31 - the return jump is to fixed address 00031. Addition of the floating point option (FPO) eliminates this interrupt category.

A Category II interrupt causes the following operations to occur:

- 1) a. If the floating point instruction was a lower instruction, the Category II Interrupt Exit is set to lower. (P) are placed in the upper address portion of the storage location at step 3.
- b. If the floating point instruction was an upper instruction, the Category II Interrupt Exit is set to upper. (P) are placed in the upper address portion of the storage location at step 3.
- 2) Sets Category II Interrupt Mode.
- 3) Performs a return jump to the fixed address specified by the operation code.

The next time P is equal to the fixed address specified above (00030-00033), the Category II Interrupt Exit is examined, and if it is set to lower, the lower instruction at the Interrupt Exit Address is executed. If the Category II Interrupt Exit is set to upper, the upper instruction at the Interrupt Exit Address is executed. The Category II Interrupt Mode is cleared.

Category III

This category of interrupt occurs when an attempt is made to execute one of the illegal instructions (00*,26,27,62,74.0* through 74.6*,74.7,77.0*,77.5*,77.6*,77.7). A return jump is made to fixed address 00020. This is accomplished as follows:

- 1) a. If the illegal instruction was a lower instruction, the Category III Interrupt Exit is set to lower. (P) are stored in the upper address portion of location 00020 at step 3.
- b. If the illegal instruction was an upper instruction, the Category III Interrupt Exit is set to upper. (P) are stored in the upper address portion of location 00020 at step 3.
- 2) Sets Category III Interrupt Mode.
- 3) Performs a return jump to fixed address 00020.

* See instruction description in chapter 3.

The next time P is equal to 00020, the Category III Interrupt Exit is examined, and if it is set to lower, the lower instruction at the Interrupt Exit Address is executed. If the Category III Interrupt Exit is set to upper, the upper instruction at the Interrupt Exit Address is executed. The Category III Interrupt Mode is cleared.

Programming Cautions

- 1.) a) Category I interrupt is the only one with hardware lockout.
- b) Category I interrupts require that the interrupt system be active.
- c) Category II and III interrupts occur immediately and unconditionally, i.e., Category II and III interrupts do not require the interrupt system to be active.
- 2.) The program MUST preset the interrupt exits to an unconditional jump (the upper instruction of addresses 00007, 00020, 00030-00033).
- 3.) Category II and III interrupts block Category I; i.e., a Category I interrupt waits until II or III is ready to execute the lower instruction in the entrance to the interrupt routine and then interrupts.
- 4.) All Category I interrupts (except a Category I interrupt caused by a bounds fault) return (after interrupt processing) and continue with the next unexecuted program step.
- 5.) Interrupts can occur only before or after the complete execution of an instruction with three exceptions:
 - a) An indirect address loop
In this case interrupt may occur after executing one or more indirect address loops and before executing another indirect address loop or the instruction proper.
 - b) An augment indirect address loop
Here the same rules apply as in 5a. Also see 8.
 - c) A search loop
If a Category I interrupt occurs during a search instruction (64-67) and interrupt is active, selected, etc., the interrupt routine will be entered at the end of the current search iter-

ation. The search instruction will not go to completion. Note that the hardware returns program control to finish a search or to begin again on an indirect address loop.

Note that if a search is being augmented, the v and b designators of the augment and search instructions respectively cannot specify the same index register. If v=b, the search cannot recover if interrupt occurs during a search iteration; i.e., some locations will be skipped (not searched) when the search continues after interrupt.

6.) If a Category I interrupt arises while another Category I interrupt is being processed, and the conditions for interrupting are all present, interrupt will immediately recur after the first interrupt is processed. The computer will not continue with the main program until all interrupts have been processed. Priority for processing "stacked" interrupts is determined by the programmer.

7.) A Category I interrupt caused by a bounds fault, and all Category II and III interrupts all return and re-execute the instruction step which caused the interrupt (after interrupt processing is complete), e.g., if a Category II interrupt is caused by attempting to execute a lower 30 instruction, the computer will continue main program instruction execution (after interrupt processing) at the same program step where the 30 instruction was located when interrupt occurred.

8.) If a Category I interrupt occurs between an Augment instruction and the instruction being augmented, the function the Augment instruction performs is lost. The hardware always returns and re-executes the Augment (after interrupt processing.)

9.) Care must be exercised in using the Internal Function instruction to set the Interrupt Exits to upper/lower. An error here could cause the computer to attempt an execution of the lower half of a 48-bit instruction or skip the augment portion of an instruction pair.

INTERRUPT EXAMPLE 1

In example 1, a Category I interrupt can occur at either point A, B, or C (figure 4-1). Figure 4-1 also shows the steps that occur if the interrupt occurs at point C. Table 4-2 gives the order of events for in-

terrupt occurring at point A, B, or C. It is assumed in this example that the upper instruction is not an Augment and interrupt is not caused by a bounds fault.

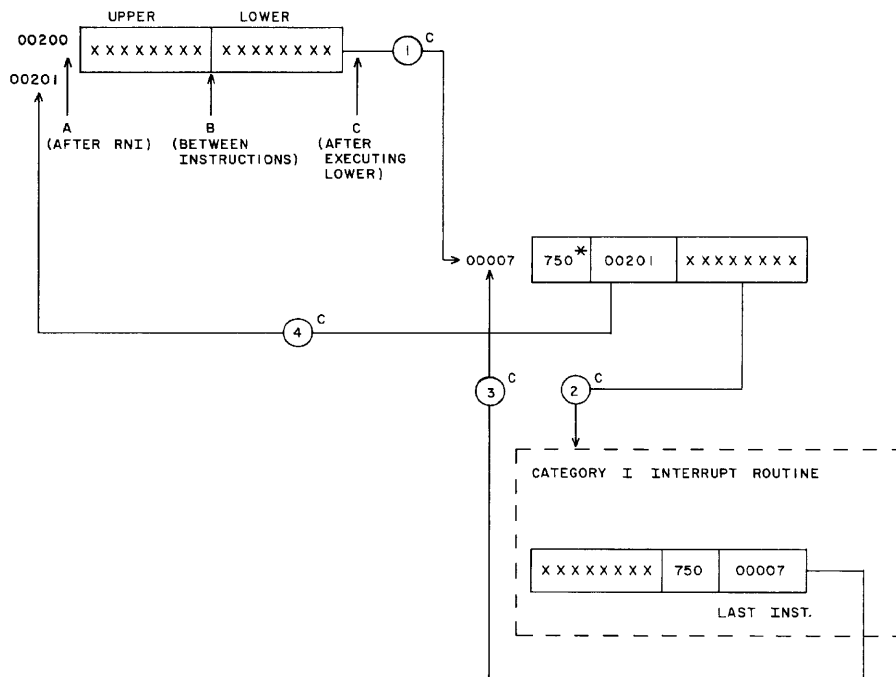


Figure 4-1. Interrupt Example 1

* The Octal Valve 750 (unconditional jump) is preset at the beginning of the main program.

TABLE 4-2. INTERRUPT EXAMPLE 1

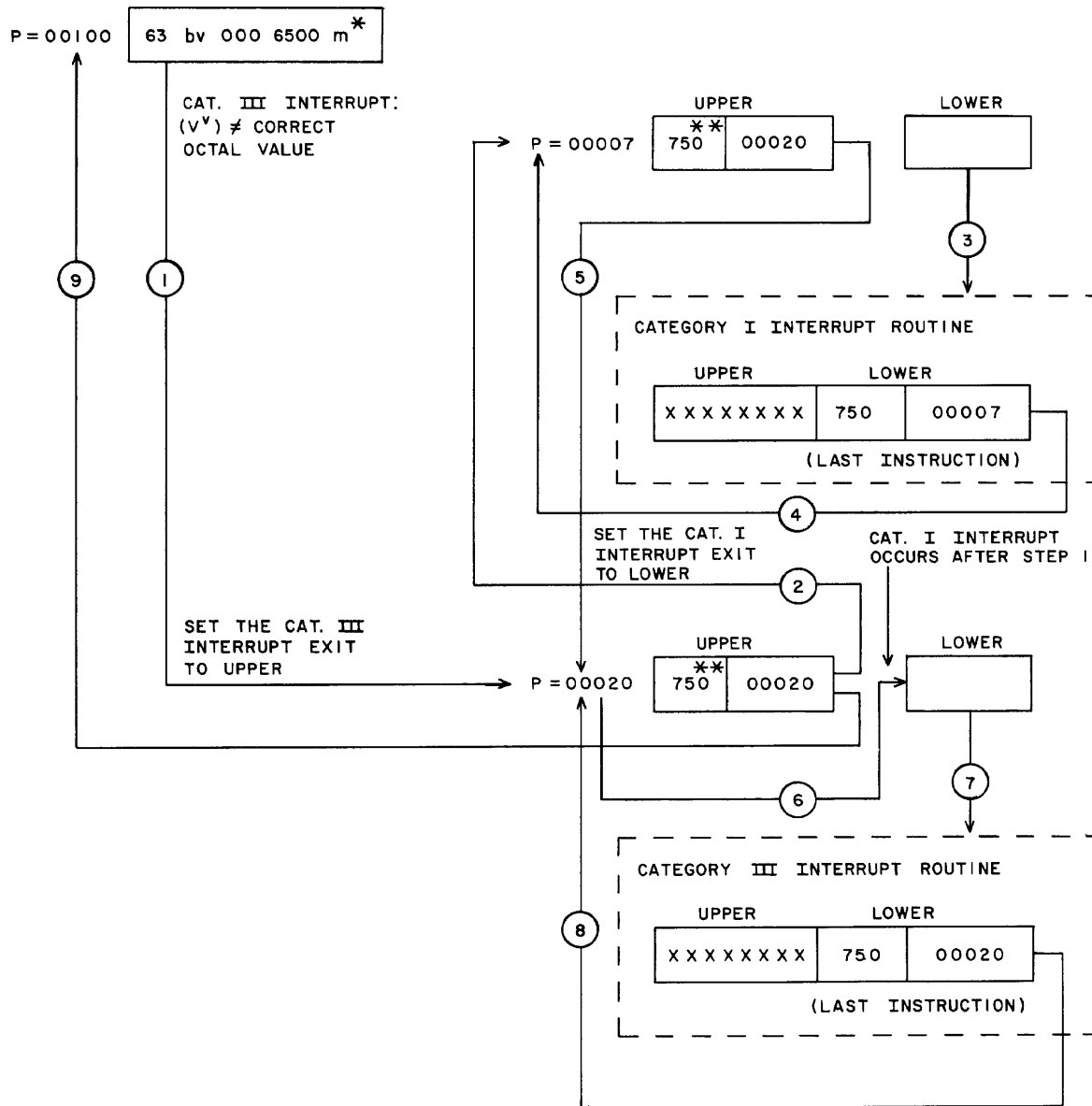
STEP	EVENT	A	B	C
1	a) Next instruction is:	upper	lower	upper
	b) Interrupt Exit is set to:	upper	lower	upper
	c) A Return Jump is made to address 00007. The quantity stored in the upper address portion of location 00007 is:	00200	00200	00201
2	a) Execute the lower instruction at address 00007. (This is entrance to the interrupt routine)	X	X	X
	b) Execute Category I interrupt routine	X	X	X
3	The last instruction performs a jump to address 00007.	X	X	X
4	Address in main program to which jump at 00007 is made:	00200	00200	00201
5	Examine Interrupt Exit. Main program continues with execution of the upper/lower instruction at the address given in 4.	upper	lower	upper

INTERRUPT EXAMPLE 2 (MULTIPLE INTERRUPTS)

See figure 4-2. In this example, a Category III interrupt occurs when the computer attempts to execute the 63 (Load Character) instruction and the contents of index register $v \neq$ one of the eight specified octal values. A Category I interrupt also occurs at the time shown. The computer handles this as follows:

- 1.) a) Set the Category III Interrupt Exit to upper (since the illegal instruction is an upper).
b) Perform a return jump to address 00020 (store 00100 in the upper address portion of location 00020).
- 2.) A Category I interrupt occurs before the lower instruction at address 00020 can be executed.
a) The Category I Interrupt Exit is set to lower (since the next instruction is a lower).
b) A return jump is performed to location 00007 (store 00020 in the upper address portion of location 00007).
- 3.) The Category I interrupt routine is executed.
- 4.) At the end of the Category I interrupt routine, a jump is made back to location 00007.
- 5.) A jump is made to location 00020.
- 6.) Since the Category I Interrupt Exit is set to lower, the lower instruction at address 00020 is executed. This is the entrance to the Category III interrupt routine.
- 7.) The Category III interrupt routine is performed.
- 8.) A jump is executed at the end of the Category III interrupt routine to location 00020.
- 9.) A jump is executed to location 00100 of the main program.
- 10.) Since the Category III Interrupt Exit is set to upper, the main program continues, starting with the execution of the upper instruction at location 00100.

NOTE: It is assumed that the Category III interrupt routine has changed the upper bits at location 00100 to a legal instruction. If not, the computer will loop indefinitely.



* Original contents are shown. The contents of main program location 00100 will be changed during execution of the category III interrupt routine.

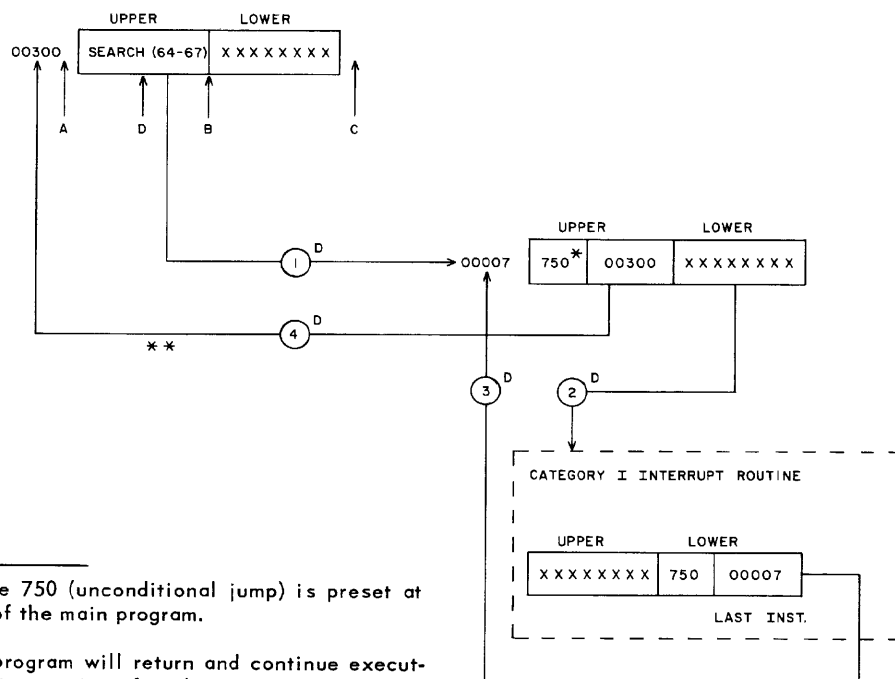
** The octal value 750 (unconditional jump) is preset at beginning of the main program.

Figure 4-2. Interrupt Example 2

INTERRUPT EXAMPLE 3

In figure 4-3, a Category I interrupt can occur at either point A,B,C, or D (the upper instruction is a normal 24-bit instruction). Figure 4-3 also shows the steps taken if the interrupt occurs at point D

(during the Search instruction). Table 4-3 gives the order of events for interrupt occurring at point A, B,C, or D.



* The octal value 750 (unconditional jump) is preset at the beginning of the main program.

** Note that the program will return and continue executing the Search instruction after the category I interrupt has been processed.

Figure 4-3. Interrupt Example 3

TABLE 4-3. INTERRUPT EXAMPLE 3

STEP	EVENT	A	B	C	D
1	a) Next instruction is:	upper	lower	upper	upper
	b) Interrupt Exit is set to:	upper	lower	upper	upper
	c) A Return Jump is made to address 00007. The quantity entered in the upper address portion of location 00007 is:	00300	00300	00301	00300
2	a) Execute the lower instruction at address 00007. (This is entrance to the interrupt routine)	X	X	X	X
	b) Execute Category I interrupt routine.	X	X	X	X
3	The last instruction performs a jump to address 00007.	X	X	X	X
4	Address in main program to which jump at 00007 is made:	00300	00300	00301	00300
5	Examine interrupt Exit. Main program continues with execution of the upper/lower instruction at the address given in 4.	upper	lower	upper	upper

INTERRUPT EXAMPLE 4

In example 4, a Category I interrupt can occur at either point A (after RNI), B (between instructions) or C (after lower instruction execution). Figure 4-4 also shows the steps taken if the interrupt occurs

at point B. Table 4-4 gives the order of events for interrupt occurring at point A, B, or C. If it is assumed in this example that the lower 24-bit instruction is not a search.

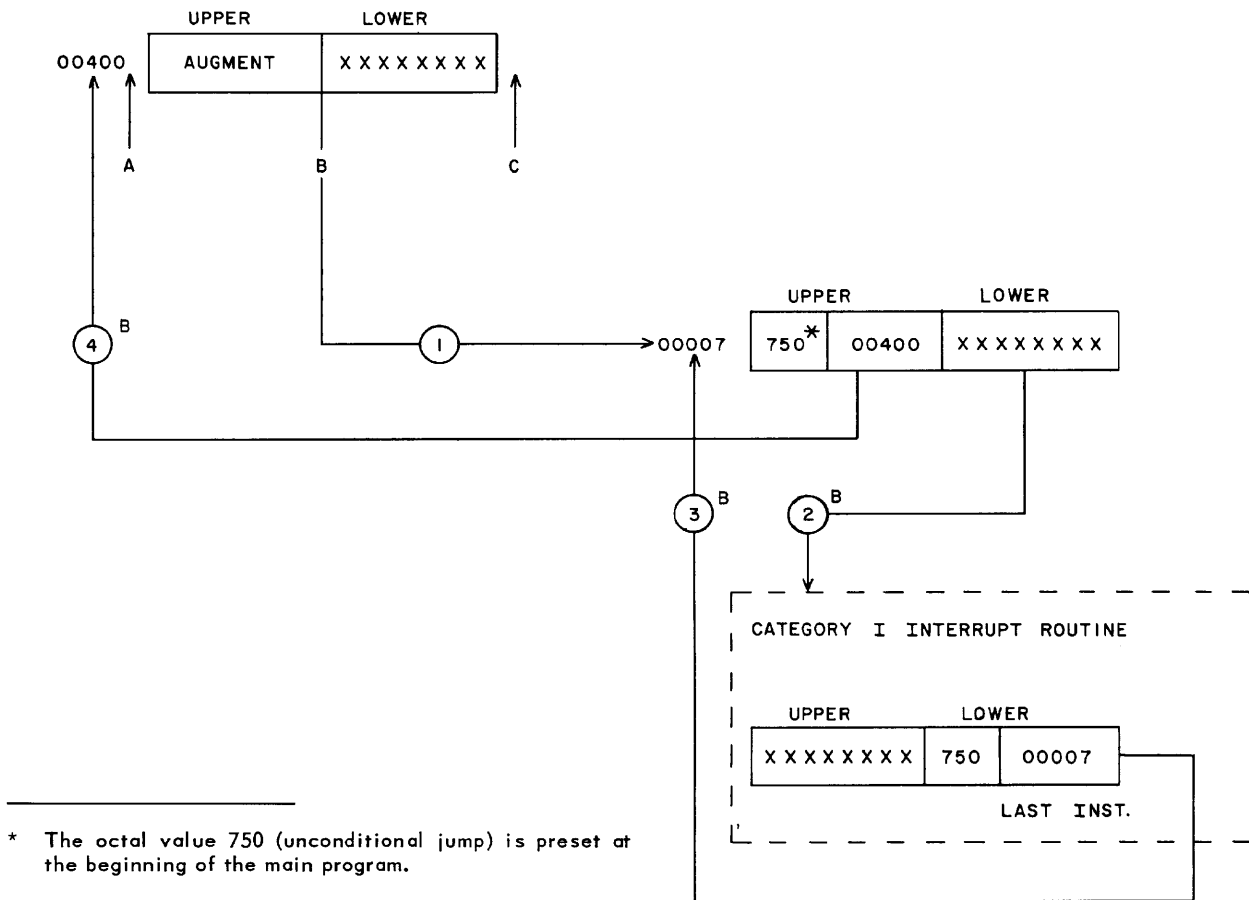


Figure 4-4. Interrupt Example 4

TABLE 4-4. INTERRUPT EXAMPLE 4

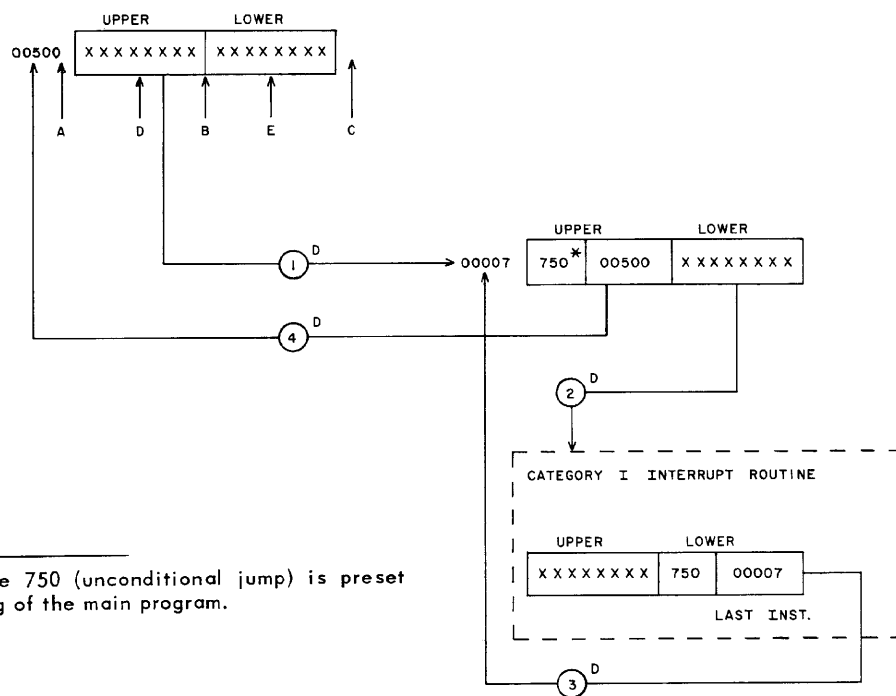
STEP	EVENT	A	B	C
1	a)	upper	lower	upper
	b)	upper	upper**	upper
	c)	00400	00400	00401
2	a)	X	X	X
	b)	X	X	X
3	The last instruction performs a jump to address 00007.	X	X	X
4	Address in main program to which jump to 00007 is made:	00400	00400	00401
5	Examine Interrupt Exit. Main program continues with execution of the upper/lower instruction at the address given in 4.	upper	upper	upper

INTERRUPT EXAMPLE 5

In example 5, it is assumed that the instructions are two normal 24-bit instructions and that each uses one or more indirect address loops (b = 7) in address modification. A Category I interrupt can occur at either point A (after RNI), B (between instructions), C (after lower instruction execution), D (after one or more indirect address loops but before instruction execution), or E (after one or

more indirect address loops but before instruction execution). Figure 4-5 also shows the steps taken if interrupt occurs at point D. Table 4-5 shows the order of events for interrupt occurring at point D or E. For interrupt occurring at point A, B, or C, the sequence of events is identical to interrupt example 1.

** Normally if the next instruction is a lower, the Interrupt Exit is set to lower. In this case it is set to upper since the Augment instruction must be re-executed after interrupt. If the Augment instruction is not re-executed, the lower instruction will not be augmented.



* The octal value 750 (unconditional jump) is preset at the beginning of the main program.

Figure 4-5. Interrupt Example 5

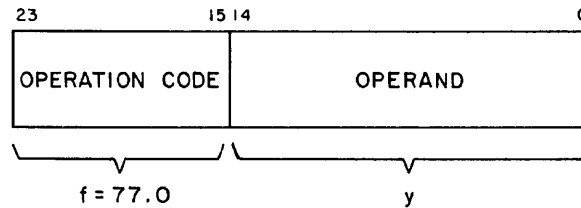
TABLE 4-5. INTERRUPT EXAMPLE 5

STEP	EVENT	D	E
1	a) Next instruction is:	same (upper)	same (lower)
	b) Interrupt Exit is set to:	upper	lower
	c) A Return Jump is made to address 00007. The quantity stored in the upper address portion of location 00007 is:	00500	00500
2	a) Execute the lower instruction at address 00007. (This is the entrance to the interrupt routine).	X	X
	b) Execute the Category I Interrupt routine.	X	X
3	The last instruction performs a jump to address 00007.	X	X
4	Address in main program to which jump at 00007 is made:	00500	00500
5	Examine Interrupt Exit. Main program continues with execution of the upper/lower instruction at the address given in 4.	upper	lower

INF INTERNAL FUNCTION Op. Code 77.0

The Internal Function instruction establishes the internal operating mode or executes the operation specified by the function code 'y'. These codes

(values) for 'y' and their designated functions, are tabulated below.



CODE	FUNCTION	COMMENTS
00000 00010 00020 00030	<u>Select Interrupt On Channel Becoming Inactive:</u> Channel 0 Channel 1 Channel 2 Channel 3	Set the designated bit in the Interrupt Mask register. Clear the designated Interrupt register bit. Interrupt occurs when the desired channel becomes inactive (i.e., when the transition from active to inactive occurs) and the corresponding bit in the Interrupt Mask register is set.
00001 00011 00021 00031	<u>Clear Select of Interrupt on Channel Becoming Inactive:</u> Channel 0 Channel 1 Channel 2 Channel 3	Clear the designated bit in the Interrupt Mask register. Clear the designated Interrupt Register bit.
00110 00120 00130	<u>Set Interrupt Exit to Lower</u> Category I Category II Category III	Sets the Interrupt Exit, enabling a return to a lower instruction of the main program step after interrupt processing is complete. (See discussion of categories I, II and III).
00111 00121 00131	<u>Set Interrupt Exit to Upper</u> Category I Category II Category III	Sets the Interrupt Exit, enabling a return to an upper instruction of the main program step after interrupt processing is complete (See discussions of categories I, II, and III.)

CODE	FUNCTION	COMMENTS
00200 00210 00220 00230	<u>Enable Interrupt From:</u> Channel 0 Channel 1 Channel 2 Channel 3	Set/Clear the designated bit in the Interrupt Mask register.
00201 00211 00221 00231	<u>Disable Interrupt From:</u> Channel 0 Channel 1 Channel 2 Channel 3	
00300 00310 00320 00330 00301 00311 00321 00331	<u>Select I/O Parity Error Interrupt:</u> Channel 0 Channel 1 Channel 2 Channel 3 <u>Clear Select of I/O Parity Error Interrupt</u> Channel 0 Channel 1 Channel 2 Channel 3	Set/Clear the designated bit in the Interrupt Mask register.
00400	Clear Arithmetic Fault	Clears any arithmetic fault condition(s) existing in the Interrupt register. (Shift fault, Divide fault, Exponent Overflow fault, Exponent Underflow fault, Arithmetic Overflow fault.)
00410 00411 00420 00421 00430 00431 00440 00441 00450 00451	Select Shift Fault Interrupt Clear Select of Shift Fault Interrupt Select Divide Fault Interrupt Clear Select of Divide Fault Interrupt Select Arithmetic Overflow Interrupt Clear Select of Arithmetic Overflow Interrupt Select Exponent Overflow Interrupt Clear Select of Exponent Overflow Interrupt Select Exponent Underflow Interrupt Clear Select of Exponent Underflow Interrupt	Set/Clear the designated bit in the Interrupt Mask register.
00412 00422 00432 00442 00452	Clear Shift Fault Clear Divide Fault Clear Arithmetic Overflow Fault Clear Exponent Overflow Fault Clear Exponent Underflow Fault	Clear the designated bit in the Interrupt register.

CODE	FUNCTION	COMMENTS
00500 00501	Select Time Interrupt Clear Select of Time Interrupt	Set/Clear the designated bit in the Interrupt Mask register. (The 00501 code also clears the bit in the Interrupt register.) Interrupt will occur only once per selection at a time interval corresponding to the power line frequency. For 60 cycle, interrupt will occur every 16-2/3 ms; for 50 cycle, interrupt will occur every 20 ms.
00510	Select Manual Interrupt	Sets the designated bit in the Interrupt Mask register. Manual interrupt will occur when this interrupt condition is selected and the Manual Interrupt switch is momentarily depressed. Interrupt will not occur if the Manual Interrupt switch is momentarily depressed and Manual Interrupt is selected some time later.
00511	Clear Manual Interrupt	Clears the designated bit in the Interrupt Mask and Interrupt registers.
00520 00521	Select Operand Parity Error Interrupt Clear Select of Operand Parity Error Interrupt	Set/Clear the designated bit in the Interrupt Mask register
00530 00531	Select Instruction Parity Error Interrupt Clear Select of Instruction Parity Error Interrupt	Set/Clear the designated bit in the Interrupt Mask register.
00522 00532	Clear Operand Parity Error Clear Instruction Parity Error	Clears the designated bit in the Interrupt register.
00540 00541	Select Out of Bounds Interrupt Clear Select of Out of Bounds Interrupt	Set/Clear the designated bit in the Interrupt Mask register. (Setting or Clearing this bit in the Interrupt Mask register always clears the corresponding bit in the Interrupt register.)
00600	Set Interrupt Active	Sets Interrupt Active which enables the Interrupt system. Interrupts will now occur if a particular bit in the interrupt Mask is set when the interrupt condition occurs.
00601	Clear Interrupt Active	Clears Interrupt Active. Regardless of the condition of bits in the Mask register, entrance into the interrupt routine does not occur when the interrupt system is inactive.
00602	Clear Interrupt System	Clears all bits in the Interrupt and Interrupt Mask registers (with the exception of I/O interrupt bits 20-23, 30-33. These are cleared with an External MC or a Function (EXTF) instruction). Also clears Interrupt Active.

CODE	FUNCTION	COMMENTS
00610	Set Internal Function and 74.0 through 74.6 I/O Lockout	<p>This code establishes a mode whereby Internal Function and 74.0 through 74.6 I/O instructions are illegal under certain conditions.</p> <p>1) When: a) Internal Function and 74.0 through 74.6 I/O Lockout is set, b) the interrupt system is active, c) the computer is not in Category I Interrupt Mode; all Internal Function and 74.0 → 74.6 instructions are illegal and will cause a Category III interrupt.</p> <p>2) If the interrupt system is not active, these instructions are executed normally, regardless of the state of the Lockout.</p>
00611	Clear Internal Function and 74.0 through 74.6 I/O Lockout	This clears the condition established by the 00610 code.

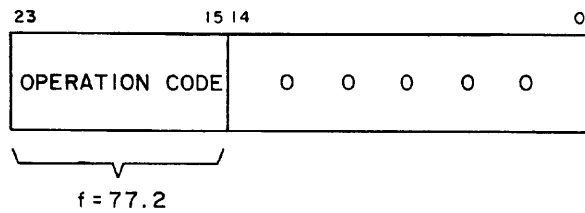
SEN Internal Sense Op. Code 77.3

This instruction senses internal computer conditions specified by 'y'. These conditions do not include shift fault, arithmetic fault, etc. If this instruction is used in bits 00 - 23, it becomes a pass

code. The program continues at P + 1. The code (values) for 'y' and their designated functions are tabulated below.

CODE	OPERATION	
00000 00001 00010 00011 00020 00021 00030 00031	Skip lower instruction if:	Channel 0 is inactive Channel 0 is active Channel 1 is inactive Channel 1 is active Channel 2 is inactive Channel 2 is active Channel 3 is inactive Channel 3 is active
00110 00111 00120 00121 00130 00131	Skip lower instruction if:	Category I Interrupt Exit is set to lower. Category I Interrupt Exit is set to upper. Category II Interrupt Exit is set to lower. Category II Interrupt Exit is set to upper. Category III Interrupt Exit is set to lower. Category III Interrupt Exit is set to upper.
00210 00211 00220 00221 00230 00231	Skip lower instruction if:	Category I Interrupt Mode is set Category I Interrupt Mode is clear Category II Interrupt Mode is set Category II Interrupt Mode is clear Category III Interrupt Mode is set. Category III Interrupt Mode is clear.

CIS Copy Interrupt Status
Op. Code 77.200000



This 24-bit instruction transmits the contents of the Interrupt register to the A register. Unused bits in A are cleared.

CMS Copy Mask Status
Op. Code 77.200001

This 24-bit instruction has the same format as the CIS instruction with the exception that the lower 5 octal digits must equal 00001. This instruction transmits the contents of the Interrupt Mask register to the A register. Unused bits in A are cleared.

CPR Copy Product Register
Op. Code 77.4

This 24-bit instruction has the same format as the Copy Interrupt Status instruction with the exception that $f = 77.4$. This instruction transmits the contents of the Product register to the A register. Unused bits in A are cleared.

CHAPTER V

INPUT OUTPUT

Input/output facility for the 3400 system is provided by the Input/Output channel. It provides the method for bidirectional data exchange and for proper control of information transmission between the system and its various 3600-type external equipments.

A simplified block diagram of a system is presented in figure 5-1. For purposes of illustration, the system in figure 5-1 shows only one data channel. A basic system includes one I/O channel. Additional I/O channels, up to a total of four, may be added.

An I/O channel may control a maximum of eight external equipments. Typical external devices are line printers, punched card equipment, and magnetic tape equipment. The Connect instruction selects the equipments individually to communicate with the system via the I/O channel.

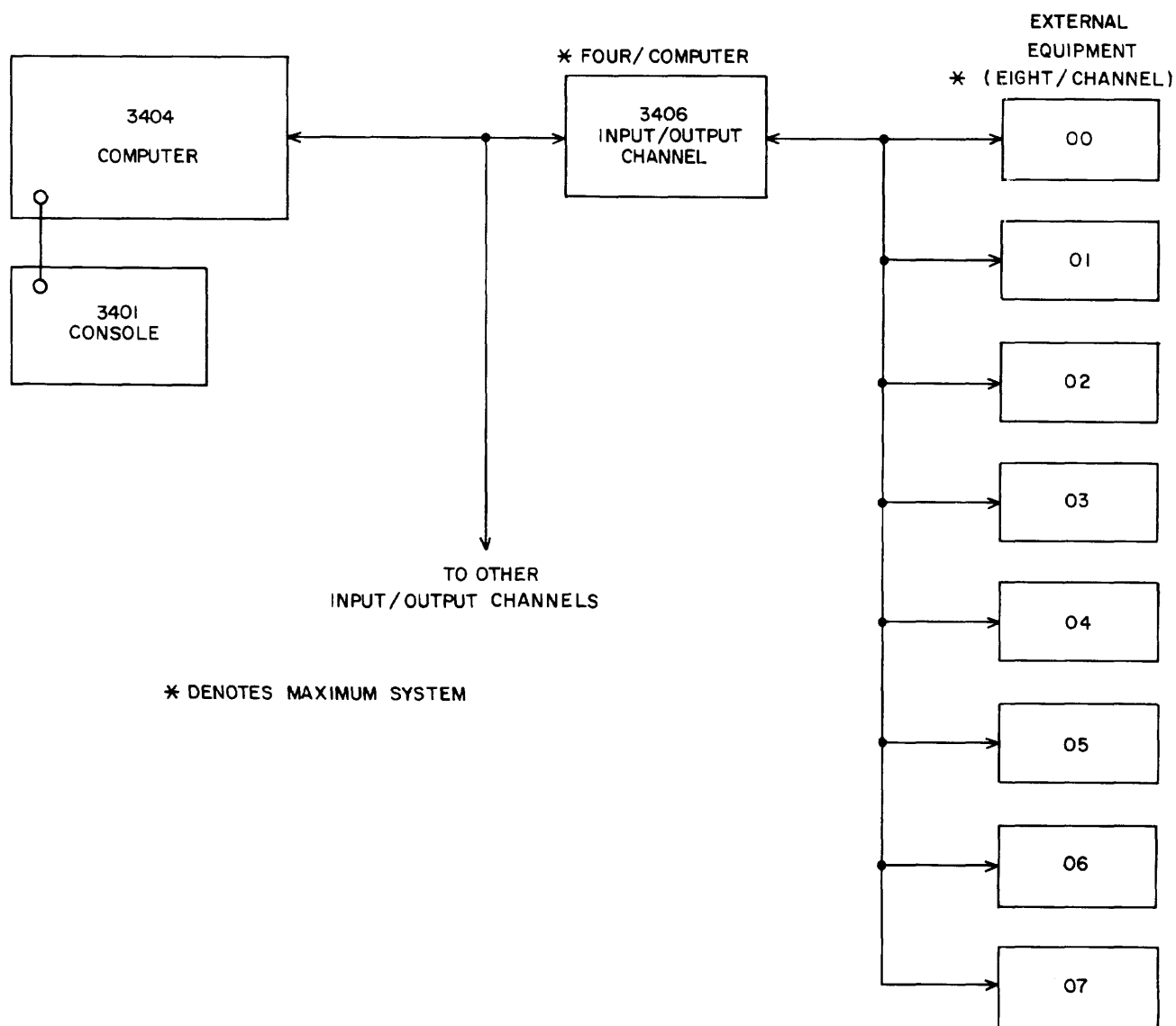


Figure 5-1. 3400 System

INPUT/OUTPUT INSTRUCTIONS

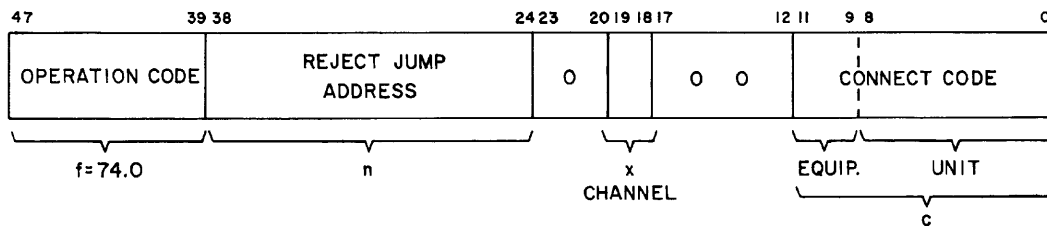
Seven instructions govern input/output operations in the system. These instructions establish operating modes within external equipments and provide for transferring data between the storage and communication modules. Provision is made for sampling the status of operating conditions in the external equipments, and for monitoring the progress of data transmissions.

Note that 74.0 through 74.6 I/O instructions are illegal when the Internal Function and 74.0 through 74.6 I/O Lockout is set.

The Connect instruction must be used to connect the external equipment to the system before data trans-

mission can be initiated. If various operating conditions within the connected equipment are to be specified, a Function instruction is executed. After initial operating conditions have been established, the Read or Write instruction may be executed to transmit data into or out of the system automatically and independent of the main program. In addition, the Read or Write instruction specifies the address of the control word which contains all other information necessary to perform the operation; i.e., starting address, and word count. While input/output operations are in progress, the status of data transmission or operating conditions within the external equipments may be sampled by using the Copy Status instruction. The Clear Channel instruction is used to clear the designated channel at the start of a program or in the event of a program failure.

CONN Connect Op. Code 74.0



An external equipment is connected to the system via an I/O channel. The digit 'x' specifies one of the four possible data channels attached to the computer.

Up to eight equipments may be attached to each I/O channel. The desired equipment is specified by the upper octal digit of the Connect code 'c'. The equipments are numbered 0 through 7 (octal). Each equipment, via an eight-position switch located on the equipment, may assume any one of the eight possible equipments codes.* Certain equipments, such as magnetic tape controllers, control more than one unit. A number specifies each unit. Unit numbers are the three lower octal digits of 'c' and may range from 000g to 777g. Normally, the legitimate range of unit numbers is from 000g to 017g.

The Connect instruction connects an equipment and/or unit to the computing system by specifying:

- 1) The 2-bit channel code (one of four channels)

- 2) The 3-bit equipment code (one of eight equipments assigned to the selected channel)
- 3) The 9-bit unit code, if any (one of 512 possible units; 16 possible units in all ordinary systems)

When the Connect instruction has been executed:

- 1) The Function instruction may be used to establish various operating modes within the equipment, if desired, and
- 2) The Read or Write instruction may be executed to transmit data from or to the designated equipment.

The external equipment remains connected until another Connect instruction is executed for the same channel, or the Clear Channel instruction is executed. Only one equipment may be connected to a channel at any one time. All four channels may have an equipment connected, and all may be active at once. Thus,

* If the operator has selected the same octal equipment number for more than one equipment on a particular channel, a Connect instruction will connect each equipment. Subsequent Read or Write instructions will reference each connected equipment resulting in loss of data or in a program malfunction.

four equipments may be simultaneously communicating with the system.

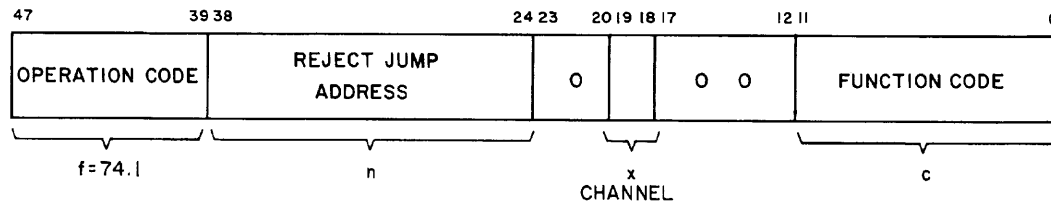
Under certain conditions, it may not be possible to connect the designated equipment. When one or more of these conditions occurs, a Reject signal is sent to the computer, and a jump is effected to the address specified by the 15-bit reject jump address.

A Reject signal will be sent to the computer only under one or more of the following conditions:

- 1) Channel Busy: The selected channel is currently performing a Read or Write operation.

- 2) Unit Unavailable: The unit referenced is in use by another I/O channel. This may occur only if an equipment is multi-channel (such as a magnetic tape controller).
- 3) False Reference: Whenever the computer receives no response within 100 usec, it generates its own Reject signal and performs the jump. This case may occur if the referenced equipment is not attached to the specified channel or the equipment is inoperative.

EXTF Function Op.Code 74.1



The Function instruction specifies operating conditions within an external equipment or a condition on which interrupt may occur. This instruction transmits a 12-bit Function code 'c' to the equipment connected to channel 'x'. This code specifies internal operating conditions of the referenced equipment. A list of codes for each external equipment is included in the associated reference manual.

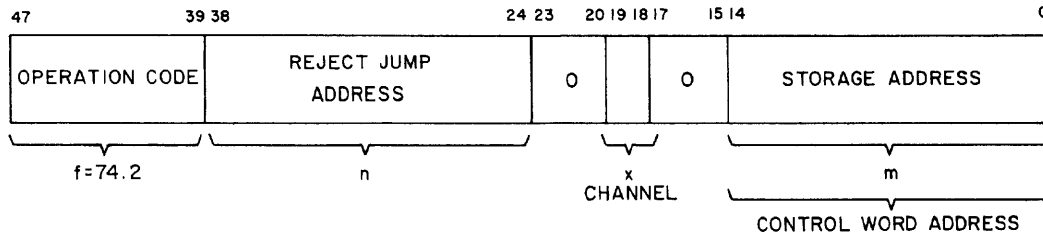
If an equipment to which the Function instruction is directed has not been previously connected to the system via a Connect instruction, the Function code cannot be recognized, and a Reject signal will be generated. The Reject signal causes the program to jump to the 15-bit reject jump address. The following conditions or combination of conditions will result in a reject.

- 1) No Unit or Equipment Connected: The referenced device is not connected to the system and

cannot recognize a Function instruction. If no response is received within 100 usec, the Reject signal is generated automatically by the computer.

- 2) Illegal Code: The Function code 'c' cannot be interpreted by the specified device. The Reject signal is generated by the external equipment after the attempted reference.
- 3) Equipment or Unit Busy or Not Ready: The device cannot perform the operation specified by 'c' without damaging the equipment or losing data. For example, a Write End of File code will be rejected by a tape unit if the tape unit is re-winding.
- 4) Channel Busy: The selected I/O channel is currently performing a Read or Write operation.

BEGR Read Op. Code 74.2



The Read instruction initiates input activity on channel 'x'.

Unless a Reject signal is generated, the main computer program proceeds independently. If a Reject signal is generated, a jump is effected to the 15-bit reject jump address. A Reject signal is generated by a Channel Busy (the designated channel is currently performing a Read or Write operation).

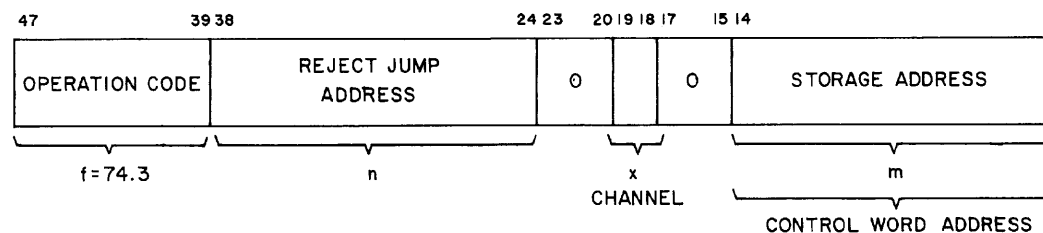
If no Reject signal is given, the control word is fetched from the storage address designated by 'm'. The word count* and starting address are placed in their respective registers in I/O channel 'x'.

This activity occurs during main computer program activity. All storage references required for a Read operation are spaced between the main program storage references.

The designated I/O channel controls all input activity until the channel becomes inactive, i.e., the word count = 0, an End of Record signal is received from the external equipment, or an external MC is issued.

If the channel only receives a partial 48-bit word and then an End of Record signal, the remainder of the word is filled with zeros and written into storage. Then the control word is updated in the normal manner and the operation terminates.

BEGW Write Op. Code 74.3



Reverse Read

Certain external equipments, such as tape units, have reverse read capabilities. When the external equipment is performing a reverse read operation, a Reverse Assembly signal is sent to the data channel. When the data channel receives this signal, the 12-bit bytes of information from the external equipment are assembled in reverse order into 48-bit words and sent to the computer.

- Note: 1) During a reverse read operation, the 48-bit word the computer receives from the data channel is identical to the 48-bit word that was sent to the data channel during the write operation.
- 2) The order in which the words are placed in storage is reversed from that of a normal read operation.

For example, suppose 100g words are stored in locations 00100g through 00177g. These words are then written on tape and a reverse read is immediately performed (store the words in locations 00100g through 00177g). The word which was in location 00177g will now occupy location 00100g, etc. The word which was in location 00100g is now located in location 00177g.

* If the original word count = 0, the channel is not activated.

The Write instruction initiates output activity on channel 'x'.

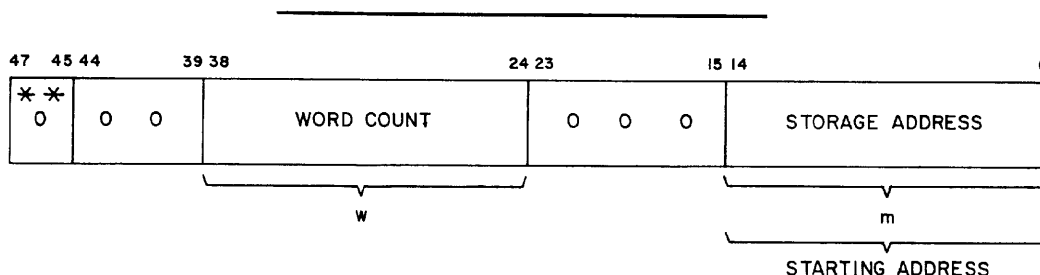
Unless a Reject signal is generated the main computer program proceeds independently. If a Reject signal is generated, a jump is effected to the 15-bit reject jump address. A Reject signal is generated by a Channel Busy (the designated channel is currently performing a Read or Write operation).

If no Reject signal is issued, the control word is

fetched from the storage address designated by 'm'. The word count* and starting address are placed in the proper registers in channel 'x'.

This activity occurs during main computer program activity. All storage references required for a Write operation are spaced between the main program storage references.

The designated I/O channel controls all output activity until the channel becomes inactive.



CONTROL WORD

All data transmission between the system and external units is governed by control words associated with each I/O channel. The control word specifies the number of data words to be transmitted into or out of the system, and the starting address in storage for the list of words.

The control word, prior to the input or output operation, is located in storage. Its location is designated by a 15-bit control word address. A Read or Write instruction transmits the control word to the data channel. The control word specifies a 15-bit starting address, 'm', from which the first output word

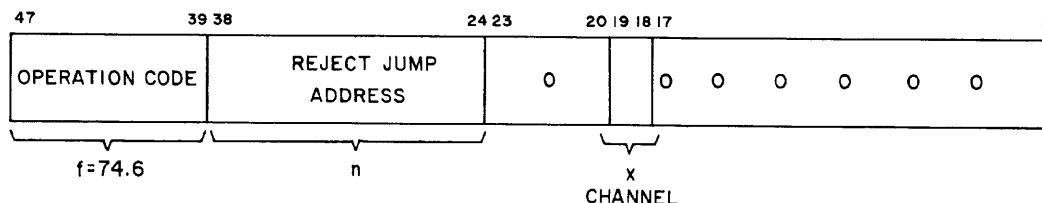
will be stored. A 15-bit word count 'w' specifies the number of data words to be transmitted. The starting address and word count are placed in their respective registers within the designated I/O channel.

When the current operation is complete (word count is zero), the I/O channel halts the input or output activity (disconnects the channel).

If the word count is not reduced to zero when address 7777 is referenced, the next word written into or read from storage will reference address 00000.

If a return to address 00000 is not desired, the word count must be such that it is reduced to zero when address 7777 is reached.

CCWD Change Control Word Op. Code 74.6



* If the original word count = 0, the channel is not activated.

** These bits correspond to the 'f' portion of the 3600 control word and are not translated in the 3400. Some programmers may wish to use an octal 3 in these bit positions to achieve an identical I/O operation when 3400 programs are run on the 3600.

The Change Control Word instruction is the only way to "chain"* a 3406 I/O channel. Previous to the execution of this instruction, the program must have preset the A register to contain the new control word for the I/O channel. (Bits 00-14 of A = storage address, bits 24-38 of A = word count.) When this instruction is executed, the following events occur:

- 1) If bits 24-38 (word count portion) of the A register are not equal to zero, AND the selected channel is active, the computer waits on this instruction until the selected channel has a word count of zero. Instead of terminating the channel operation, the new control word in the A register is sent to the I/O channel and read/write operation continues. Instruction execution then continues.
- 2) If bits 24-38 of the A register are equal to zero, or the selected channel is inactive, or both, a jump to the reject jump address is executed.

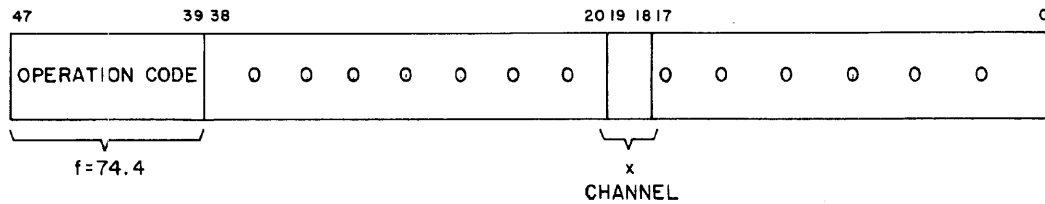
RESTRICTIONS

- a) During execution of this instruction, all interrupts are locked out.
- b) Note that one or several clock interrupts could be missed entirely.
- c) No computation takes place during the execution of CCWD.
- d) If the data channel receives an End of Record signal from the external equipment during the time the new control word is sent to the data channel, this becomes a pass instruction. For example, it is not possible to "chain over" an End of Record check character when doing a Read operation on magnetic tape.

WARNING

Do not attempt to "chain" more than one channel without full consideration of timing limitations.

COPY Copy Status Op. Code 74.4



After an external equipment has been connected, the Copy Status instruction may be used to determine:

- 1) The operating conditions
- 2) The progress of a data transmission

The external equipment issues a 12-bit status code to indicate current operating conditions. (For a list of status codes, refer to the reference manual for each equipment.) This code is present at all times on lines from the external equipment to the data channel to which it is "connected".

The status of a data transmission may be determined by examining the word count and the current address. These quantities are held in registers in the I/O

channel. Since the word count is reduced by one and the address is increased by one for each data word transmitted, the number of words processed in the operation may be determined at any time by examining these quantities. (The sum of the word count and the current address is always constant.)

The Copy Status instruction is used to determine if the data channel is busy, and to sample the status of the external equipment. This instruction transmits the current 'x' word count and address to A, and the channel status bits to Q. Bits are placed in A as follows: .00 → 14 = current address, 24 → 38 = current word count. Bits are placed in Q as follows: 0 → 11 = equipment status, 24 = 31 = channel interrupt lines (bit 24 = equipment 0, bit 31 = equipment 7).

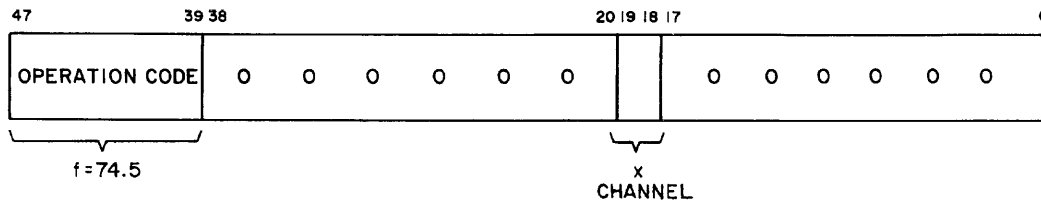
The external equipment sends and Interrupt signal

* "Chaining" refers to reading/writing to/from non-sequential locations in storage using a single read or write instruction. The blocks of information must contain at least one 48-bit word.

to the data channel on one of eight interrupt lines when the interrupt occurs. The line number corresponds to the setting of the equipment number selection switch. For example, if an equipment is designated number 3, any interrupt occurring in equipment #3 will be transmitted to the data channel on

interrupt line 3. If the interrupt is active when a Copy Status instruction is executed, bit 26 of the Q register will be set. Further discussions of external interrupts will be found in the external equipment reference manuals.

CLCH Clear Channel Op. Code 74.5



The Clear Channel instruction:

- 1) Disconnects all equipments from the specified channel, preventing any communication until a Connect instruction is executed.
- 2) Disconnects all units within an equipment.
- 3) Clears the channel control word to its original state; i.e., a control word containing all zeros. This cuts off any data transmission in progress.

This instruction assures a cleared channel prior to use when the previous condition of the channel is unknown, or removes a hang-up condition caused by a malfunction.

AUTO-LOAD

The auto-load feature provides a means by which data or instructions contained in an external storage medium may be automatically loaded into storage in the 3400 system. The 3404 performs an internal and external master clear, connects a data channel and an external equipment, and specifies a function. The 3404 then directs the channel to read to end of record, without fetching a control word from storage. The operator can specify the channel and equipment to be connected and the function to be executed by means of switches on the maintenance panel.

The sequence of events after the auto-load switch is depressed is:

1. 40 μ sec external master clear.
40 μ sec internal master clear.
2. Connect external equipment via the data channel.
3. Send specified function code to external equipment.
4. Transmit control word to data channel and start the read operation. The word count is set to 77777g and the first word is read into location 00000g in storage. Reading continues until the data channel receives an End of Record signal from the external equipments.
5. When the channel goes inactive, a Go pulse is issued and the computer begins executing instructions starting at P = 00000g. (Note that one of the results of pressing the Restart switch is a return jump to P = 00000g. This will change the original upper address of storage location 00000g.)

Any external storage medium may be auto-loaded, including magnetic tape, disc file, and punched cards. The magnetic tape produces an End of Record signal after each record. In the case of punched cards, each individual card constitutes a logical record. If several cards are to be auto-loaded, the first card must contain a "loader". This is a programmed Read instruction covering the remaining cards.

CHAPTER VI

PARITY

In the 3400 system, parity bits are generated and checked to determine one of two possible conditions:

- 1) Errors in I/O channel transmissions
- 2) Errors in storage of data or instruction words.

The presence of a parity error may indicate one or more of the above has occurred. Odd parity is used; that is, the parity bit is set such that the total number of ones in the word portion is odd.

In addition to the two cases listed above, the external equipment itself may generate and check parity on its internal operations. Treatment of this case is discussed later in this chapter. For operational details, refer to the Reference Manual for the specific equipment.

PARITY GENERATION

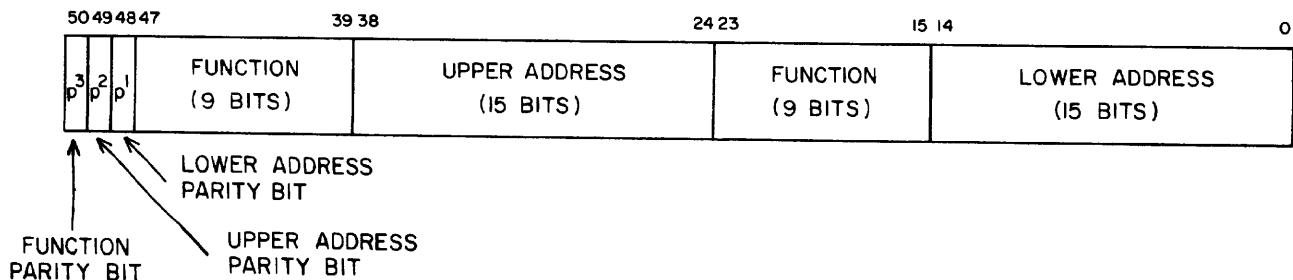
Parity bits are generated in the following cases:

- 1) On all words entering storage
- 2) On each 12-bit byte of data transmitted to an external equipment from the I/O channel (generated by the data channel).
- 3) On each 12-bit byte of data transmitted to the I/O channel from an external equipment (generated by the external equipment).

Data Parity Generation

The computer provides three parity bits along with the 48-bit instruction or data word on a Write into Storage operation. The format of this word with its associated parity bits is diagrammed below.

A parity bit is generated by the computer for each of the three portions of the instruction or data word.



Parity Generation For I/O Channel Transmissions

Each 12-bit byte of data being transmitted into an I/O channel from an external equipment has an accompanying parity bit. This parity bit is generated by the external equipment. The I/O channel generates a parity bit to accompany a 12-bit byte of data being transmitted to an external equipment.

PARITY CHECKING

Parity Checking on Storage or Transmission of Data

When an instruction or data word is read from storage by the computer, the computer checks for a parity error. The existence of a parity error may indicate one or more of the following:

- 1) An error may have occurred either when the word was initially sent to storage or when the word was read from storage.

- 2) The information was garbled in the storage read/write process itself.

Parity Checking on I/O Channel Transmissions

Each 12-bit byte of data being transmitted from an external equipment into the I/O channel is checked for a parity error by the I/O channel. A parity error indicates a transmission error has occurred. A 12-bit byte of data being transmitted to an external equipment is checked for a parity error by the external equipment.

PARITY ERRORS

When the parity errors occur, the manner in which they are handled depends on the program. Descriptions of parity errors given below list the cases.

Operand Parity Error

A word read from storage for use as an operand by the computer or for use as an output word by an I/O channel is checked for parity. If an operand parity error occurs, and if the Operand Parity Error bit in the Interrupt Mask register is set (assuming interrupt is active), interrupt occurs. Otherwise the word is accepted regardless of error. In either case the Operand Parity Error indicator lights on the console.

Instruction Parity Error

An instruction word read from storage is checked for parity in the computer. If an instruction parity error occurs, and if the Instruction Parity Error bit in the Interrupt Mask register is set (assuming interrupt is active) interrupt occurs. Otherwise the instruction will not be executed and the computer stops. In either case, the instruction parity error lights an indicator on the console.

I/O Channel Transmission Parity Error

I/O channel transmission parity errors may occur during several operations involving the I/O channel. Each of these cases is outlined below.

- 1) If a parity error occurs in transmitting the 12-bit Function code to an external equipment from the I/O channel, the following actions take place:
 - a) The function specified by the code is not executed.

- b) No external Reject signal is generated by the external equipment; the Reject signal is generated internally by the 3404.
 - c) No Reply signal is sent to the computer from the external equipment.
 - d) A Parity Error signal is sent through the I/O channel to the 3404. (This signal sets the I/O Parity Error bit in the data channel.) Interrupt will occur if the conditions for interrupting are present.
 - e) In order to initiate another operation, either a Restart (from the console) or a Clear Channel instruction must be executed to clear the condition.
- 2) If a parity error occurs in transmitting the 12-bit Connect code to an external equipment, the following actions occur:
 - a) The connection specified by the Connect code is not effected.
 - b) No external Reject signal is generated by the external equipment; the Reject signal is generated internally by the computer.
 - c) No Reply signal is sent to the computer from the external equipment.
 - d) No Parity Error signal is generated.
 - e) In order to initiate subsequent operation, either an external master clear (from the console) or a Clear Channel instruction must be performed to clear the condition.
 - f) An equipment detecting a Transmission Parity Error in a Connect code is disconnected from the channel, i.e., no status information is available to the channel from that equipment.
 - 3) If a parity error occurs in transmitting a 12-bit byte of data from the I/O channel to an external equipment, the following actions occur:
 - a) A Parity Error signal is sent through the I/O channel to the 3404. (This signal sets the I/O Parity Error bit in the data channel.) Interrupt will occur if the conditions for interrupting are present.
 - b) The Write operation continues as specified by the control word unless interrupt on I/O

parity error is selected. (Refer to following discussion on interrupt selection on I/O parity errors.)

- 4) If a parity error occurs in transmitting a 12-bit byte of data into the I/O channel from an external equipment, the following actions occur:
 - a) A Parity Error signal is returned to the 3404. (The I/O Parity Error bit in the I/O channel is set.) Interrupt will occur if the conditions for interrupting are present.
 - b) The Read operation continues as specified by the control word unless interrupt on I/O parity error is selected. (Refer to following discussion on interrupt selection on I/O parity errors.)
 - c) In order to initiate subsequent operation, either an external master clear (from the console) or a Clear Channel instruction must be performed to clear the condition.

Interrupt Selection on I/O Parity Error

Interrupt may be selected to recognize parity errors occurring in Input/Output operations. Selecting interrupt causes the I/O Parity Error bit in the I/O channel to be examined. The conditions which set this bit are:

Data transmission parity errors:

- a) Parity error on Function code
- b) Parity error on 12-bit data transmission to external equipment from channel

- c) Parity error on 12-bit data transmission from external equipment to channel

Selecting interrupt to recognize one of these occurrences is accomplished as follows:

- 1) Execute Internal Function instruction (77.0003X0 - Select Interrupt on I/O Parity Error).
- 2) Activate interrupt system via Internal Function instruction (77.00600 - Set Interrupt Active).

When interrupt is selected and one of the above conditions occurs, an interrupt routine is entered. Within the interrupt routine, the computer isolates the cause of interrupt to an I/O parity error. An I/O parity error constitutes a major machine malfunction. (Refer to chapter 4 and appendix A.)

External Equipment Parity Error

Parity errors within an external equipment (e.g., longitudinal parity error during a Read/Write operation on magnetic tape) are handled differently than the I/O parity errors. These parity errors indicate a difficulty in reading or writing information on the equipment's medium.

If a parity error occurs within the external equipment, a Parity Error signal is returned to the I/O channel via one of the twelve status lines.

Once the interrupt routine has isolated the cause of interrupt to a particular equipment, a Copy Status instruction must be executed to determine the particular condition causing interrupt (e.g., longitudinal parity error). Refer to chapter 4.

CHAPTER VII

3401 CONSOLE

The 3401 console includes various switches, a System Status Display panel, a speaker, and an electric typewriter. Through the use of these devices, the operator can control and monitor the computer and associated peripheral equipments.

SWITCHES

Console switches and their operations are listed in table 7-1.

Table 7-1. Console Switches

Switch	M or L*	Function
Selective Jump 1	L	Pressing the Selective Jump switches provides the manual conditions for executing a program jump on the Selective Jump Instruction (75). The Selective Jump switches are illuminated when depressed and can be activated when the computer is running or stopped.
Selective Jump 2	L	
Selective Jump 3	L	
Auto-Load	M	Pressing the Auto-Load switch provides for automatically loading storage with information from a given external equipment and initiating program execution.
Emergency Off	M	Pressing the Emergency Off removes power from the entire system.
Manual Interrupt	M	Pressing the Manual Interrupt switch forces the computer into a Category I interrupt routine if: <ol style="list-style-type: none"> 1) The Manual Interrupt bit in the Interrupt Mask register is set, and 2) The interrupt system is active.
End of Record	M	When depressed, this switch ends an End of Record signal to the data channel if the typewriter attached to the data channel is "connected" and read and data signals are present.
Equipment Number	L	This eight-position (0-7) switch designates the console typewriter as equipment N. Bits 9, 10, and 11 of the Connect code must match the octal setting of this switch. (This switch is mounted inside the console on the typewriter logic chassis.)
Typewriter On/Off	L	This switch is located near the typewriter keyboard. It must be in the On position for any typewriter operation.

* M - Momentary, L - Locking

Table 7-1. (Continued)

Switch	M or L	Function
Fake Reply On/Off	L	When in the On position, this switch gives a fake Reply to Read/Write signals on the data channel cabled to the typewriter. This switch is for maintenance use only.
Restart	M	When this switch is depressed, the following events occur: 1) Stop computer and issue an External MC. This stops all computer and I/O activity. 2) Clear Interrupt Active 3) Restart computer with a return jump to address 00000g. P is stored at the upper address portion of location 00000g, when the return jump is executed. This P is equal to the address of the upper instruction that was about to be executed when the computer stopped.
System Active On/Off	L	When this switch is On, a "running time" meter (located in the power control cabinet) accumulates the time that power is on and the System Active switch is on. If both the System Active and the Maintenance Mode switch (located on the maintenance panel) are off, all other switches except power control will be locked out. When the System Active switch goes from Off to On, both Internal and External Master Clears are issued. When the System Active switch goes from On to Off, the computer stops. Any I/O activity continues until normal termination. The System Active switch is illuminated when in the On state.

INDICATORS

By means of lights, the System Status Display panel indicates certain operating conditions in the computer system. Figure 7-1 shows the System Status

Display panel. Table 7-2 gives the conditions indicated by the lights. All displays are active when the condition arises, whether the computer is running or stopped. For definitive descriptions of various fault conditions, refer to appendix A.

Channel 0 Active	Channel 0 Parity Error	Computer Running	Arithmetic Overflow	Shift Fault	Read Next Instruction	Execute Cycle	Storage Overload 1	Temperature 1	Circuit Breaker
Channel 1 Active	Channel 1 Parity Error	Program Stop	Exponent Overflow	Divide Fault	Upper Instruction	Operand Cycle	Storage Overload 2	Temperature 2	Temperature
Channel 2 Active	Channel 2 Parity Error	Interrupt Active	Exponent Underflow	Instruction Parity Error	Lower Instruction	Extended Cycle		Temperature 3	Interlock Bypassed
Channel 3 Active	Channel 3 Parity Error	Interrupt Mode		Operand Parity Error	Maintenance Mode	Jump Condition		Temperature 4	Terminator Power Fault

Figure 7-1. System Status Display Panel

Table 7-2. Display Panel Indicators

Display	Color	Condition When Illuminated
Channel 0 Active Channel 1 Active Channel 2 Active Channel 3 Active	Green	The channel active indicator(s) lights when an I/O channel(s) is performing a read or write operation.
Channel 0 Parity Error Channel 1 Parity Error Channel 2 Parity Error Channel 3 Parity Error	Red	These indicate an I/O channel transmission Parity Error has occurred in one or more I/O channels. (See Chapter 6 of this manual.)
Computer Running	Green	This indicates the computer is running and/or I/O operations are taking place.
Program Stop	Red	The Program Stop indicates that the computer is no longer executing program instructions. (If the Computer Running light comes on and then goes out, but the Program Stop light does not come on, it indicates a "deep-end" condition: i.e., there is an electronic malfunction in the computer.)
Interrupt Active	Orange	The interrupt system is enabled, permitting examination of selected interrupt conditions in an interrupt routine when these conditions occur.
Interrupt Mode	Orange	The computer is processing an interrupt routine.
Arithmetic Overflow	Orange	The absolute value of the sum or difference of two fixed point integers is $\leq 2^{47}$.
Exponent Overflow	Orange	The value of the exponent formed during a floating point add, subtract, multiply, or divide is $> 2^{10} - 1$ (1777 ₈).
Exponent Underflow	Orange	The value of the exponent formed during a floating point add, subtract, multiply, or divide is $<$ negative $2^{10} - 1$ (-1777 ₈).
Shift Fault	Orange	A register shift of more than 48 or 96 places is specified in a Shift instruction.
Divide Fault	Orange	1) The absolute value of the quotient resulting from a Divide Integer instruction is $\geq 2^{47}$. 2) A fixed point or floating point divide by zero is attempted.
Instruction Parity Error	Red	A parity error occurs when reading a 48-bit instruction word from storage.

Table 7-2. (Continued)

Display	Color	Condition When Illuminated
Operand Parity Error	Red	A parity error occurs in a word read from storage for use as an operand by the computer or for use as an output word by an I/O channel.
Read Next Instruction	White	The computer is reading a 48-bit instruction word from storage.
Upper Instruction	White	The computer is executing the upper instruction portion of a 48-bit word.
Lower Instruction	White	The computer is executing the lower instruction portion of a 48-bit word. (Only the Upper Instruction indicator is illuminated when the computer is executing a 48-bit instruction.)
Maintenance Mode	Red	The Maintenance Mode switch on the maintenance panel is active.
Execute Cycle	White	The computer is executing an instruction which requires no memory reference.
Operand Cycle	White	The computer is executing an instruction which requires referencing memory.
Extended Cycle	White	The computer is executing an instruction which requires two cycles.
Jump Condition	Green	The computer is about to execute a Jump instruction and the condition for jumping has been met.
Storage Overload 1 Storage Overload 2	Red Red	A current overload has occurred in the storage stack(s).
Temperature 1 Temperature 2 Temperature 3 Temperature 4	Orange	The temperature in one or more cabinet in the system is exceeding the normal range.
Circuit Breaker	Red	A circuit breaker(s) in the system has tripped due to a current overload.
Temperature	Red	The temperature in a section of the system has reached a point where the equipment may be harmed. The motor-alternator is turned off.
Interlock Bypassed	Orange	The Interlock Bypassed switch on the maintenance panel has been depressed. Computation proceeds regardless of the temperature in any part of the system.

Table 7-2. (Continued)

Display	Color	Condition When Illuminated
Terminator Power Fault	Red	If the terminator power fluctuates or drops, this indicator lights.
Type-In	White	This indicator is located next to the typewriter. When this light comes on, it indicates that the computer is executing a read instruction and the input operation may begin via the typewriter.

TYPewriter

A 731 Selectric typewriter is used as a program monitor device in the computer system. The typewriter is cabled to an I/O channel and is operated using computer instructions pertinent to external equipments. These instructions include Connect, Function, Copy Status, Read, and Write. (For a description of switches and indicators related to typewriter operations, see the preceding section on switches and indicators in this chapter.)

The typewriter codes in tables 7-3 and 7-4 are used for both input and output operations.

CONNECT

Connect Equipment N (NXXX)

Bits 9, 10, and 11 of the Connect code must match the octal setting of the Equipment Number switch. If the switch setting and these bits do not agree, the typewriter will not be connected. No signals will be returned to the I/O channel. A new Connect code not equal to NXXX clears a previous connection unless a transmission parity error occurs or has occurred and has not been cleared.

FUNCTION

Set Interrupt On Abnormal Operation (XX01)

If bit 0 is present in the Function code (accompanied by a Function signal), an Interrupt on Abnormal Operation condition is established in the typewriter. (In all discussion of codes, bit 0 is in the rightmost position, and bits are numbered from right to left in ascending order.) This allows the typewriter to send an interrupt signal to the computer when either or both of two conditions occur:

1) End of Line

The typewriter has reached the right margin of the line it is typing.

2) Type Parity Error

A parity error has occurred in the typewriter logic. (See the Type Parity Error discussion under Status Lines for additional comments.)

The Interrupt signal is transmitted on the line which corresponds to the equipment number (N) of the typewriter.

Clear Interrupt On Abnormal Operation (XX02)

If bit 1 is present in the Function code, the Interrupt on Abnormal Operation condition is cleared.

Clear Interrupt (XX04)

If bit 2 is present in the Function code, the Interrupt signal is cleared.

STATUS

Ready (XXX1)

If bit 0 is present in the status code, power has been applied to the control logic, the typewriter is turned on and "connected", and I/O operations may proceed.

Busy (XXX2)

If bit 1 is present in the Status code, it indicates that the typewriter is busy. When in this condition,

Table 7 - 3. 731 Typewriter Codes

Manifold Ten*

Lower Case	Code	Upper Case	Lower Case	Code	Upper Case
A	12				
B	01		0 (zero)	43)
C	11		1	77	±
D	55		2	37	@
E	51		3	33	#
F	30		4	47	\$
G	74		5	57	%
H	45		6	13	¢
I	16		7	53	& (and)
J	70		8	17	* (asterisk)
K	15		9	07	(
L	41		!	76	° (degree)
M	72		.	32	.
N	31		,	52	''
O	42		;	50	:
P	54		,	14	,
Q	10		/	44	?
R	56		=	34	+
S	46		Dash -	04	Underline _
T	75		Space	60	Space
U	35		Backspace	61	Backspace
V	36		Tab	62	Tab
W	02		C. R.	63	C. R.
X	71		U. C.	64	U. C.
Y	40		L. C.	66	L. C.
Z	73				

* Named after the 731 type font.

Table 7-4. Connect, Function, and Status Codes

CONNECT Connect Equipment N	NXXX
FUNCTION Set Interrupt On Abnormal Operation Clear Interrupt On Abnormal Operation Clear Interrupt	XX01 XX02 XX04
STATUS Ready Busy End of Line Upper/Lower Case Type Parity Error	XXX1 XXX2 XX4X XXX4 2XXX

the typewriter is doing one of the following operations:

- 1) Typing a character
- 2) Backspacing
- 3) Doing a carriage return or tab
- 4) Shifting to the upper case
- 5) Shifting to the lower case
- 6) Processing a 00, 65, or 67 code.

If one or both 6-bit frames of the 12-bit word received by the typewriter during a normal output (Write) operation equal 00, 65, or 67, nothing is typed or spaced corresponding to that code. The time interval required to process these codes is approximately 3 usec.

Upper/Lower Case (XXX4)

If bit 2 is present in the Status code, the typewriter is in upper case. If bit 2 is not present in the Status code, the typewriter is in lower case.

End of Line (XX4X)

If bit 4 is present on the status lines, it indicates the typewriter has reached the end of a line. Any further codes received by the typewriter will all be typed in the end character location unless a carriage return is executed.

Type Parity Error (2XXX)

If bit 10 is present in the Status code, a type parity error has occurred. This indicates some logic has failed in the typewriter control and the character typed is not necessarily the one specified by the code. A type parity error can only occur during an output operation. The Type Parity Error signal drops when an I/O master clear or a Clear Channel instruction is performed. It also drops if a Carriage Return or Clear Interrupt Function code is received.

PROGRAMMING

The general order of events when using the typewriter for an input/output operation via an I/O channel is:

- 1) Set tabs, margins, and spacing. Turn on the typewriter, logic control power.
- 2) Clear
- 3) Connect
- 4) Check status
- 5) Function
- 6) Write/Read

Set Tabs, Margins, and Spacing

All tabs, margins, and paper spacing must be set manually prior to the output operation. A tab may be set for each space on the typewriter between the margins.

Clear

There are two types of clears which may be used to clear all conditions existing in the typewriter control. These are:

- 1) External Master Clear.
This signal is sent out on all I/O channels. In the case of the typewriter, it clears all functions, control logic, and an Interrupt signal if one exists. The external MC clears the Interrupt condition selected by a XX01 function code.
- 2) Clear Channel Instruction.
This instruction performs the same operation as an external master clear, except it normally only applies to one I/O channel. (See this reference manual for a detailed description of this instruction.)

Connect

The 12-bit portion of the Connect instruction (accompanied by a Connect signal) connects the typewriter to an I/O channel. A Reply signal is returned to the channel when the connection has been made. There is no external reject under any circumstances.

Check Status

The programmer may wish to check the status of the connected typewriter before proceeding. This is done with a Copy Status instruction. (See this reference manual for a detailed description of Connect, Function, Copy Status, Bit Sensing, and Write instructions.) Status information is returned to the I/O channel on five of twelve status lines. The Bit Sensing instruction may be used to determine the status of the connected typewriter. If the programmer is certain of the status of the typewriter, this operation may be omitted.

Function

The 12-bit portion of the Function instruction (accompanied by a Function signal) performs a certain operation (depending on the code). These codes are listed in table 7-4. Since only 1 bit of the Function code needs to be interpreted to perform a specified operation, it is possible to combine operations using one code. For example: $XXX6g = (XXX110)_2$ would Clear Interrupt on Abnormal Operation and Clear Interrupt.

Example:

XXX4	Clear Interrupt
<u>XXX2</u>	Clear Interrupt on Abnormal Operation
XXX6	Clear both

Write

A Write instruction starts the output operation using the codes listed in table 7-3. The upper 6 bits of the 12-bit data word received from the I/O channel are translated and the character matching the code is typed. Then the lower 6 bits of the code are translated, the character is typed, and a reply is returned to the data channel.

Note that the typewriter must be in the lower case to type letters. If the typewriter is in the upper case when a letter is to be typed, the typewriter will space. No letter will be typed, but the operation will continue as in a normal output.

The typewriter is automatically placed in lower case at the beginning and end of every Write (output) operation (upper case can still be selected when the Write operation begins).

The typewriter keyboard does not have a lockout during an output operation. If a key, space bar, etc., is accidentally pressed during output, the typewriter may miss a character or type something other than the normal output character.

Read (input)

A Read instruction starts the input operation using the codes listed in table 7-3. When the type-in indicator on the console lights, the operator may enter information on the typewriter.

Input is character mode only. Six bits of information are entered into bit positions 0-5 of the 48-bit input word. Bit positions 6-47 are filled with zeros. Thus, the first three 12-bit words the I/O channel receives from the typewriter are all equal to zero. The last 12-bit word the I/O channel receives from the typewriter has a six-bit code in bit positions 0-5 (bit positions 6-11 = zeros).

During a Read operation, codes are entered corresponding to the symbols typed, except for the capital letters A through Z. If the typewriter is in the upper case during a Read operation, the code corresponding to the letter will be sent to the channel, the letter will not be typed, and the typewriter will space.

The mechanical operations, such as backspace or carriage return, send their corresponding codes to the I/O channel. For example: shifting from lower to upper case enters a 64g; shifting from upper case to lower case enters a 66g.

Note: When the End of Record switch is depressed, the channel terminates after writing a 48-bit word of zeros into memory and updating the control word.

3404 MAINTENANCE PANEL AND POWER CONTROL PANEL

The 3404 Maintenance Panel includes a System Status Display Panel, switches and indicators. The power control panel includes circuit breakers, meters, and dials for regulating the 14 autotransformers which supply ± 20 volts dc power to the computer logic.

Figure 7-2 shows the Maintenance Panel. This side includes the System Status Display panel, switches, and indicators.

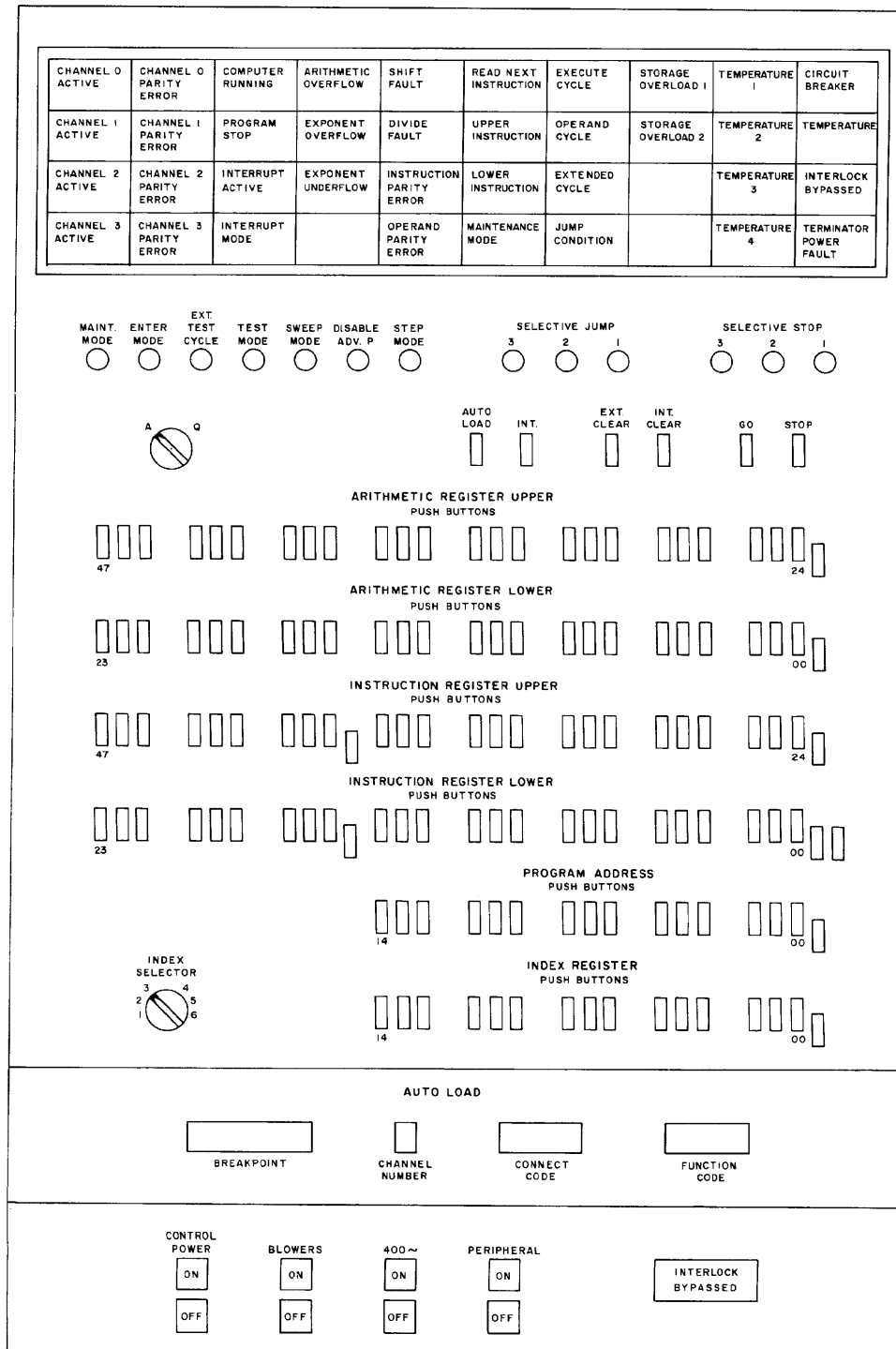


Figure 7-2. 3404 Maintenance Panel

System Status Display Panel

This is identical to the System Status Display panel on the console. Refer to table 7-2 in this chapter for a discussion of these indicators.

Switches and Indicators

Table 7-5 lists the switches and indicators used when operating the computer from the Maintenance Panel.

Table 7-5. 3404 Switches and Indicators

Switch	No.	M or L*	Function
Maintenance Mode	1	L	<p>This places the computer in Maintenance Mode. The following events occur:</p> <ol style="list-style-type: none"> 1) The push buttons are enabled for entering/clearing information in the Arithmetic, Instruction, Program Address, and Index registers (provided the computer is stopped and no data channel(s) is active). 2) The Go switch is enabled (providing no data channel(s) is active and the computer is not running in Test Mode). 3) The Maintenance Mode indicators on the console and maintenance panel light. These lights remain lit until the Maintenance Mode switch is deactivated. <p>Note: All console switches remain active when the computer is in Maintenance Mode.</p>
Enter Mode	1	L	<ol style="list-style-type: none"> 1.) If the A register is selected, the following events occur when the Go switch is depressed (Enter Mode switch activated): <ol style="list-style-type: none"> a) The contents of the Arithmetic register (A) are transferred to the Instruction register (U). b) The contents of U are entered into the storage location designated by the contents of the Program Address register (P). c) The Arithmetic register is cleared. d) P+1 replaces the contents of P; the computer stops. 2.) If the Q register is selected, the following events occur when the Go switch is depressed (Enter Mode switch activated): <ol style="list-style-type: none"> a) The contents of the Arithmetic register (Q) are transferred to the Instruction register (U).

* M - Momentary, L - Locking

Table 7-5. (Continued)

Switch	No.	M or L	Function
Enter Mode (cont'd.)			<p>b) The contents of the Instruction register are entered into the storage location designated by contents of the Program Address register (P).</p> <p>c) P+1 replaces the contents of P, the computer stops.</p>
Test Mode	1	L	<p>When this switch is activated the following events occur (Extended Test Cycle switch not activated):</p> <ol style="list-style-type: none"> 1) 10μs internal MC. 2) 10μs wait. 3) 0.1μs Go pulse. 4) 120μs program run. 5) 10μs stop computer (clear run). <p>Steps 1-5 continue repeating until the Test Mode switch is deactivated.</p> <p>Note: If Step Mode is active, 1) above = 10μs wait.</p>
Extended Test Cycle	1	L	<p>If this switch is activated, the following events occur the next time the Test Mode switch is activated:</p> <ol style="list-style-type: none"> 1) 5ms internal and external MC. 2) 10μs wait. 3) 0.1μs Go pulse. 4) 30ms program run. 5) 10μs stop computer (clear run). <p>Steps 1-5 repeat until the Test Mode switch is deactivated. If the Extended Test Cycle switch is deactivated while the Test Mode switch is still active, the computer continues in the normal Test Mode.</p> <p>Note: If Step Mode is active, 1) above = 5ms wait.</p>
Sweep Mode	1	L	<p>When the Sweep Mode switch is activated (Step Mode switch active), the contents of the storage location specified by the contents of the Program Address register are displayed by the U Instruction register indicators the next time the Go switch is depressed. The computer stops when the above operation is complete (No instruction execution). When the Go switch is depressed again, the contents of P + 1 are displayed by the U register indicators.</p> <p>Activating the Sweep Mode switch disables the Enter Mode switch logic. This prevents accidental program destruction.</p> <p>Note: For maintenance use, the computer will continually sweep memory (no instruction execution) if the Step Mode switch is inactive when Sweep Mode is active and Go is depressed.</p>

Table 7-5. (Continued)

Switch	No.	M or L	Function
Disable Advance P	1	L	<p>This disables (prevents) advancing the Program Address register when the Go switch is depressed. The computer will loop on the same instruction or instruction pair.</p> <p>If a Jump instruction is present, the computer will perform the jump if the conditions are met. The computer will then continue looping at the new address unless another jump instruction is present and the jump conditions are met, in which case the jump will again be performed.</p>
Step Mode	1	L	<p>Activating the Step Mode switch places the computer in Step Mode. A step may be defined as that portion of a program step executed between successive pressings of the Go switch (where a program step is an instruction pair or a single 48-bit instruction). Since there are steps which differ slightly from the foregoing (depending on the sequence of events), the possible cases are outlined below.</p> <p>Case 1: Perform internal master clear; select Step Mode; press Go switch. In this case, the instruction is read from storage and the computer stops before its execution. Pressing the Go switch again results in the following operations:</p> <ul style="list-style-type: none"> a) If indirect addressing is specified by the instruction, the indirect address is determined and the computer stops. The next time the Go switch is pressed, the instruction is executed (if indirect addressing is not called for again) and the computer stops after: 1) reading (from the U register) the lower instruction of an instruction pair; or 2) reading a new instruction word from storage (if the previous instruction was a 48-bit instruction). b) If indirect addressing is not specified by the instruction, the instruction is executed and the computer stops after 1) or 2) above. <p>Case 2: No internal master clear has been performed, select Step Mode; press Go switch. In this case, the following operations occur:</p> <ul style="list-style-type: none"> a) If indirect addressing is specified by the instruction, the indirect address is determined and the computer stops before executing the instruction. b) If no indirect addressing is specified, the instruction is executed and the computer stops after 1) or 2) above (case 1).

Table 7-5. (Continued)

Switch	No.	M or L	Function
Selective Jump 1 Selective Jump 2 Selective Jump 3	1 1 1	L L L	Activating the Selective Jump switches provides the manual conditions for executing a program jump on the Selective Jump (75) instruction. These switches can be activated when the computer is running or stopped.
Selective Stop 1 Selective Stop 2 Selective Stop 3	1 1 1	L L L	Activating the Selective Stop switches provides the manual conditions for stopping the computer on the Selective Stop (76) instruction (computer in Maintenance Mode). These switches can be activated when the computer is running or stopped.
Auto-Load	1	L	Pressing the Auto-Load switch starts the automatic sequence of events which loads storage from a specified external equipment. This switch is active (i.e., the auto load sequence will start) when the computer is running or stopped. (The computer need not be in Maintenance Mode to activate Auto-Load with the Auto-Load switch). Refer to the Auto-Load discussion in chapter 5 for additional information.
Interrupt	1	M	Pressing the Interrupt switch forces the computer into an interrupt routine if: 1) the Manual Interrupt bit in the Interrupt Mask register is set, and 2) the interrupt system is active.
External Clear	1	M	Pressing this switch master clears all external equipments. It also clears all data registers, control registers, and most control logic in all data channels.
Internal Clear	1	M	Pressing this switch master clears all 3404 computation logic, i.e., most registers and control FFs.
Go	1	M	Pressing the Go switch initiates execution of the instruction currently in the U Instruction register if Internal Master Clear was not previously pressed. If a previous Internal Master Clear was performed, the computer reads the instruction from the address specified by the current value of P and initiates execution of that instruction. Instruction execution continues until a stop condition arises.
Stop	1	M	Pressing the Stop switch stops the computer at the completion of the current instruction. When the computer stops, the contents of the selected operational registers are displayed by the register indicators on the Maintenance Panel.

Table 7 - 5. (Continued)

Switch	No.	M or L	Function
A/Q Register Selection	1	L	<p>This switch selects which arithmetic register (A or Q) will be manipulated and displayed by the 48 Set and two Clear push buttons.</p> <p>If this switch is changed while the computer is running, the register being displayed will change.</p>
<u>Set Push Buttons</u> Arithmetic Register Instruction Register Program Address Register Index Register	48 48 15	M M M	<p>These four registers each have an indicating Set push button associated with each register stage. The set push buttons are binary switches numbered in the powers of 2, beginning with zero. Each group of 3 is an octal digit. Pressing a Set push button forces that particular stage of a register to the set (or "1") state (providing the computer is in Maintenance Mode and stopped) The particular push button is illuminated when pressed.</p> <p>Note that the A or Q Arithmetic register is chosen with the A/Q Register Selection switch; the particular Index register is chosen with the Index Register Selection switch.</p>
<u>Clear Push Buttons</u> Arithmetic Register Instruction Register Program Address Register Index Register	2 5 1 1	M M M M	<p>Pressing the Clear push button(s) associated with the register clears all or some stages of the register to "0". Five Clear push buttons are associated with the Instruction (U) register. Pressing a particular push button results in clearing:</p> <ol style="list-style-type: none"> 1) The entire register (48 bits). 2) The lower address portion (15 bits). 3) The lower operation code and B designator portion (9 bits). 4) The upper address portion (15 bits). 5) The upper operation code and B designator portion (9 bits). <p>Two Clear push buttons are associated with the A/Q Arithmetic register. Pressing a particular push button results in clearing:</p> <ol style="list-style-type: none"> 1) The upper 24 bits of the A/Q register . 2) The lower 24 bits of the A/Q register. <p>The Clear push buttons associated with the Program Address register and the Index register clear each entire register (15 bits).</p>

Table 7-5. (Continued)

Switch	No.	M or L	Function
Index Selector	1	L	This six-position switch selects which of six index registers will be displayed and manipulated by the 15 Set push buttons and the one Clear push button. When the computer is running, this switch has no function. The index register displayed is specified by the current instruction.
Breakpoint Address	5	L	Five 8-position thumb-nail switches can be set to octal addresses 00000 through 77777. (This switch may be changed while the computer is running.)
Breakpoint Address Selector	1	L	This 4-position switch permits selection of: 1) Operand address. 2) Instruction address. 3) Operand or Instruction address. 4) Off. The computer stops when the Operand/Instruction address and the address selected by the Breakpoint Address switches are equal. (The Breakpoint Address Selector switch may be changed while the computer is running.)
Channel Number	1	L	This thumb-nail switch designates which data channel (0-3) will be used during the auto-load operation. (See chapter 5.)
Connect Code	4	L	Four 8-position thumb-nail switches specify the four digit octal connect code used during the auto-load operation. (See chapter 5.)
Function Code	4	L	Four 8-position thumb-nail switches specify the four digit octal function code used during the auto-load operation. (See chapter 5.)
Control Power On	1	M	When the On switch is depressed, power is made available to the individual switches in the system, i.e., this switch brings power to the various distribution points. This switch must be activated first when applying power to the system. This switch is illuminated when depressed.
Control Power Off	1	M	This switch turns off the control power. When control power is turned off, this switch is the only one on the maintenance panel that is illuminated.

Table 7-5. (Continued)

Switch	No.	M or L	Function
Blowers On	1	M	This turns the blowers on for the entire system. This switch must be on before the 400 cycle power switch can be turned on. This switch is illuminated when depressed.
Blowers Off	1	M	This switch turns the blowers off 5-10 minutes after it is depressed (providing the 400 cycle power has been turned off). If the 400 cycle power is still on, pressing this switch has no effect. This switch is illuminated when depressed.
400 Cycle On	1	M	This switch applies 400 cycle power to the entire system, provided the blowers are turned on. It also executes a 40ms Power On MC for the 3404 and 3406 logic. If the blowers are not turned on, pressing this switch has no effect. This switch is illuminated when depressed.
400 Cycle Off	1	M	This switch turns off the 400 cycle power for the entire system. This switch is illuminated when depressed.
Peripheral On	1	M	This applies power to all peripheral equipment switches which use 220v, 60 cycle power. This switch is illuminated when depressed.
Peripheral Off	1	M	This switch drops power to all peripheral equipment switches which use 220v, 60 cycle power. This switch is illuminated when depressed.
Interlock Bypassed	1	M	Pressing the Interlock Bypassed switch overrides stopping the computer on reaching abnormal temperature conditions. The Interlock Bypassed indicators on the maintenance and console display panels light. The Interlock Bypassed switch is illuminated when depressed.

Figure 7-3 shows the Power Control Panel. This side includes the circuit breakers, autotransformer dials, and meters.

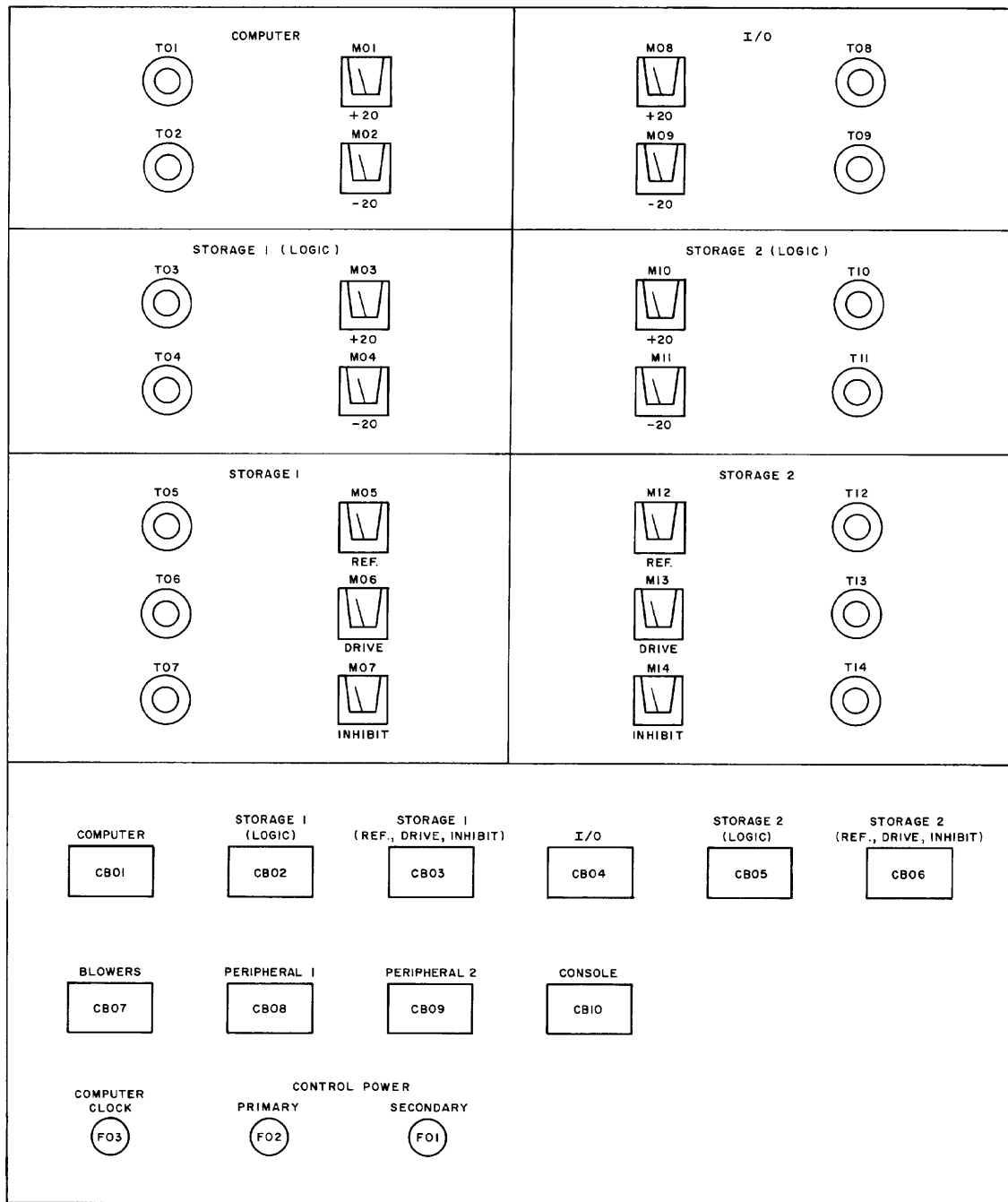


Figure 7-3. 3400 Power Control Panel

Circuit Breakers (CB01-CB10)

Protective circuit breakers are used throughout the 3400 system. Whenever an excess amount of current is drawn which may harm the equipment, the associated circuit breaker will trip. Power is restored when the circuit breaker is manually reset.

Meters (M01-M14) and Dials (T01-14)

The meters indicate the voltages that are being applied to a particular circuit by the autotransformers. The dial associated with each meter enables the operator to vary the circuit voltages by changing the autotransformer settings.

APPENDIX SECTION

APPENDIX A
INTERRUPTIBLE CONDITIONS AND FAULTS

APPENDIX A

INTERRUPTIBLE CONDITIONS AND FAULTS

Under certain internal conditions in the execution of a computer program, faults may occur. Most of these fault conditions are associated with a particular bit of the Interrupt register, and may be tested in an interrupt routine. In most cases, a fault condition does not stop operation, but a visual indication of a fault occurrence is provided on the console.

SHIFT FAULT

When a single register shift of more than $60_8 = 48_{10}$ places or a double register shift of more than $140_8 = 96_{10}$ is specified by the shift count in a Shift instruction, bit 01 in the Interrupt register is automatically set to "1". The shift will not occur. The Interrupt register bit remains set until:

- 1) an Internal Function Instruction (Clear Arithmetic Fault, Clear Shift Fault or Clear Interrupt System) is executed, or
- 2) a manual internal master clear is performed.

A shift fault lights an indicator on the console.

DIVIDE FAULT

Bit 02 of the Interrupt register is automatically set to "1" when:

- 1) The absolute value of the quotient resulting from a Divide Integer instruction is ≥ 247 .
- 2) A fixed point or floating point divide by zero is attempted.
- 3) The coefficient of the dividend ≥ 2 times the coefficient of the divisor in a floating point divide.

This Interrupt Register bit remains set until:

- 1) The Internal Function instruction (Clear Arithmetic Fault, Clear Divide Fault, or Clear Interrupt System) is executed, or
- 2) A manual internal master clear is performed.

A divide fault lights an indicator on the console.

EXPONENT OVERFLOW FAULT

Bit 04 of the Interrupt register is automatically set to "1" when the unbiased value of the exponent formed during a floating point add, subtract, multiply, or divide is $> 2^{10.1}$ (1777₈). This Interrupt Register bit remains set until:

- 1) The Internal Function instruction (Clear Arithmetic Fault, Clear Exponent Overflow Fault, or Clear Interrupt System) is executed, or
- 2) A manual internal master clear is performed.

The resulting answer is left as formed when an exponent overflow occurs. An exponent overflow fault lights an indicator on the console.

For the cases where A_i and M_i coefficients are both not equal to ± 0 and the coefficient of the result is to ± 0 , the following rule applies (note that the only equal way to obtain this situation is to start with unnormalized coefficients): If exponent overflow occurs, Q contains the residue and bit 04 in the Interrupt register is set. The exponent overflow indicator lights.

EXPONENT UNDERFLOW FAULT

Bit 05 of the Interrupt register is automatically set to "1" when the unbiased value of the exponent formed during a floating point add, subtract, multiply, or divide is $< \text{negative } 2^{10.1}$ (-1777₈). This Interrupt Register bit remains set until:

- 1) The Internal Function instruction (Clear Arithmetic Fault, Clear Exponent Underflow Fault, or Clear Interrupt System) is executed, or
- 2) A manual internal master clear is performed.

The A and Q registers are both cleared to positive zero after the end sign correction. An exponent underflow fault lights an indicator on the console.

For the cases where A_i and M_i coefficients are both not equal to ± 0 and the coefficient of the result is equal to ± 0 , the following rule applies (note that the only way to obtain this is to start with unnormalized coefficients): If exponent underflow occurs, bit 05 of the Interrupt register is not set. The exponent underflow indicator does not light.

ARITHMETIC OVERFLOW FAULT

Bit 03 of the Interrupt register is automatically set to "1" when:

- 1) The absolute value of the sum or difference of two fixed point integers is $\geq 2^{47}$. This Interrupt Register bit remains set until:
 - a) The Internal Function instruction (Clear Arithmetic Fault, Clear Arithmetic Overflow Fault or Clear Interrupt System) is executed, or
 - b) A manual internal master clear is performed.

An arithmetic overflow fault lights an indicator on the console.

I/O PARITY ERROR

When an I/O Transmission Parity Error occurs in an I/O channel, one of four bits (20-23) is set in the Interrupt register. This bit remains set until:

- 1) An Internal Function instruction (Clear I/O Parity Error or Clear Interrupt System) is executed, or
- 2) A manual internal master clear is executed.

OPERAND PARITY ERROR

A word read from storage for use as an operand by the computer or for use as an output word by an I/O channel is checked for parity. Bit 45 of the Interrupt register is automatically set to "1" if a parity error occurs.

This Interrupt Register bit remains set until:

- 1) The Internal Function instruction (Clear Interrupt System or Clear Parity Error) is executed, or

- 2) A manual internal master clear is performed.

An operand parity error lights an indicator on the console.

MANUAL INTERRUPT

When the momentary Manual Interrupt switch on the console is pressed, bit 43 of the Interrupt register is set to "1" only if the corresponding bit of the Mask register is set.

This bit in the Interrupt register remains set until:

- 1) The Internal Function instruction (Select Manual Interrupt) is executed.
- 2) The Internal Function instruction (Clear Manual Interrupt) is executed.
- 3) The Internal Function instruction (Clear Interrupt System) is executed.
- 4) A manual internal master clear is performed.

INSTRUCTION PARITY ERROR

If a parity error occurs when reading an instruction from storage, the Instruction Parity Error bit (46) in the Interrupt Register is set to "1". This bit remains set until:

- 1) An Internal Function instruction (Clear Parity Error or Clear Interrupt System) is executed, or
- 2) A manual internal master clear is executed.

OUT OF BOUNDS

Bit position 42 of the Interrupt register is automatically set to "1" when:

- 1) The interrupt system is active, and
- 2) Bit position 42 of the Interrupt Mask register is set to "1" (check bounds), and
- 3) A write reference is attempted out of bounds (the bounds addresses are defined by the contents of the two 8-bit bounds registers), or a jump is attempted out of bounds, or an attempt is made to read an instruction from out of bounds.

This Interrupt register bit remains set until:

- 1) The Out of Bounds Mask register bit is cleared by an Internal Function instruction 77.000541, or
- 2) An attempt is made to set the Out of Bounds Mask Register bit 42 with an Internal Function

instruction 77.000540, even though bit 42 is already set, or

- 3) An Internal Function instruction (Clear Interrupt System) is executed, or
- 4) A manual internal master clear is performed.

APPENDIX B

CONTROL DATA 3400 COMPUTER INSTRUCTIONS

APPENDIX B

CONTROL DATA 3400 COMPUTER INSTRUCTIONS

SYMBOLS

k = Address portion of instruction
 K = k + (B^b), Modified shift count
 m = Address portion of instruction
 M = m + (B^b), Modified operand address
 y = Address portion of instruction
 Y = y + (B^b), Modified operand
 * = 48-bit instruction

b = Designator for index register
 j = Designator for 22, 23, 75, 76
 # = Restrict instruction to upper
 † = Complemented
 () = Contents of
 NI = Next instruction
 + = Floating point option

FULL WORD DATA TRANSMISSION

LDA 12 Load A (M) → A
 LAC 13 Load A complement (M)† → A
 STA 20 Store A (A) → M
 LDQ 16 Load Q (M) → Q
 LQC 17 Load Q complement (M)† → Q
 STQ 21 Store Q (Q) → M

PARTIAL WORD DATA TRANSMISSION

* LDCH63 Load character (Byte0-7) → A05-00
 (63 b v0006 50 0 m)
 * STCH63 Store character (A05-00) → Byte0-7
 (63 b v0006 50 5 m)
 SAU 60 Substitute address upper
 (A14-00) → MUA
 SAL 61 Substitute address lower
 (A14-00) → MLA

INTER-REGISTER TRANSMISSION

IAQ 00 Interchange A and Q (A) → Q, (Q) → A
 (00 7 00554)
 ATI 00 Transmit A to index (A14-00) → B^b
 (00 7 4054b)

INDEXING

ENI 50 Enter index y → B^b
 INI 51 Increase index y + (B^b) → B^b
 LIU 52 Load index upper (mUA) → B^b
 LIL 53 Load index lower (mLA) → B^b
 SIU 56 Store index upper (B^b) → mUA
 SIL 57 Store index lower (B^b) → mLA
 ISK # 54 Index skip:
 (B^b) ≠ y: (B^b) + 1 → B^b, continue
 (B^b) = y: 0 → B^b, skip lower
 instruction
 IJP 55 Index jump:
 (B^b) ≠ 0: (B^b) - 1 → B^b, jump to m
 (B^b) = 0: continue

SHIFTING

ARS 01 Shift (A) right by K
 QRS 02 Shift (Q) right by K
 LRS 03 Shift (AQ) right by K
 ALS 05 Shift (A) left by K
 QLS 06 Shift (Q) left by K
 LLS 07 Shift (AQ) left by K

ARITHMETIC

FIXED POINT

ADD 14 Add to A [(A) + (M)] → A
 SUB 15 Subtract from A [(A) - (M)] → A
 MUI 24 Multiply integer (M) (A) → QA
 DVI 25 Divide integer (QA) / (M) → A, rem Q
 RAD 70 Replace add [(M) + (A)] → M & A
 RSB 71 Replace subtract [(M) - (A)] → M & A
 RAO 72 Replace add one [(M) + 1] → M & A
 RSO 73 Replace sub. one [(M) - 1] → M & A
 SCA 34 Scale A Shift (A) left until A₄₇ ≠ A₄₆ or k = 0; (k - no. of shifts) → B^b
 SCQ 35 Scale AQ Shift (AQ) left until A₄₇ ≠ A₄₆ or k = 0; (k - no. of shifts) → B^b

FLOATING POINT

+ FAD 30 Floating add. [(A) + (M)] → A
 + FSB 31 Floating sub. [(A) - (M)] → A
 + FMU 32 Floating mult. (A) (M) → A
 + FDV 33 Floating div. (A) / (M) → A

LOGICAL

SST 40 Selective set
Set (A_n) to 1 for (M_n) = 1
SCL 41 Selective clear
Clear (A_n) for (M_n) = 1
SCM 42 Selective complement
Complement (A_n) for (M_n) = 1
SSU 43 Selective substitute
(M_n) \rightarrow A_n for (Q_n) = 1
LDL 44 Load Logical L (Q) (M) \rightarrow A
ADL 45 Add logical [(A) + L (Q) (M)] \rightarrow A
SBL 46 Sub. logical [(A) - L (Q) (M)] \rightarrow A
STL 47 Store logical L (Q) (A) \rightarrow M

INPUT/OUTPUT

* CONN 74.0 Connect
* EXTF 74.1 Function
* BEGR 74.2 Read
* BEGW 74.3 Write
* COPY 74.4 Copy status
* CLCH 74.5 Clear channel
* CCWD 74.6 Change control word

NO MEMORY REFERENCE

ENQ 04 Enter Q Extend sign Y, Y \rightarrow Q
ENA 10 Enter A Extend sign Y, Y \rightarrow A
INA 11 Increase A Extend sign Y,
Y + (A) \rightarrow A
EUB 77.5 Enter upper bound
ELB 77.6 Enter lower bound

GENERAL

INF 77.0 Internal function
AUG 77.1 Augment
CIS 77.2 Copy interrupt status
CMS 77.2 Copy mask status
SEN 77.3 Internal sense
CPR 77.4 Copy product register

MEMORY TEST

SSK# 36 Storage skip (M) neg: skip lower instruction; (M) pos: continue
SSH# 37 Storage shift (M) neg: skip lower instruction, left 1; (M) pos: continue, left 1
EQS# 64 Search (B^b) words, if ($M - 1$), or ($M - 2$), etc. = (A) skip lower instruction; \neq A, continue
THS# 65 Search (B^b) words, if ($M - 1$), or ($M - 2$), etc. > (A) skip lower instruction; \leq A, continue
MEQ# 66 Search (B^b) words, if L (Q) ($M - 1$), or ($M - 2$), etc. = (A) skip lower instruction; \neq A, continue
MTH# 67 Search (B^b) words, if L (Q) ($M - 1$), or ($M - 2$), etc. > (A) skip lower instruction; \leq A, continue

A AND Q TEST

AJP 22 Jump to m on condition j
QJP 23 Jump to m on condition j

SELECTIVE JUMP AND STOP

SLJ 75 Jump to m on condition j
SLS 76 Stop on j, and jump to m

j	22	23	75	76
0	(A) = 0: Jump	(Q) = 0: Jump	Jump	Stop: Jump
1	(A) \neq 0: Jump	(Q) \neq 0: Jump	Key 1: Jump	Key 1: Stop: Jump
2	(A) Pos: Jump	(Q) Pos: Jump	Key 2: Jump	Key 2: Stop: Jump
3	(A) Neg: Jump	(Q) Neg: Jump	Key 3: Jump	Key 3: Stop: Jump
4	(A) = 0: Ret. Jump	(Q) = 0: Ret. Jump	Ret. Jump	Stop: Ret. Jump
5	(A) \neq 0: Ret. Jump	(Q) \neq 0: Ret. Jump	Key 1: Ret. Jump	Key 1: Stop: Ret. Jump
6	(A) Pos: Ret. Jump	(Q) Pos: Ret. Jump	Key 2: Ret. Jump	Key 2: Stop: Ret. Jump
7	(A) Neg: Ret. Jump	(Q) Neg: Ret. Jump	Key 3: Ret. Jump	Key 3: Stop: Ret. Jump

BANK JUMPS

UBJP 63.0, s = 0 Jump unconditionally to Jump address M.
BJPL 63.1 Jump unconditionally to lower instruction at M.
BRTJ 63.0, s = 1 Store an unconditional Jump instruction (UBJP) at M.
Continue instruction execution at M + 1.

APPENDIX C
INSTRUCTION EXECUTION TIMES

APPENDIX C

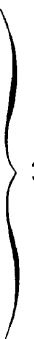


INSTRUCTION EXECUTION TIMES

The time required to execute a given instruction may vary from application to application, depending on several factors. Factors to be considered in computing execution times for a particular program are as follows:

- 1) If indirect addressing is specified, at least one additional storage reference will be needed to execute the instruction (the new index designator may itself specify indirect addressing).
- 2) If an instruction is augmented via the Augment instruction, the execution time of the augmented instruction may be decreased (i.e., un-normalized arithmetic, etc.).
- 3) In repetitive operations (e.g., Search, etc.) the number of repetitions of an operation increases execution times.

In computing the instruction execution times tabulated below, the following criteria were used:




- 1) The RNI time was assumed to be approximately 1.6 microseconds.
- 2) Times listed include the time required for instruction acquisition from storage.
- 3) Indirect addressing was not specified for any of the instructions.
- 4) The time listed for each 24-bit instruction was derived by executing a long list of that instruction with both the upper and lower instruction positions of the instruction word holding the particular instruction. After executing this list of instructions, the total elapsed time was divided by the number of instructions executed to provide an approximate time for each instruction of the list.

INSTRUCTIONS	COMMENTS	Approximate Time*
<u>Inter-Register Transmission</u> 00700554 (IAQ)		3.0
<u>Full-Word Transmission</u> 12 (LDA) 16 (LDQ) 20 (STA) 21 (STQ) 13 (LAC) 17 (LQC)		 3.0
<u>Address Transmission</u> 61 (SAL) 60 (SAU) 04 (ENQ) 10 (ENA) 53 (LIL) 52 (LIU) 57(SIL) 56 (SIU) 50 (ENI) 0074054b (ATI)		 3.0
<u>Fixed Point Arithmetic</u> 14 (ADD) 15 (SUB) 24 (MUI) 25 (DVI)		3.0 3.0 22.5 22.5
<u>Address Arithmetic</u> 11 (INA) 51 (INI) 54 (ISK)		 3.0

* Times are in microseconds

INSTRUCTIONS	COMMENTS	Approximate Time	
<u>Jumps and Stops</u>			
22 (AJP)	} Upper instruction, Jump executed	2.2*	
23 (QJP)		3.0*	
55 (IJP)		1.4*	
75 (SLJ)		3.0*	
76 (SLS)		3.0*	
63.0 (UBJP)	No Jump	3.8	
	Jump executed	2.2	
	63.0 (BRTJ)	No Jump	3.8
		Jump executed	3.8
	63.1 (BJPL)	No Jump	3.8
		Jump executed	3.0
<u>Input/Output</u>			
74.0 (CONN)	Rejected because channel is active	2.2	
	Normal (depending upon response time of external equipment).	5.8 up to 100 max.	
74.1 (EXTF)	Same as 74.0		
74.2 (BEGR)	Reject	2.2	
	No reject	3.8	
74.3 (BEGW)	Same as 74:2		
74.4 (COPY)		3.8	
74.5 (CLCH)		100	
74.6 (CCWD)	Reject	2.2	

* Add 0.8 μ sec to these times for all return Jumps executed.

INSTRUCTIONS	COMMENTS	Approximate Time
<u>Input/Output (Cont'd)</u> 77.2 (CIS) 77.2 (CMS) 77.3 (SEN) 77.4 (CPR) 77.0 (INF)		 3.0
<u>Miscellaneous</u> 77.1 (AUG) 77.5 (EUB) 77.6 (ELB) 63bv0006500m (LDCH) 63bv0006505m (STCH)		 3.0 3.8 6.1
<u>Illegal Codes</u> 00* 26 27 62 63* 74.0* → 74.6* 74.7 77.0* 77.5* 77.6* 77.7	The instructions marked with asterisks are only illegal under certain conditions. Refer to the instruction descriptions in this manual.	 3.7 (Lower) 4.5 (Upper)

APPENDIX D
NUMBER SYSTEMS

APPENDIX D

NUMBER SYSTEMS

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

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

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

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

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

BINARY NUMBER SYSTEM

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

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

The binary number 011010 represents:

$$\begin{array}{r} 0 \times 2^5 = 0 \times 32 = 0 \\ +1 \times 2^4 = 1 \times 16 = 16 \\ +1 \times 2^3 = 1 \times 8 = 8 \\ +0 \times 2^2 = 0 \times 4 = 0 \\ +1 \times 2^1 = 1 \times 2 = 2 \\ +0 \times 2^0 = 0 \times 1 = 0 \\ \hline 26_{10} \end{array}$$

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

$$2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \quad 2^{-5} \quad \dots$$

$$\text{Binary Point} = 1/2 = 1/4 = 1/8 = 1/16 = 1/32$$

The binary number 0.10110 may be represented as:

$$\begin{array}{r} 1 \times 2^{-1} = 1 \times 1/2 = 1/2 = 8/16 \\ +0 \times 2^{-2} = 0 \times 1/4 = 0 = 0 \\ +1 \times 2^{-3} = 1 \times 1/8 = 1/8 = 2/16 \\ +1 \times 2^{-4} = 1 \times 1/16 = 1/16 = 1/16 \\ \hline 11/16_{10} \end{array}$$

OCTAL NUMBER SYSTEM

The octal number system uses eight discrete symbols, 0 through 7. With the base 8 the positional value is:

$$\begin{array}{cccccccc} \dots & 8^5 & 8^4 & 8^3 & 8^2 & 8^1 & 8^0 & \\ & 32,768 & 4,096 & 512 & 64 & 8 & 1 & \end{array}$$

The octal number 513_8 represents:

$$\begin{array}{r} 5 \times 8^2 = 5 \times 64 = 320 \\ +1 \times 8^1 = 1 \times 8 = 8 \\ +3 \times 8^0 = 3 \times 1 = 3 \\ \hline 331_{10} \end{array}$$

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

$$\begin{array}{cccccccc} 8^{-1} & 8^{-2} & 8^{-3} & 8^{-4} & \dots & \\ 1/8 & 1/4 & 1/512 & 1/4096 & & \end{array}$$

The octal number 0.4520 represents:

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

ARITHMETIC

Addition and Subtraction

Binary numbers are added according to the following rules:

$$\begin{aligned}
 0 + 0 &= 0 \\
 0 + 1 &= 1 \\
 1 + 0 &= 1 \\
 1 + 1 &= 0 \text{ with a carry of } 1
 \end{aligned}$$

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

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

Subtraction may be performed as an addition:

<u>8</u> (minuend)	or	<u>+4</u> (ten's comp. of sub.)
-6 (subtrahend)		2 (difference - omit carry)
<u>2</u> (difference)		

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

One's Complement

The 3604 performs all arithmetic operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in one's complement notation.

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

For example:

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

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

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

$$\begin{aligned}
 0000 &\rightarrow 00_2 \quad \text{positive (+) zero} \\
 1111 &\rightarrow 11_2 \quad \text{negative (-) zero}
 \end{aligned}$$

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

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

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

Two's Complement

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

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

Count	Two's Comp. Rep.	One's Comp. Rep.
+2	00010	00010
+1	00001	00001
0	00000	00000
-1	11111	11110
-2	11110	11101

The difference in the representation of negative values in these two systems is due to the skipping of the all "1's" count in one's complement notation. In the one's complement system the end-around-carry feature of the register automatically changes a count of all "1's" to all "0's". (Note exception under One's Complement.)

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

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

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

Multiplication

Binary multiplication proceeds according to the following rules:

$$\begin{array}{l} 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{array}$$

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

Decimal example:

$$\begin{array}{r} \text{multiplicand} \quad 14 \\ \text{multiplier} \quad \underline{12} \\ \text{partial products} \quad 28 \\ \quad \quad \quad \underline{14} \quad (\text{shifted one place left}) \\ \text{product} \quad \quad \quad 168_{10} \end{array}$$

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

Binary example:

$$\begin{array}{r} \text{multiplicand (14)} \quad 1110 \\ \text{multiplier (12)} \quad \underline{1100} \\ \text{partial products} \quad \left\{ \begin{array}{l} 0000 \\ 0000 \\ 1110 \\ \underline{1110} \end{array} \right. \begin{array}{l} \\ \\ \text{shift to place digits} \\ \text{in proper columns} \end{array} \\ \text{product (168}_{10}\text{)} \quad 10101000_2 \end{array}$$

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

Division

The following example shows the familiar method of decimal division:

$$\begin{array}{r} \text{divisor} \quad 13 \quad \overline{)185} \\ \underline{13} \\ 55 \\ \underline{52} \\ 3 \end{array} \begin{array}{l} \text{quotient} \\ \text{dividend} \\ \\ \text{partial dividend} \\ \\ \text{remainder} \end{array}$$

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

$$\begin{array}{r} \text{divisor} \quad 1101 \quad \overline{)10111001} \\ \underline{1101} \\ 10100 \\ \underline{1101} \\ 1110 \\ \underline{1101} \\ 11 \end{array} \begin{array}{l} \text{quotient (14)} \\ \text{dividend} \\ \\ \text{partial dividend} \\ \\ \text{partial dividend} \\ \\ \text{remainder (3)} \end{array}$$

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The

computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

CONVERSIONS

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

RECOMMENDED CONVERSION PROCEDURES (INTEGER AND FRACTIONAL)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Double Dabble
Decimal to Octal	Double Dabble
Binary to Octal	Substitution
Octal to Binary	Substitution
GENERAL RULES	
* $r_i > r_f$: use Double Dabble, Substitution	
$r_i < r_f$: use Power Addition, Substitution	
r_i = Radix of initial system	
r_f = Radix of final system	

Power Addition

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

Example 1 Binary to Decimal (Integer)

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

Example 2 Binary to Decimal (Fractional)

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

Example 3 Octal to Decimal (Integer)

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

Example 4 Octal to Decimal (Fractional)

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

Double Dabble

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

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

To convert a fractional number from r_i to r_f :

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

* r refers to the radix of the number system used.

Example 1 Decimal to Binary (Integer)

$45 \div 2 = 22$ remainder 1; record 1
 $22 \div 2 = 11$ remainder 0; record 0
 $11 \div 2 = 5$ remainder 1; record 1
 $5 \div 2 = 2$ remainder 1; record 1
 $2 \div 2 = 1$ remainder 0; record 0
 $1 \div 2 = 0$ remainder 1; record 1
101101

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

Example 2 Decimal to Binary (Fractional)

$.25 \times 2 = 0.5$; record 0
 $.5 \times 2 = 1.0$; record 1
 $.0 \times 2 = 0.0$; record 0
.010

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

Example 3 Decimal to Octal (Integer)

$273 \div 8 = 34$ remainder 1; record 1
 $34 \div 8 = 4$ remainder 2; record 2
 $4 \div 8 = 0$ remainder 4; record 4
421

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

Example 4 Decimal to Octal (Fractional)

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

Thus: $.55_{10} = .431 \dots_8 .431 \dots$

Substitution

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

Example 1 Binary to Octal

Binary = 110 000 . 001 010
 Octal = 6 0 . 1 2

Example 2 Octal to Binary

Octal = 6 5 0 . 2 2 7
 Binary = 110 101 000 . 010 010 111

COMMON PURE NOTATIONS

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

POWERS OF COMMON NUMBER SYSTEMS

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

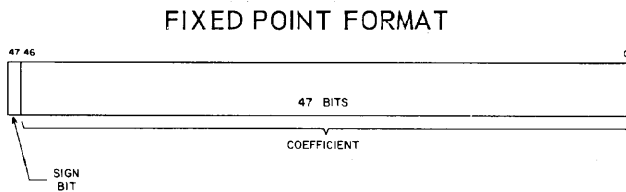
FIXED POINT AND FLOATING POINT NUMBERS

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

A fixed point number assumes:

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

A 3604 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any 3604 fixed point number designates the sign of the coefficient (47 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.



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

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of

places shifted) is expressed as the power of the base. For example, $5,100,000_{10}$ may be expressed as 0.51×10^7 , 0.051×10^8 , 0.0051×10^9 , etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are multiplied, the result is a number less than 1.

If the same two fractions are added, subtracted, or divided, the result may be greater than 1 and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

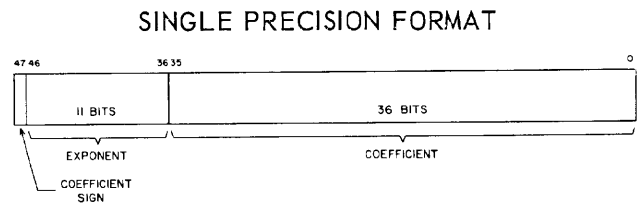
As an alternative to fixed point operation, a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

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

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

$$F \times 2^E \text{ where: } F = \text{fraction} \\ E = \text{exponent}$$

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



Coefficient

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

Exponent

The floating point exponent is expressed as an 11-bit quantity with a value ranging from 0000g to 3777g. Within this range, both positive and negative exponents must be expressed. Biasing the exponent provides the ability to distinguish between positive and negative exponents. It is formed by adding a true positive exponent and a bias of 2000g or a true negative exponent and a bias of 1777g. This results in a range of biased exponents as shown below.

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

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

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

Example 1, Number = +52

0	00 000 000	110	(36 bits)
Coefficient Sign	Exponent	Coefficient	

Number = +0.02

0	11 111 111	011	(36 bits)
Coefficient Sign	Exponent	Coefficient	

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

Example 2, Number = +52g

0	10 000 000	110	(36 bits)
Coefficient Sign	Exponent	Coefficient	

Number = +0.02g

0	01 111 111	011	(36 bits)
Coefficient Sign	Exponent	Coefficient	

Conversion Procedures

Fixed Point to Floating Point

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

Example 1 Convert +4.0 to Floating Point

- 1) The number is expressed in octal.

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

- 2) Normalize. $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$.
- 3) Since the sign of the true exponent is positive, add 2000g (bias) to the true exponent. Biased exponent = $2000 + 3$.
- 4) Assemble number in floating point format.
Coefficient = 400 000 000 000g
Biased exponent = 2003g
Assembled word = 2003 400 000 000 000g
- 5) Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Example 2 Convert -4.0 to Floating Point

- 1) The number is expressed in octal.
- 2) Normalize. $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$.
- 3) Since the sign of the true exponent is positive, add 2000g (bias) to the true exponent. Biased exponent = $2000 + 3$.
- 4) Assemble number in floating point format.
Coefficient = 400 000 000 000g
Biased exponent = 2003g
Assembled word = 2003 400 000 000 000g
- 5) Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for $-4.0 = 5774\ 377\ 777\ 777\ 777g$.

Example 3 Convert 0.5₁₀ to Floating Point

- 1) Convert to octal. $0.5_{10} = 0.4g$.
- 2) Normalize. $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$.
- 3) Since the sign of the true exponent is positive, add 2000g (bias) to the true exponent. Biased exponent = $2000 + 0$.
- 4) Assemble number in floating point format.
Coefficient = 400 000 000 000g
Biased exponent = 2000g
Assembled word = 2000 400 000 000 000g
- 5) Since the sign of the coefficient is positive, the floating point representation of +0.5₁₀ is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This

example is a special case of floating point, since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777g instead of 2000g because of the negative exponent. The 3604, however, recognizes -0 as +0 and biases the exponent by 2000g.

Example 4 Convert 0.04g to Floating Point

- 1) The number is expressed in octal.
- 2) Normalize. $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$.
- 3) Since the sign of the true exponent is negative, add 1777g (bias) to the true exponent. Biased exponent = $1777g + (-3) = 1774g$.
- 4) Assemble number in floating point format.
Coefficient = 400 000 000 000g
Biased exponent = 1774g
Assembled word = 1774 400 000 000 000g
- 5) Since the sign of the coefficient is positive, the floating point representation of 0.04g is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point

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

Example 1 Convert Floating Point Number 2003 400
000 000 000g to Fixed Octal

- 1) The floating point number is positive and remains uncomplemented.
- 2) The biased exponent $> 2000_8$, therefore subtract 2000_8 from the biased exponent to obtain the true exponent of the number. $2003 - 2000 = +3$.
- 3) Coefficient = $400\ 000\ 000\ 000_8 = .100_2$. Move binary point to the right three places. Coefficient = 100.0_2 .
- 4) The sign of the coefficient is positive because the floating point number was not complemented in step 1.
- 5) Represented in fixed octal notation. $100.0 \times 2^0 = 4.0 \times 8^0$

- 4) The sign of the coefficient is positive because the floating point number was not complemented in step 1.
- 5) Represent in fixed octal notation. $.000100_2 = .04_8$.

Example 2 Convert Floating Point Number 5774 377
777 777 777g to Fixed Octal

- 1) The sign of the coefficient is negative, therefore complement the floating point number.
Complement = $2003\ 400\ 000\ 000\ 000_8$
- 2) The biased exponent (in complemented form) is $> 2000_8$, therefore subtract 2000_8 from the biased exponent to obtain the true exponent of the number. $2003 - 2000 = +3$.
- 3) Coefficient = $400\ 000\ 000\ 000_8 = 0.100_2$
Move binary point to the right three places.
Coefficient = 100.0_2
- 4) The sign of the coefficient will be negative because the floating point number was originally complemented.
- 5) Convert to fixed octal. $-100.0_2 = -4.0_8$.

Example 3 Convert Floating Point Number 1774 400
000 000 000g to Fixed Octal

- 1) The floating point number is positive and remains uncomplemented.
- 2) The biased exponent $< 2000_8$, therefore subtract 1777_8 from the biased exponent to obtain the true exponent of the number. $1774_8 - 1777_8 = -3$.
- 3) Coefficient = $400\ 000\ 000\ 000_8 = .100_2$
Move binary point to the left three places.
Coefficient = $.000100_2$

APPENDIX E
TABLE OF POWERS OF 2

APPENDIX E

TABLE OF POWERS OF 2

2^n	n	2^{-n}																	
1	0	1.0																	
2	1	0.5																	
4	2	0.25																	
8	3	0.125																	
16	4	0.0625	5																
32	5	0.03125	25																
64	6	0.015625	625																
128	7	0.0078125	8125	5															
256	8	0.00390625	90625	25															
512	9	0.001953125	953125	125															
1024	10	0.0009765625	9765625	5															
2048	11	0.00048828125	48828125	25															
4096	12	0.000244140625	244140625	625															
8192	13	0.0001220703125	1220703125	5															
16384	14	0.00006103515625	6103515625	25															
32768	15	0.000030517578125	30517578125	125															
65536	16	0.0000152587890625	152587890625	5															
131072	17	0.00000762939453125	762939453125	25															
262144	18	0.000003814697265625	3814697265625	625															
524288	19	0.0000019073486328125	19073486328125	5															
1048576	20	0.00000095367431640625	95367431640625	25															
2097152	21	0.000000476837158203125	476837158203125	125															
4194304	22	0.0000002384185791015625	2384185791015625	5															
8388608	23	0.00000011920928955078125	11920928955078125	25															
16777216	24	0.000000059604644775390625	59604644775390625	625															
33554432	25	0.0000000298023223876953125	298023223876953125	5															
67108864	26	0.00000001490116119384765625	1490116119384765625	25															
134217728	27	0.000000007450580596923828125	7450580596923828125	125															
268435456	28	0.0000000037252902984619140625	37252902984619140625	5															
536870912	29	0.00000000186264514923095703125	186264514923095703125	25															
1073741824	30	0.000000000931322574615478515625	931322574615478515625	625															
2147483648	31	0.0000000004656612873077392578125	4656612873077392578125	5															
4294967296	32	0.00000000023283064365386962890625	23283064365386962890625	25															
8589934592	33	0.000000000116415321826934814453125	116415321826934814453125	125															
17179869184	34	0.0000000000582076609134674072265625	582076609134674072265625	5															
34359738368	35	0.00000000002910383045673370361328125	2910383045673370361328125	25															
68719476736	36	0.000000000014551915228366851806640625	14551915228366851806640625	625															
137438953472	37	0.0000000000072759576141834259033203125	72759576141834259033203125	5															
274877906944	38	0.00000000000363797880709171295166015625	363797880709171295166015625	25															
549755813888	39	0.000000000001818989403545856475830078125	1818989403545856475830078125	125															
1099511627776	40	0.0000000000009094947017729282379150390625	9094947017729282379150390625	5															
2199023255552	41	0.00000000000045474735088646411895751953125	45474735088646411895751953125	25															
4398046511104	42	0.000000000000227373675443232059478759765625	227373675443232059478759765625	625															
8796093022208	43	0.0000000000001136868377216160297393798828125	1136868377216160297393798828125	5															
17592186044416	44	0.00000000000005684341886080801486968994140625	5684341886080801486968994140625	25															
35184372088832	45	0.000000000000028421709430404007434844970703125	28421709430404007434844970703125	125															
70368744177664	46	0.0000000000000142108547152020037174224853515625	142108547152020037174224853515625	5															
140737488355328	47	0.00000000000000710542735760100185871124267578125	710542735760100185871124267578125	25															
281474976710656	48	0.000000000000003552713678800500929355621337890625	3552713678800500929355621337890625	625															
562949953421312	49	0.0000000000000017763568394002504646778106689453125	17763568394002504646778106689453125	5															

APPENDIX F
OCTAL-DECIMAL CONVERSION TABLE

APPENDIX F

OCTAL-DECIMAL CONVERSION TABLE

0000 0000
to to
0777 0511
(Octal) (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 0512
to to
1777 1023
(Octal) (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal) | 1024 to 1535 (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770								

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

4000 | 2048
to |
4777 | 2559
(Octal) | (Decimal)

Octal | Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 | 2560
to |
5777 | 3071
(Octal) | (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal) | 3072 to 3583 (Decimal)

Octal Decimal
 10000 - 4096
 20000 - 8192
 30000 - 12288
 40000 - 16384
 50000 - 20480
 60000 - 24576
 70000 - 28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal) | 3584 to 4095 (Decimal)

APPENDIX G

OCTAL-DECIMAL FRACTION CONVERSION TABLE

APPENDIX G

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont,d)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

GLOSSARY

GLOSSARY

ABSOLUTE ADDRESS	A specific storage location; contrast with relative address.
ACCESS TIME	The time needed to perform a storage reference, either read or write. In effect, the access time of a computer is one storage reference cycle.
ACCUMULATOR	A register with provisions for the addition of another quantity to its content. It is also the name of the A register.
ADDER	A device capable of forming the sum of two or more quantities.
ADDRESS	A 15-bit quantity which identifies a particular storage location.
ALPHABETIC CODING	A system of abbreviation used in preparing information for input into a computer; e. g., Q Right Shift would be QRS.
AND FUNCTION	A logical function in Boolean algebra that is 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 48-bit subtractive accumulator (modulus 248-1.)
AUGMENT	Noun: The instruction. Verb: To increase the capability of an ordinary instruction by prefacing the instruction with one of the above instructions.
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.
BOUNDS REGISTERS	Two 8-bit registers holding two 8-bit addresses which together define an upper and lower bound in storage. Operations are typically confined to the addresses within these bounds.
BRANCH	A conditional jump.
BREAKPOINT	A point in a routine at which the computer may be stopped by manual switches for a visual check of progress.
B1 - B6 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 data channel (output buffer), or a word is sent from an external equipment to storage via data channel (input buffer).

BYTE	A portion of a computer word.
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 from the typewriter. 2) A group of 12 bits which represents a column of information from the card reader.
CLEAR	A command that removes a quantity from a register by placing every stage of the register in the "0" state. The initial contents of the register are destroyed by the Clear operation.
COMMAND	A signal that performs a unit operation, such as transmitting the contents of one register to another, 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 pseudo 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 bi-stable device for storing a bit in a memory plane.
COUNTER	A register with provisions for increasing or decreasing its contents by 1.
DOUBLE PRECISION	Providing greater precision in the results of arithmetic operations by appending to the initial operands 48 additional bits (of lesser significance) of operand.
ENTER	The Enter operation is composed of two steps: a) The register is cleared, and b) The operand Y, is copied into the cleared register.
EXCLUSIVE OR	A logical function in Boolean algebra that is satisfied (has the value "1") when any of its terms are "1". It is not satisfied when all its terms are "1" or when all its terms are "0".
EXECUTION ADDRESS	The lower 15 bits of a 24 or 48-bit instruction. Most often used to specify the storage address of an operand. Sometimes used as the operand.

EXIT	Initiation of a second control sequence by the first, occurring when the first is near completion; the circuit involved in exiting.
FAULT	Operational difficulty which lights an indicator or for which interrupt may be selected.
FIXED POINT	A notation or system or arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the binary point implicitly located at some predetermined position; contrasted with floating point.
FLIP-FLOP (FF)	A bi-stable storage device. A "1" input to the set side puts the FF in the "1" state; a "1" input to the clear side puts the FF in the "0" state. The FF remains in a state indicative of its last "1" input. A stage of a register consists of a FF.
FLOATING POINT	A means of expressing a number X by a pair of numbers, Y and Z , such that $X = Yn^z$, Z is an integer called the exponent or characteristic; n is a base, usually 2 or 10; and Y is called the fraction or mantissa.
INCREASE	The increase operation adds a quantity (y or Y) to the contents of the specified register.
INDEX CODE	A 3-bit quantity in an instruction; usually specifies an index register whose contents are to be added to the execution address; sometimes specifies the conditions for executing the instructions.
INSTRUCTION	A 24 or 48-bit quantity consisting of an operation code and several other designators.
INTERRUPT	Leaving the main program routine to execute a special sequence of instructions.
INTERRUPT REGISTER	A 48-bit register whose individual bits are set to "1" by the occurrence of specific interrupt conditions, either internal or external.
INTERRUPT MASK REGISTER	A 48-bit register whose individual bits match those of the Interrupt register. Setting bits of the Interrupt Mask register to "1's" is one of the conditions for selecting interrupt.
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	The Load operation is composed of two steps. a) The register is cleared, and b) The contents of storage location M are copied into the cleared 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"

LOGICAL PRODUCT (Cont'd)	only when all the terms are "1"; otherwise it is "0". Sometimes referred to as the result of bit-by-bit multiplication.
LOGICAL SUM	In Boolean algebra, the OR function of several terms. The sum is "1" when any or all of the terms are "1"; it is "0" only when all are "0".
LOOP	Repetition of a group of instructions in a routine.
LOWER ADDRESS	The executive address portion of a lower instruction; bits 0 through 14 of a 48-bit register or storage location.
LOWER INSTRUCTION	See Program Step.
MASK	In the formation of the logical product 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 "1", the other quantity is left unaltered.
MASTER CLEAR	A general command produced by pressing one of two switches: a) Internal Master Clear - Clears all operational registers and control FFs in the 3404. b) External Master Clear - Clears all external equipments and data channels.
MNEMONIC CODE	A three of four letter code which represents the function or purpose of an instruction. Also called Alphabetic Code.
MODULUS	An integer which describes certain arithmetic characteristics of registers, especially counters and accumulators, within a digital computer. The modulus of a device is defined by r^n for an open-ended device and r^n-1 for a closed (end-around) device, where 'r' is the base of the number system used and 'n' is the number of digit positions (stages) in the device. Generally, devices with modulus r^n use two's complement arithmetic; devices with modulus r^n-1 use one's complement.
NORMALIZE	To adjust the exponent and mantissa of a floating point result so that the mantissa lies in the prescribed standard (normal) range.
NORMAL JUMP	An instruction that jumps from one sequence of instruction 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.
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.

OPERATION CODE	A 6 or 9-bit quantity in an instruction specifying the operation to be performed.
OPERATIONAL REGISTERS	Registers which are displayed on the maintenance section of the console.
OR FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") when any of its terms are "1". It is not satisfied when all terms are "0". Often called the inclusive OR function.
OVERFLOW	The capacity of a register is exceeded.
PARITY CHECK	A summation check in which the binary digits in a character are added and the sum checked against a previously computed parity digit; i. e., a check which tests whether the number of ones is odd or even.
PARTIAL ADD	An addition without carries. Accomplished by toggling each bit of the augend where the corresponding bit of the addend is a "1".
P REGISTER	The Program Address Counter (P register) 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	Either a single 48-bit instruction or two 24-bit instructions contained in one 48-bit storage addresses; the higher order 24 bits are the upper instruction; lower order 24 bits, the lower instruction. An instruction or pair of instructions is read from storage, and the upper instruction is executed first. The lower one is then executed, except when the upper one provides for skipping the lower one.
Q REGISTER	Auxiliary arithmetic register which assists the A register in the more complicated arithmetic operations (modulus 2^{48-1}).
RANDOM ACCESS	Access to storage under conditions in which the next position from which information is to be obtained is in no way dependent on the previous one.
READ	To remove a quantity from a storage location.
REJECT	A signal generated under certain circumstances by either the external equipment or the 3404 during the execution of Input/Output instructions.
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	When used 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. When replace is used in the description of an instruction, the contents of a location or register are substituted by the operand. The replace operation implies clearing the register or portion of the register in preparation for the new quantity.

REPLY	A response signal in I/O operations that indicates a positive response to some previous operation or request signal.
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.
SCALE FACTOR	One or more coefficients by which quantities are multiplied or divided so that they lie in a given range of magnitude.
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.
STAGE	The FFs and inverters associated with a bit position of a register.
STATUS	The state or condition of input/output operations.
STORE	To transmit information to a device from which the unaltered information can later be obtained. The Store operation is essentially the reverse of the Load operation. Storage location M is cleared and the contents of the register are copied into M.
SUBINSTRUCTION	In a typical 24-bit instruction, the Index code specifies one of eight forms of the instruction indicated by the Operation code. Such forms are called subinstructions.
SUBOPERATION CODE	In an instruction having several options, the Suboperation code specifies which of these options is to be performed.
TOGGLE	To complement each bit of a quantity as a result of an individual condition.
TRANSMISSION, FORCED	A transfer of bits into a register which has not been cleared previously.
TRANSMIT	The term transmit implies register contents are moved, i. e., the contents of register 1 are copied into register 2. Unless specifically stated, the contents are not changed during transmission. The term transfer is often used synonymously with transmit.
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. Holds a program step while the single 48-bit instruction or the two 24-bit instructions contained in it are executed.

UNDERFLOW	That fault occurring in exponent arithmetic when the value of the exponent formed in a floating point sum, difference, product, or quotient of two numbers is $< 2^{-10.1}$ (-17778).
UPPER ADDRESS	The execution address portion of an upper instruction; bit positions 24 through 38 of a 48-bit or storage instruction.
UPPER INSTRUCTION	See Program Step
WORD	A unit of information which has been coded for use in the computer as a series of bits. The normal word length is 48 bits.
WRITE	To enter a quantity into a storage location.

INDEX TO INSTRUCTIONS (OCTAL CODES)

<u>Octal Code</u>	<u>Operation</u>	<u>Page</u>	<u>Octal Code</u>	<u>Operation</u>	<u>Page</u>
00	Interchange A & Q	3-10	50	Enter Index	3-11
00	Transmit A to index	3-11	51	Increase Index	3-13
01	A Right Shift	3-15	52	Load Index (Upper)	3-10
02	Q Right Shift	3-15	53	Load Index (Lower)	3-11
03	Long Right Shift (AQ)	3-15	54	Index Skip	3-13
04	Enter Q	3-11	55	Index Jump	3-17
05	A Left Shift	3-15	56	Store Index (Upper)	3-11
06	Q Left Shift	3-15	57	Store Index (Lower)	3-11
07	Long Left Shift (AQ)	3-15	60	Substitute Address (Upper)	3-11
10	Enter A	3-11	61	Substitute Address (Lower)	3-11
11	Increase A	3-13	63	Load Character	3-20
12	Load A	3-10	63	Store Character	3-20
13	Load A, Complement	3-10	63.0 }	Bank Jumps	3-19
14	Add	3-11	63.1 }		
15	Subtract	3-11	64	Equality Search	3-16
16	Load Q	3-10	65	Threshold Search	3-16
17	Load Q, Complement	3-10	66	Masked Equality Search	3-16
20	Store A	3-10	67	Masked Threshold Search	3-16
21	Store Q	3-10	70	Replace Add	3-15
22	A Jump	3-17, 3-18	71	Replace Subtract	3-15
23	Q Jump	3-17, 3-18	72	Replace Add One	3-15
24	Multiply Integer	3-12	73	Replace Subtract One	3-16
25	Divide Integer	3-12	74.0	Connect	5-2
30	Floating Add	3-12	74.1	Function	5-3
31	Floating Subtract	3-12	74.2	Read	5-4
32	Floating Multiply	3-13	74.3	Write	5-4
33	Floating Divide	3-13	74.4	Copy Status	5-6
34	Scale A	3-15	74.5	Clear Channel	5-7
35	Scale AQ	3-15	74.6	Change Control Word	5-5
36	Storage Skip	3-16	75	Selective Jump	3-17, 3-19
37	Storage Shift	3-16	76	Selective Stop	3-17, 3-19
40	Selective Set	3-14	77.0	Internal Function	4-11
41	Selective Clear	3-14	77.1	Augment	3-20
42	Selective Complement	3-14	77.2	Copy Interrupt Status	4-15
43	Selective Substitute	3-14	77.3	Internal Sense	4-14
44	Load Logical	3-14	77.2	Copy Mask Status	4-15
45	Add Logical	3-14	77.4	Copy Product Register	4-15
46	Subtract Logical	3-14	77.5	Enter Upper Bound	3-22
47	Store Logical	3-14	77.6	Enter Lower Bound	3-22

CONTROL DATA SALES OFFICES

ALAMOGORDO • ALBUQUERQUE • ATLANTA • BOSTON • CAPE CANAVERAL
CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS • DAYTON
DENVER • DETROIT • DOWNEY, CALIFORNIA • HONOLULU • HOUSTON • HUNTSVILLE
ITHACA • KANSAS CITY, KANSAS • LOS ANGELES • MADISON, WISCONSIN
MINNEAPOLIS • NEWARK • NEW ORLEANS • NEW YORK CITY • OAKLAND • OMAHA
PALO ALTO • PHILADELPHIA • PHOENIX • PITTSBURGH • SACRAMENTO
SALT LAKE CITY • SAN BERNARDINO • SAN DIEGO • SEATTLE • WASHINGTON, D.C.

INTERNATIONAL OFFICES

FRANKFURT, GERMANY • HAMBURG, GERMANY • STUTTGART, GERMANY
GENEVA, SWITZERLAND • ZURICH, SWITZERLAND • CANBERRA, AUSTRALIA
MELBOURNE, AUSTRALIA • SYDNEY, AUSTRALIA • ATHENS, GREECE
LONDON, ENGLAND • OSLO, NORWAY • PARIS, FRANCE • STOCKHOLM, SWEDEN
MEXICO CITY, MEXICO, (REGAL ELECTRONICA DE MEXICO, S.A.)
OTTAWA, CANADA, (COMPUTING DEVICES OF CANADA, LIMITED) • TOKYO, JAPAN,
(C. ITOH ELECTRONIC COMPUTING SERVICE CO., LTD.)

CONTROL DATA
CORPORATION

8100 34th AVENUE SOUTH, MINNEAPOLIS, MINNESOTA 55440