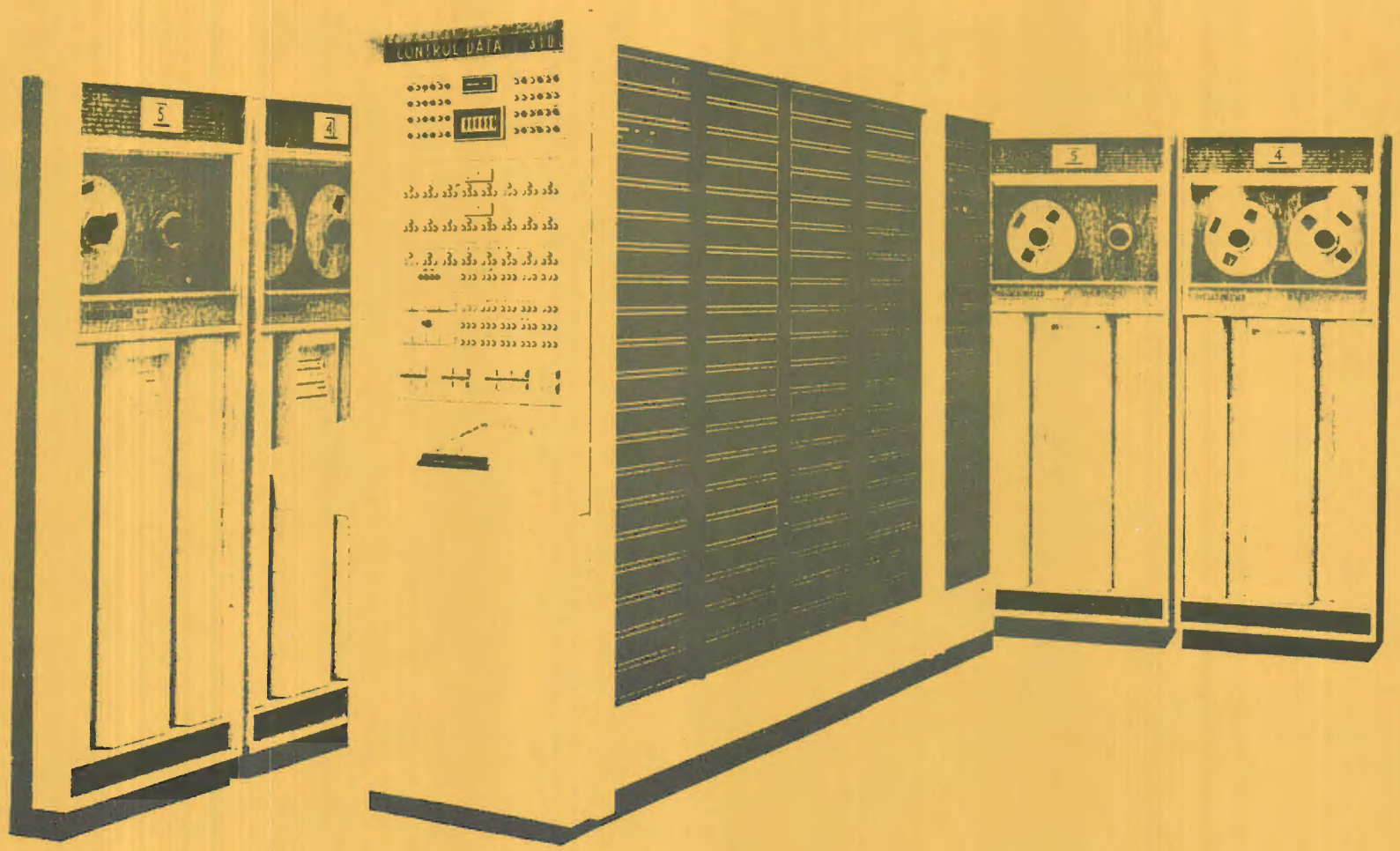
CONTROL DATA
CORPORATION

CONTROL DATA®
3100 COMPUTER SYSTEM

REFERENCE MANUAL



CONTROL DATA 3100 Computer System

CONTROL DATA
CORPORATION

CONTROL DATA®
3100 COMPUTER SYSTEM

REFERENCE MANUAL

RECORD of REVISIONS

REVISION	NOTES
A	Obsoletes previous editions.
B	Publications Change Order 11527 which incorporates extensive revisions into this manual. This
(9-27-65)	edition obsoletes previous editions.
C	Publications Change Order 15673. Pages 4-4, 7-8, 7-11, 7-12, 7-14, 7-19, 7-22, 7-28, 7-50,
(3-3-67)	7-64, 7-66, 7-68, 7-69, 7-72, 7-73, 7-74, 7-75, 7-76, 7-78, 7-79, C-4, C-7, 11, 12, 17, 18
	and 20 revised.
D	Publication Change Order 18404. Pages 2-2, 7-26, 7-65, C-11, C-12, C-13 and Comment Sheet
(1-15-68)	revised.

FORM CA230 REV. 1-67

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

Pub No. 60108400
© 1965, 1967, 1968
by Control Data Corporation
Printed in United States of America

CONTENTS

Section 1. SYSTEM DESCRIPTION

INTRODUCTION 1-1

COMPUTER MODULARITY 1-1
- Main Control and Arithmetic Module 1-3
- Block Control and Interrupt Module 1-3
- Storage Module 1-3
- Input/Output Sub-Modules 1-3
- Optional Arithmetic Module 1-4
- Consoles 1-4
- Power Control Panel 1-7

3104 COMPUTER 1-7

COMPUTER ORGANIZATION 1-7
- Computer Word Format 1-7
- Register Descriptions 1-7
- Data Bus and 'S' Bus 1-11
- Block Control 1-11
- Integrated Register File 1-12
- Real-Time Clock 1-12
- Parity 1-13

PERIPHERAL EQUIPMENT 1-14

Section 2. STORAGE CHARACTERISTICS

STORAGE MODULE CONTROL PANEL 2-1

STORAGE REGISTERS 2-2
- S Register 2-2
- Z Register 2-2

READ/WRITE CHARACTERISTICS 2-2
- Single-Character Mode 2-2
- Double-Character Mode 2-2
- Triple-Character Mode 2-2
- Full-Word Mode 2-2
- Address Mode 2-2

STORAGE ADDRESSING 2-3

STORAGE SHARING 2-3

STORAGE PROTECTION 2-3
- Permanent Protection 2-4
- Selective Protection 2-4
- No Protection 2-4

Section 3. INPUT/OUTPUT CHARACTERISTICS

INTERFACE SIGNALS 3-1

I/O PARITY 3-2
- Parity Checking with the 3106 3-2
- Parity Checking with the 3107 3-2

AUTO LOAD/AUTO DUMP 3-3
Preliminary Considerations 3-3
Auto Load 3-3
Auto Dump 3-3
3100 Auto Load/Auto Dump Interim Subroutine 3-3

SATELLITE CONFIGURATIONS 3-5

Section 4. INTERRUPT SYSTEM

GENERAL INFORMATION 4-1

INTERRUPT CONDITIONS 4-1
Internal Interrupts 4-1
Trapped Instruction Interrupts 4-2
Power Failure Interrupt 4-2
I/O Interrupts 4-3

INTERRUPT MASK REGISTER 4-3

INTERRUPT CONTROL 4-3
Enabling or Disabling Interrupt Control 4-4
Interrupt Priority 4-4
Sensing Interrupts 4-4
Clearing Interrupts 4-4

INTERRUPT PROCESSING 4-5

Section 5. CONSOLES AND POWER CONTROL PANEL

CONSOLES 5-1
Register Displays 5-1
Console Loudspeaker 5-4
Status Indicators 5-4
Switches 5-7

POWER CONTROL PANEL 5-15
Switches 5-15
Elapsed Time Meters 5-15

Section 6. TYPEWRITER

DESCRIPTION 6-1

OPERATION 6-2
Set Tabs, Margins, and Spacing 6-2
Clear 6-2
Status Checking 6-2
Type In and Type Load 6-3
Type Out and Type Dump 6-3

CONSOLE SWITCHES AND INDICATORS 6-3

CHARACTER CODES 6-5

Section 7. INSTRUCTIONS

GENERAL INFORMATION 7-1
Instruction Word Formats 7-1

Word Addresses vs. Character Addresses 7-2
Symbol Definitions 7-3
Indexing and Address Modification 7-3
Addressing Modes 7-4
Indexing and Addressing Mode Examples 7-5
I/O Channel Considerations 7-6
Trapped Instructions 7-6

INSTRUCTION LIST 7-7
Register Operations without Storage Reference 7-12
Load 7-20
Store 7-23
Inter-register Transfer, 24-bit Precision 7-26
Inter-register Transfer, 48-bit Precision 7-29
Stops and Jumps 7-30
Logical Instructions with Storage Reference 7-37
Arithmetic, Fixed Point, 24-bit Precision 7-38
Arithmetic, Fixed Point, 48-bit Precision 7-40
Arithmetic, Floating Point 7-43
BCD 7-46
Storage Shift, Searches, Compare and Register Shifts 7-50
Search 7-56
Move 7-58
Sensing 7-60
Control 7-63
Interrupt 7-65
Input/Output 7-68

Section 8. SOFTWARE SYSTEMS

GENERAL DESCRIPTION 8-1
3100, 3200, 3300 SCOPE 8-1
3100, 3200, 3300 COMPASS 8-2
3100, 3200, 3300 Data Processing Package 8-3
3100, 3200, 3300 Utility 8-4
3100, 3200, 3300 COBOL 8-4
3100, 3200, 3300 FORTRAN 8-5
Generalized Sort/Merge Program 8-5
3100, 3200, 3300 BASIC System 8-6

CODING PROCEDURES 8-7
Instruction Format 8-7
Pseudo-Instructions 8-9
Assembly Listing Format 8-17
Error Codes 8-18

APPENDIX A – Control Data 3100, 3200, 3300 Computer Systems Character Set
B – Supplementary Arithmetic Information
C – Programming Reference Tables and Conversion Information

GLOSSARY, INSTRUCTION TABLES AND INDEX

FIGURES

FIGURE

1-1 3100 Modular Configuration 1-2
1-2 3101 Console 1-5
1-3 3200 Console 1-6
1-4 Computer Word Character Positions and Bit Assignments 1-7
1-5 Storage Addressing and Data Paths of Typical Installation 1-11
1-6 Parity Bit Assignments 1-13
2-1 Storage Module Control Panel 2-1
3-1 Principal Signals Between I/O Channel and External Equipment 3-1
3-2 Satellite Configurations 3-5
5-1 Front View of Integrated Console 5-2
5-2 $E_U E_L$ Register Display 5-3
5-3 E_D Register Display 5-3
5-4 External Status Indicators 5-4
5-5 Internal Status Indicators 5-5
5-6 Temperature Warning Designations for an Expanded 3100 Computer, Front View 5-6
5-7 Console Keyboard 5-8
5-8 Breakpoint Switch Examples 5-13
5-9 Power Control Panel 5-16
6-1 3192 Console Typewriter 6-1
6-2 Typewriter Control Panel 6-3
7-1 Word-Addressed Instruction Format 7-1
7-2 Character-Addressed Instruction Format 7-2
7-3 Indexing and Indirect Addressing Routine Flow Chart 7-3
7-4 Operand Formats and Bit Allocations for MUAQ and DVAQ Instructions 7-41
7-5 Operand Formats and Bit Allocations for Floating Point Arithmetic Instructions 7-45
7-6 Search Operation 7-57
7-7 Move Operation 7-59
7-8 73 I/O Operation with Storage 7-73
7-9 74 I/O Operation with Storage 7-75
7-10 75 I/O Operation with Storage 7-77
7-11 76 I/O Operation with Storage 7-79
7-12 73 I/O Operation with A 7-81
7-13 74 I/O Operation with A 7-83
7-14 75 I/O Operation with A 7-85
7-15 76 I/O Operation with A 7-87
8-1 COMPASS Coding Form 8-19
8-2 FORTRAN Coding Form 8-19

TABLES

TABLE

1-1 Optional Memory Configurations 1-3
1-2 Characteristics of 3100 Computer Registers 1-10
1-3 Register File Assignments 1-13
2-1 Absolute Addresses 2-3
2-2 Auto Load/Auto Dump Reserved Addresses 2-4
2-3 Storage Protection Switch Descriptions 2-5
2-4 Storage Protection Switch Settings 2-5
4-1 Interrupt Mask Register Bit Assignments 4-3
4-2 Interrupt Priority 4-4
4-3 Representative Interrupt Codes 4-5
5-1 Keyboard Switch Functions 5-9
5-2 Console Main-Frame Switches 5-10
5-3 Power Control Panel Switch Functions 5-15
6-1 Console Typewriter Switches and Indicators 6-4
6-2 Console Typewriter Codes 6-5
7-1 List of Trapped Instructions 7-7
7-2 Instruction Synopsis and Index 7-8
7-3 Summary of Instruction Execution Times 7-11
7-4 Interrupt Mask Register Bit Assignments 7-61
7-4a Bit Assignments for Interrupt Sensing Conditions 7-61
7-5 Internal Status Sensing Mask 7-62
7-6 Block Control Clearing Mask 7-63
7-7 Pause Sensing Mask 7-64
7-8 Interrupt Mask Register Bit Assignments 7-65
7-9 Modified I/O Instruction Words 7-69
8-1 Instruction Interpretations 8-8
8-2 COMPASS Coding Form Description 8-18

Section 1
SYSTEM DESCRIPTION

INTRODUCTION

The CONTROL DATA* 3100 is a medium-sized, solid-state, general-purpose digital computing system. Advanced design techniques are used throughout the system to provide expedient solutions for scientific, real-time, and data processing problems. Modular packaging facilitates expansion of the basic 3100 System to accommodate increasing customer needs.

The 3100 is upwards compatible with the CONTROL DATA 3200 and 3300 Computer Systems; i.e., as computation requirements exceed the capabilities of the 3100 System, the user may escalate to a 3200 or 3300 System without the necessity to revise existing 3100 programs. Its input/output characteristics are identical to the 3200, 3300, 3400, 3600 and 3800 Computer Systems—a fact which facilitates incorporating the 3100 into a SATELLITE* configuration.

Various software systems are available for the 3100 System. The SCOPE operating system is used in 3100 Systems to provide efficient job processing. SCOPE requires a minimum of storage and time requirements. COMPASS, operating under the control of SCOPE, is the assembly system used to assemble relocatable machine language programs. Other applicable software includes FORTRAN, COBOL, the Data Processing Package, Generalized Sort/Merge and Basic System. These systems are described in the Software Section of this manual. Other software and hardware publications pertinent to 3100 Systems may be obtained from the nearest Control Data sales office listed on the back cover of this manual.

A wide selection of peripheral equipment is available for use in a 3100 System. Equipment that is applicable for 3100 Systems may be found in the 3000 Series Computer System Peripheral Equipment Reference Manual, publication number 60108800.

This manual provides programming and operating information in conjunction with a description of special features of the 3100. Reference information and supplementary information may be found in the Appendix section.

COMPUTER MODULARITY

A 3100 Computer consists of various logic cabinet modules designed to perform specific operations. If additional storage, input/output channels, or arithmetic capabilities are desired for an existing installation, an appropriate module is integrated into the system. The 3104 (Figure 1-1), described later in this section, constitutes the basic 3100 modular configuration.

*Registered trademarks of Control Data Corporation

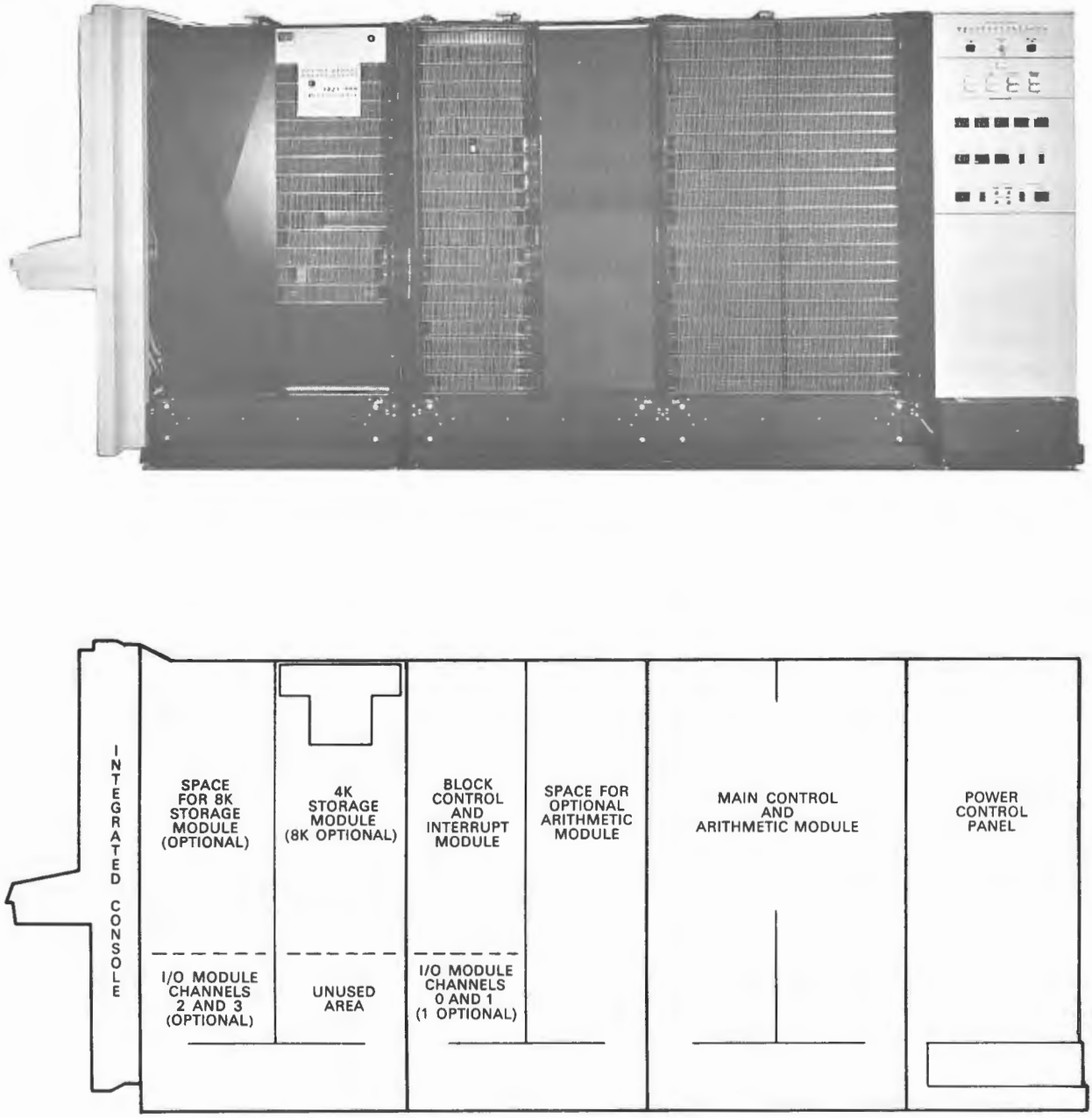


Figure 1-1. 3100 Modular Configuration

MAIN CONTROL AND ARITHMETIC MODULE

This module, standard in all 3100 systems, controls internal operations, executes 24-bit precision fixed point arithmetic and 48-bit precision fixed point addition and subtraction instructions. Boolean, character word processing, and decision operations are also processed by this module. Floating point, BCD, and 48-bit precision multiplication and division instructions are classified as trapped instructions if the optional arithmetic module is absent from the system. Trapped instructions may be processed under control of an interpretive software routine.

BLOCK CONTROL AND INTERRUPT MODULE

Logic associated with this module controls Search and Move operations, external equipment and typewriter I/O, real-time referencing, and operations with the register file. Interrupt logic, also located in this module, processes Internal, I/O, Trapped Instruction, and Power Failure interrupts.

STORAGE MODULE

A unique 4,096-word memory module is standard in every 3100 System. A customer may select combinations of magnetic core storage (MCS) modules to increase the total storage capacity of his computer system to 8,192; 16,384 or 32,768 words. The following optional storage modules are available:

- **3108 — 4,096-word (16,384 characters) MCS memory expansion unit. This unit is used only to expand the standard 4,096 word memory to 8,192 words.**
- **3109 — 8,192-word (32,768 characters) MCS memory module (requires additional chassis).**
- **3103 — 16,384-word (65,536 characters) MCS memory module (requires additional chassis).**

Memory configurations are shown in Table 1-1.

TABLE 1-1. OPTIONAL MEMORY CONFIGURATIONS

Total Expanded Memory Capacity	Memory Modules Required in Addition to 4K Memory in 3104
8K	3108
16K	3108 and 3109
32K	3108, 3109 and 3103

INPUT/OUTPUT SUB-MODULES

Two types of I/O Channels are available:

3106 Communication Channel (12-bit)
3107 Communication Channel (24-bit)

3106

The 3106 is a bidirectional 12-bit, parallel data channel. A maximum of four 3106 channels may be used in a 3100 System and up to eight peripheral controllers may be connected to each channel. Cabinet space is provided for mounting two 3106 channels per I/O module.

3107

The 3107 is a bidirectional 24-bit, parallel data channel with twice the data transfer rate of the standard 3106 I/O channel. One 3107 occupies the same cabinet space required for two 3106 channels. If a 3107 is installed in a system, the maximum number of 3106 channels is limited to two and the 3107 is installed in the space reserved for channels 2 and 3.

OPTIONAL ARITHMETIC MODULE

The floating point/48-bit precision standard arithmetic option No. 10018 provides the necessary logic to execute 36-bit precision coefficient floating point arithmetic. It also permits the 48-bit precision multiply and divide instructions to be executed directly by the hardware.

The BCD standard arithmetic option No. 10019 permits decimal numbers to be added, subtracted, loaded, stored or sensed directly without the use of interpretive software.

When both standard arithmetic options are incorporated into a system, it is referred to as the 3100 General Arithmetic Option No. 10020. If one or both options are absent, the instructions pertaining to that option(s) can be executed by entering a trapped routine and utilizing the appropriate software.

If either or both arithmetic options are present in a system, they are contained in chassis 4 which is normally located between chassis 1 and 2.

CONSOLES

A choice of three consoles is available for use in 3100 Computer Systems. Two consoles are provided with binary displays and identical programming switches. A third, the standard 3200 console, provides octal register readouts.

Standard Integrated Console

The integrated console consists of an upright control panel and display assembly mounted on the end of the main computer frame. This console features binary displays, status indicators, programming switches, monitor loudspeaker, and a removable keyboard for remote operation. A CONTROL DATA 3192 On-Line Typewriter is compatible for systems using an integrated console. Refer to the Consoles and Power Control Panel section for a more detailed description of this console.

3101

The optional 3101 Desk Console is electrically identical to and replaces the integrated console. It features a special display panel for status conditions and binary register representation. The entry keyboard and on-line typewriter are integral parts of this console. A chair is provided for the computer operator. Refer to Figure 1-2.

3200 Console

The 3200 Console may be used in 3100 Systems to provide octal register displays. This console is referred to as Standard Option No. 10013 when used in a 3100 System. If the system incorporates the BCD arithmetic option, the E_D register is displayed by decimal digits with visual plus and minus signs. An operator's chair is also provided with this console. A full view of a 3200 Console appears in Figure 1-3.

Figure 1-2. 3101 Console

Figure 1-3. 3200 Console

POWER CONTROL PANEL

The Power Control Panel enables the computer operator to initially connect power to the main computer, typewriter, and groups of peripheral equipment. Semipermanent storage protection switches are located on the upper section of this panel. Operating time and maintenance time meters and the main equipment circuit breakers are also mounted on the control panel. Detailed information pertaining to the Power Control Panel appears in Section 5.

3104 COMPUTER

The 3104 is the basic 3100 Computer configuration. A 3104 consists of a Main Control and Arithmetic module, Block Control and Interrupt module, single channel I/O module, 4,096 Word Storage module, Integrated Console, and a Power Control Panel. The basic 3104 is expanded by adding I/O channels, adding arithmetic option(s), and/or increasing the system's magnetic core storage capacity to a maximum of 32,768 words. Either desk console may be substituted for the Integrated Console.

COMPUTER ORGANIZATION

COMPUTER WORD FORMAT

The standard 3100 computer word consists of 24 binary digits. Each word is divided into four 6-bit characters. In storage, an odd parity bit is generated and checked for each of the four characters, lengthening the storage word to 28 bits. Figure 1-4 illustrates the bit assignments of a computer word in storage.

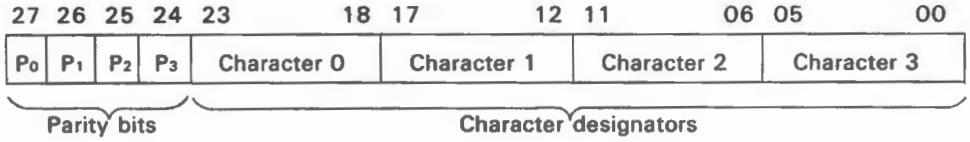


Figure 1-4. Computer Word Character Positions and Bit Assignments

REGISTER DESCRIPTIONS

A Register *(Arithmetic)*

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

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

Q Register *(Arithmetic)*

The Q register is an auxiliary register and is generally used in conjunction with the A register.

The principal functions of Q are:

- **Providing temporary storage for the contents of A while A is used for another arithmetic operation.**
- **Forming a double-length register, AQ.**
- **Shifting to the right or left, separately or in conjunction with A.**
- **Serving as a mask register for 06, 07, and 27 instructions.**

Both A and Q may load or be loaded from any of the three index registers without the use of storage references.

X Register *(Arithmetic)*

The X register is a transfer register, used only for internal instruction processing. Contents of this register cannot be displayed by any external indicators.

F Register *(Main Control)*

The program control register, F, holds an instruction during the time it is being executed. During execution, the program may modify the instruction in one of three ways:

- *Indexing* (Address Modification)—A quantity in one of the index registers (B^b) is added to the lower 15 bits of F for word-addressed instructions, or to the lower 17 bits of F for character-addressed instructions. The signs of B^b and F are extended for the addition process.
- *Indirect Addressing*—The lower 18 bits of F are replaced by new a, b, and m designators from the original address M (modified if necessary, $M=m+B^b$).
- *Indirect Addressing (load and store index instructions)*—Bits 00-14 and 17 of F are replaced by new a and m designators from the original address M (no modification possible).

After executing an instruction, a Normal Exit, Skip Exit or Jump Exit is performed. F is displayed on the console whenever the keyboard is inactive and the computer is not in the GO mode.

C Register *(Main Control)*

Quantities to be entered into the A, Q, B or P registers or into storage from the entry keyboard are temporarily held in the Communication (C) register until the TRANSFER switch is pushed. If an error is made while entering data into the Communications register, the KEYBOARD CLEAR switch may be used to clear this register.

The C register holds words read from storage during a Sweep or Read Storage operation. The contents of C are displayed on the console whenever the keyboard is active.

P Register *(Main Control)*

The P register is the Program Address Counter. It provides program continuity by generating in sequence the storage addresses which contain the individual instructions. During a Normal Exit the count in P is incremented by 1 at the completion of each instruction to specify the address of the next instruction. These addresses are sent via the S (address) Bus to the specified storage module where the instruction is read. A Skip Exit advances the count in P by 2, bypassing the next sequential instruction and executing the following one. For a Jump Exit, the execution address portion of the jump instruction is entered into P, and used to specify the starting address of a new sequence of instructions.

B^b Registers *(Main Control)*

The three index registers, B^1, B^2 and B^3, are used in a variety of ways, depending on the instruction. In a majority of the instructions they hold quantities to be added to the execution address ($M=m+B^b$). The B^b registers have no provision for arithmetic operations.

Data Bus Register *(DBR-Main Control)*

A 24-bit Data Bus register is used to temporarily hold the data received from storage, the Communication register and other logic areas. It is a nondisplayed and nonaddressable register.

During character-addressed or input/output operations, data entering the DBR may be shifted one, two, or three character positions during the transfer to reach the correct character position within the DBR.

E Register

The optional arithmetic register, E, is present in a system whenever one of the two optional arithmetic logic packages is present. Its characteristics and functions depend upon whether it is being used for floating point/48-bit precision or for BCD operations.

During floating point/48-bit precision operations, the E register is divided into two parts: E_U and E_L (E_{Upper} and E_{Lower}), each composed of 24 bits. It is used as follows:

- 48-bit precision multiplication; holds the lower 48 bits of a 96-bit product.
- 48-bit precision division; initially holds the lower 48 bits of the dividend; upon completion, holds the remainder.
- Floating point multiplication; holds the residue of the coefficient of the 48-bit product.
- Floating point division; holds the remainder.

During BCD operations the E register is designated the E_D register ($E_{Decimal}$). The unique decimaldigits can be expressed in 4 bits, i.e., $8_{10}=10_8$ and $9_{10}=11_8$. Accordingly, E_D is extended from 48 to 53 bits in order to handle 13 of these 4-bit characters, plus one sign bit. This register is used in conjunction with storage to perform BCD addition and subtraction.

D Register

The D register is a field length register and is used in conjunction with loading, storing, adding, and subtracting numeric BCD characters. This register is set to a field length of 1 to 12 characters by executing a SET (70.7) instruction. The field length remains the same until it is changed by another SET instruction.

The D register is present only when the BCD arithmetic option is incorporated into a system. The contents of the D register cannot be displayed.

S Register *(Storage)*

The S register holds the address of the storage word currently being referenced. It is displayed on the Storage Module control panel.

Z Register *(Storage)*

The Z register is the Storage Resoration and Modification register. Data stored or being transferred to or from the address specified by the S register must pass through Z. The entire storage word including the four parity bits is represented by the Z register and is displayed on the Storage Module control panel.

Program requests are produced as the direct result of a programmed instruction. An instruction of this type may perform the following:

1. Initiate the processing of a block of data,
2. Terminate the processing of a block of data,
3. Direct inter-register transfers in conjunction with the Register File,
4. Transmit Connect or Function commands to I/O channels.

Any request to Block Control, except for Connect or Function, results in one or more Register File references. Block operation requests process blocks of data as directed by the initial instruction.

Priority access is established for all Block Control operations with one exception: program requests have either first or last priority. Last priority is granted only if a channel request occurs almost simultaneously with the program request. The order of priority, with program request listed first, is as follows:

1. Program request.
2. Channel 0 request.
3. Channel 1 request.
4. Channel 2 request.
5. Channel 3 request.
6. Real-time clock request.
7. Typewriter request or Read-Store or Write-Store with Register File request.
8. Search or Move request.

Every Block Control cycle normally completes the operation specified by the request. A Move operation is the only possible exception. Due to the length of time required for the Move operation cycle, it may be terminated in the middle of its cycle by a Channel request. The aborted Move operation cycle results in no loss of data. The next complete Move operation cycle moves the word or character of the aborted cycle.

INTEGRATED REGISTER FILE

The Integrated Register File is a 64_{10} word (24 bits per word) memory integrated into the upper 64 addresses of storage. Although the programmer has access to all registers in the file with the 53 instruction, certain registers are reserved for specific purposes (see Table 1-3). All reserved registers may be used for temporary storage if their use will not disrupt other operations that are in progress. The contents of any register in the file may be examined by transferring them to the A register. Register 77 corresponds to the uppermost storage address.

REAL-TIME CLOCK

The real-time clock is a 24-bit counter that is incremented each millisecond to a maximum period of 16,777,216* milliseconds. After reaching its maximum count, the clock returns to zero and the cycle is repeated continuously. The clock, which is controlled by a 1 kilocycle signal, starts as soon as power is applied to the computer. The current time is stored in register 22 of the Register File. It is removed from storage, updated, and compared with the contents of register 32 once each millisecond. When the clock time equals the time specified by the clock mask, an interrupt is set. When necessary, the real-time clock may be reset to any 24-bit quantity including zero by loading A and then transferring (A) into register 22.

*16,777,216 milliseconds equals approximately 4 hours and 40 minutes.

TABLE 1-3. REGISTER FILE ASSIGNMENTS

Register Numbers	Register Function
00-03	Modified I/O instruction word containing the current character address (channel 0-3 control)
04-07	Temporary storage
10-13	Modified I/O instruction word containing the last character address ±1, depending on the instruction (channel 0-3 control)
14-17	Temporary storage
20	Search instruction word containing the current character address (search control)
21	Move instruction word containing the source character address (move control)
22	Real-time clock, current time
23	Current character address (typewriter control)*
24-27	Temporary storage
30	Instruction word containing the last character address +1 (search control)
31	Instruction word containing the destination character address (move control)
32	Real-time clock, interrupt mask
33	Last character address +1 (typewriter control)*
34-77	Temporary storage

PARITY

Parity bits are generated and checked in 3100 systems for the following two conditions:

1. **Whenever a data word is read from or written into storage.**
2. **When a data word is transferred via an I/O channel.**

Storage Parity

A parity bit is generated and checked for each 6-bit character of a storage word. Refer to Figure 1-6.

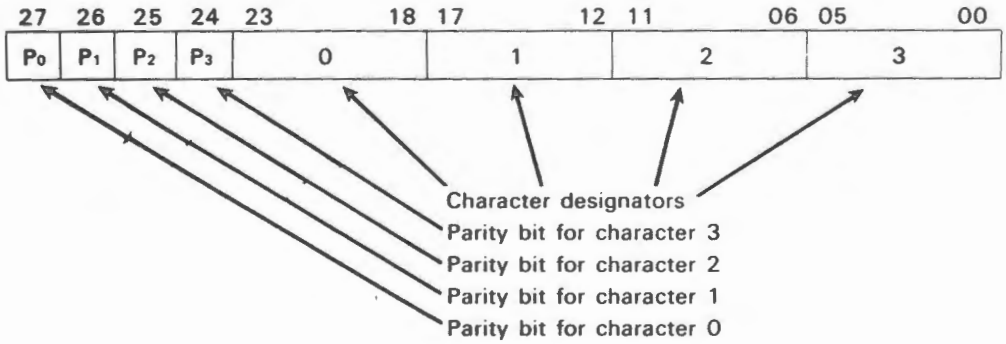


Figure 1-6. Parity Bit Assignments

*The upper 7 bits of registers 23 and 33 should contain zeros.

During each Write cycle, a parity bit is stored along with each character. When part or all of a word is read from storage, parity is checked for a loss or gain of bits. Failure to produce the correct parity during Read operations causes the PARITY FAULT indicators on the Storage Module Control Panel and internal status lights to glow. As soon as a parity error is recognized by Main Control, program execution is halted. Master Clearing the computer clears the fault condition.

If the DISABLE PARITY switch has been depressed and is active, subsequent parity errors will not cause parity error indications to glow and program execution will not be affected.

The total number of "1's" in a character, plus the parity bit, is always an odd number in the odd parity system used in the 3100.

I/O Parity

The I/O Communication Channels provide parity lines in addition to the other signals that interface with external equipment. Parity is checked in the I/O channels to detect parity errors during data transmission to the external equipment and errors when data is received from external equipment. I/O parity errors can be detected by a sensing instruction; however, the parity error indicator will not be activated. A complete description of I/O parity generation and checking may be found in the I/O section of this manual.

PERIPHERAL EQUIPMENT

A large variety of peripheral equipment is available for use with the 3100 computer. All peripheral equipment available for 3100, 3200, 3300, 3400, 3600 and 3800 systems may be attached to a 3106 communication channel. For programming instructions, as well as a list of function codes and status response codes, refer to the Control Data 3000 Series Computer Systems Peripheral Equipment Reference Manual, publication number 60108800.

Section 2
STORAGE CHARACTERISTICS

STORAGE MODULE CONTROL PANEL

Figure 2-1 shows the Storage Control Panel which is mounted at the top of each 3109 Storage Module. The Drive Voltage Control is used to adjust the meter reading to zero percent (22.5 volts). The Z and S registers are displayed on this panel, as well as three storage faults. The indicator lamps represent an x or y drive line voltage failure and a storage parity fault. The Control Panel on the 3103 Module is similar to the 3109 Control Panel but is laid out on a vertical plane.

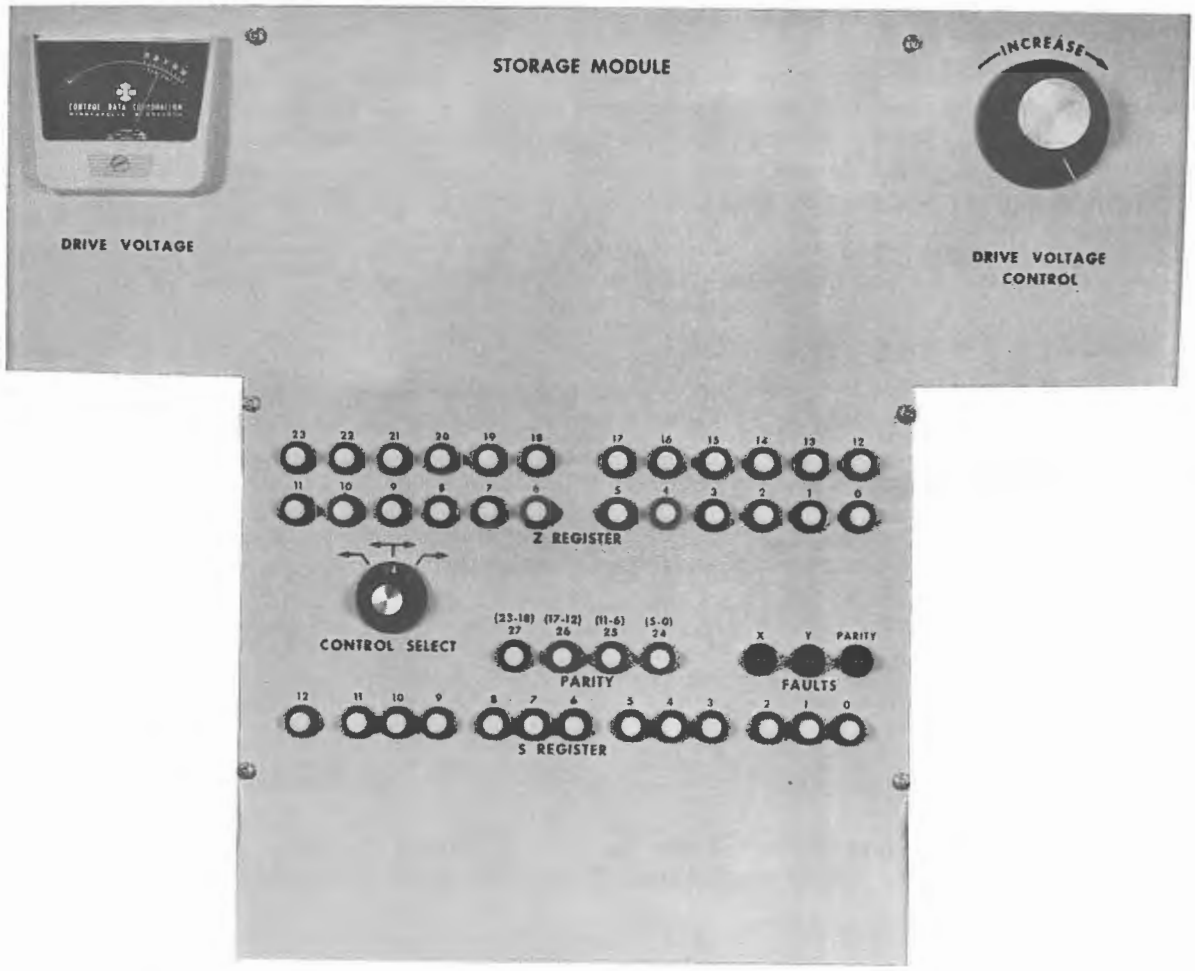


Figure 2-1. Storage Module Control Panel

STORAGE REGISTERS

S REGISTER

The 13-bit S register contains the address of the word being currently processed. Bit 12 specifies field 0 or field 1 in the memory stack. Bits 00-11 specify the co-ordinates of the word.

Z REGISTER

The 28-bit Z register is the storage restoration and modification register. All data that is transferred to or from the storage module passes through Z.

READ/WRITE CHARACTERISTICS

During a normal memory cycle, all bits of a word referenced by (S)* are read out of core storage in parallel, loaded into Z, used for some purpose, then written back into storage, intact. Five modes exist in the 3100 Computer for storage modification. In all cases, Z is initially in the cleared state.

The Z register is only cleared at the beginning of each memory cycle (except in the case of a Master Clear). If the program stops as the result of a parity error, the operator can examine (Z) on the Storage Module Control Panel, Figure 2-1.

SINGLE-CHARACTER MODE

Any one character may be ignored during the Read cycle. New data is then loaded into the corresponding character position of Z and the whole (Z) is stored.

DOUBLE-CHARACTER MODE**

The upper, middle, or lower half of a word is ignored during the Read cycle. New data is loaded into the unfilled half of Z and the whole (Z) is stored.

TRIPLE-CHARACTER MODE**

Either of the two possible triple-character groups may be ignored during the Read cycle. New data is then loaded into the corresponding character positions of Z and the whole (Z) is stored.

FULL-WORD MODE

The whole word is ignored during the Read cycle. A new word is entered into Z and (Z) is stored.

ADDRESS MODE

The lower 15 or 17 bits of a word may be ignored during the Read cycle. A new word or character address is then loaded into Z, and the whole (Z) is stored.

---

*The parentheses are an accepted method for expressing the words "the content(s) of " (in this case, "the contents of S").

**Except when Double Character mode is used during buffered I/O, these two modes can only be used if the BCD option is present.

STORAGE ADDRESSING

Table 2-1 gives the absolute addresses for a specific storage capacity.

TABLE 2-1. ABSOLUTE ADDRESSES

Storage Word Capacity	Encompassing Addresses
4K (4,096)	00000 → 07777
8K (8,192)	ALL PRECEDING ADDRESSES AND: 10000 → 17777
16K (16,384)	ALL PRECEDING ADDRESSES AND: 20000 → 37777
32K (32,768)	ALL PRECEDING ADDRESSES AND: 40000 → 77777

NOTE

If an address is referenced that exceeds the storage capacity of a system, the uppermost digit is adjusted to conform to the available storage. No fault indication is given for this case.

Example: Address 67344 referenced.

Actual address referenced: 67344 — 32K system
27344 — 16K system
07344 — 8K system
07344 — 4K system

STORAGE SHARING

Two 3100 Computers may share the memory of a 3109 Storage Module. A switch on each Storage Module Control Panel allows the operator to give exclusive control to the right or left computer. A middle position on this switch actuates a two-position priority scanner. Storage Control honors the requests in the order they are received. Neither computer has priority over the other, and the computer involved in the current storage cycle relinquishes control to the requesting computer at the end of its cycle. Either computer can therefore be delayed a maximum of one storage cycle. A similar program delay may occur within either computer when an internal scanner determines whether Main Control or Block Control has access to the storage module.

Direct access to 3100 type storage modules is available for certain installations. The normal I/O channel route is bypassed and the customer's special equipment interfaces directly with the storage logic.

STORAGE PROTECTION

It is often desirable to protect the contents of certain storage addresses against alteration during the execution of a program. There are three catagories of addresses: those that are always protected; those that are protected at the option of the programmer; and those that are never protected during special sequences.

An attempt to write at a protected address is defined as an Illegal Write. No writing actually takes place, however, and the attempt to write does not stop or interrupt the execution of the program. An Illegal Write causes a console indicator to light and the program may sense an Illegal Write as bit 05 of the internal status response code. An Illegal Write is cleared by a Master Clear, an Internal Clear, or by sensing.

PERMANENT PROTECTION

The upper 100_8 addresses are used for the Register File and are not protected and should not be protected to insure proper operation of the Register File. The next lower 40_8 memory locations, which contain the program for an Auto Load or Auto Dump operation, are always protected against alteration by a special storage protection circuit. The 40_8 addresses protected depend upon the memory size and encompass the addresses shown in Table 2-2.

TABLE 2-2. AUTO LOAD/AUTO DUMP RESERVED ADDRESSES

Memory Size	Auto Load and Auto Dump Reserved Storage Addresses
4K	07640-07677
8K	17640-17677
16K	37640-37677
32K	77640-77677

Logic circuits sense the total storage capacity of the system and check each storage address as it appears on the S (address) Bus to see if it is among the protected addresses. If it is one of those to be protected, reading but no writing is allowed at that address. The only time that this protection is disabled is when an operator presses the ENTER AUTO PROGRAM switch on the console so that he may store a new Auto Load or Auto Dump program. Refer to Section 3, Input/Output Characteristics, for additional information on the Auto Load and Auto Dump features.

SELECTIVE PROTECTION

There are 15 three-position toggle switches mounted on the Power Control Panel. Each switch corresponds to one bit of the 15-bit storage address. The operator may protect an address or block of addresses in storage by setting each of the switches to one of its three positions. A view of the Storage Protect switches on the Power Control Panel appears in the Consoles and Power Control Panel section, and Table 2-3 describes the switch positions.

Selective protection may be disabled by pressing the Disable Storage Protect switch on the console. Table 2-4 gives examples of the switch settings needed to protect various blocks of addresses.

NO PROTECTION

Addresses 00002 through 00005, 00010 and 00011, which are used by the interrupt system, are never protected during the interrupt sequence.

TABLE 2-3. STORAGE PROTECTION SWITCH DESCRIPTIONS

Output	Switch Position	Description
"1"	Up	Each address protected will have a "1" in this bit position.
"N"	Center	Each address protected may have either a "1" or a "0" in this position. For example, when all switches are set to the neutral position, all storage is protected, provided that the protect feature is enabled.
"0"	Down	Each address protected will have a "0" in this bit position.

TABLE 2-4. STORAGE PROTECTION SWITCH SETTINGS

Description of Protected Addresses	Examples:	
	Settings — Storage Protection Switches	Addresses Protected (octal)
Single storage address	000 000 000 001 111	00017
Two nonsequential addresses of a group of 10_8.*	000 000 000 010 0N0 000 000 000 010 N10	00020 & 00022 00022 & 00026
Four nonsequential addresses of a group of 10_8.*	000 000 000 010 N0N 000 000 000 010 NN1	00020, 00021, 00024, & 00025 00021, 00023, 00025, & 00027
Four address block — may be the upper or lower half of a group of 10_8.*	000 000 000 100 0NN 000 000 000 100 1NN	00040-00043 00044-00047
10_8 address block	000 000 000 010 NNN	00020-00027
20_8 address block	000 000 001 00N NNN 000 000 001 11N NNN	00100-00117 00160-00177
40_8 address block — may be the upper or lower half of a group of 100_8.*	100 000 000 0NN NNN 100 000 000 1NN NNN	40000-40037 40040-40077
Numerous other groups and combinations of the above groups may also be protected.	000 000 000 NNN 110 NNN NNN NNN NNN 111 NNN NNN 001 NNN NNN	00006, 00016, 00026 . . . 00076 All XXXX7 addresses All XX1XX addresses (00100-00177, 01100-01177, etc.)

* The first address of all groups of 10_8, 20_8, 40_8, 100_8, etc., must have a lower octal digit of zero. Blocks of 100_8, 200_8, 400_8, 1000_8, 2000_8, 4000_8, etc., may be protected in the same manner as blocks of 10_8, 20_8, & 40_8.

Section 3
INPUT/OUTPUT CHARACTERISTICS

Data is transferred between a 3100 Computer and its associated external equipment via a 3106 or 3107 Communication Channel. For programming purposes, the four possible 3106 channels in a system are designated by numbers 0 through 3. A 3107 replaces the 3106 type I/O channels 2 and 3 in expanded systems. It is programmed as channel 2.

INTERFACE SIGNALS

Up to eight external equipment controllers may be attached in parallel to each 3106 Communication Channel. Figure 3-1 shows the principal signals which flow between a 3106 and its external equipment. The 12 status lines are active only between the channel and the controller to which it has been connected by the CON (77.0) instruction. The eight interrupt lines, designated 0-7, connect to all eight controllers attached to a channel. These lines match the Equipment Number switch setting on each controller. For a complete description of the I/O interface signals as well as an I/O timing chart, refer to the 3000 Series I/O Specification, publication number 600048800.

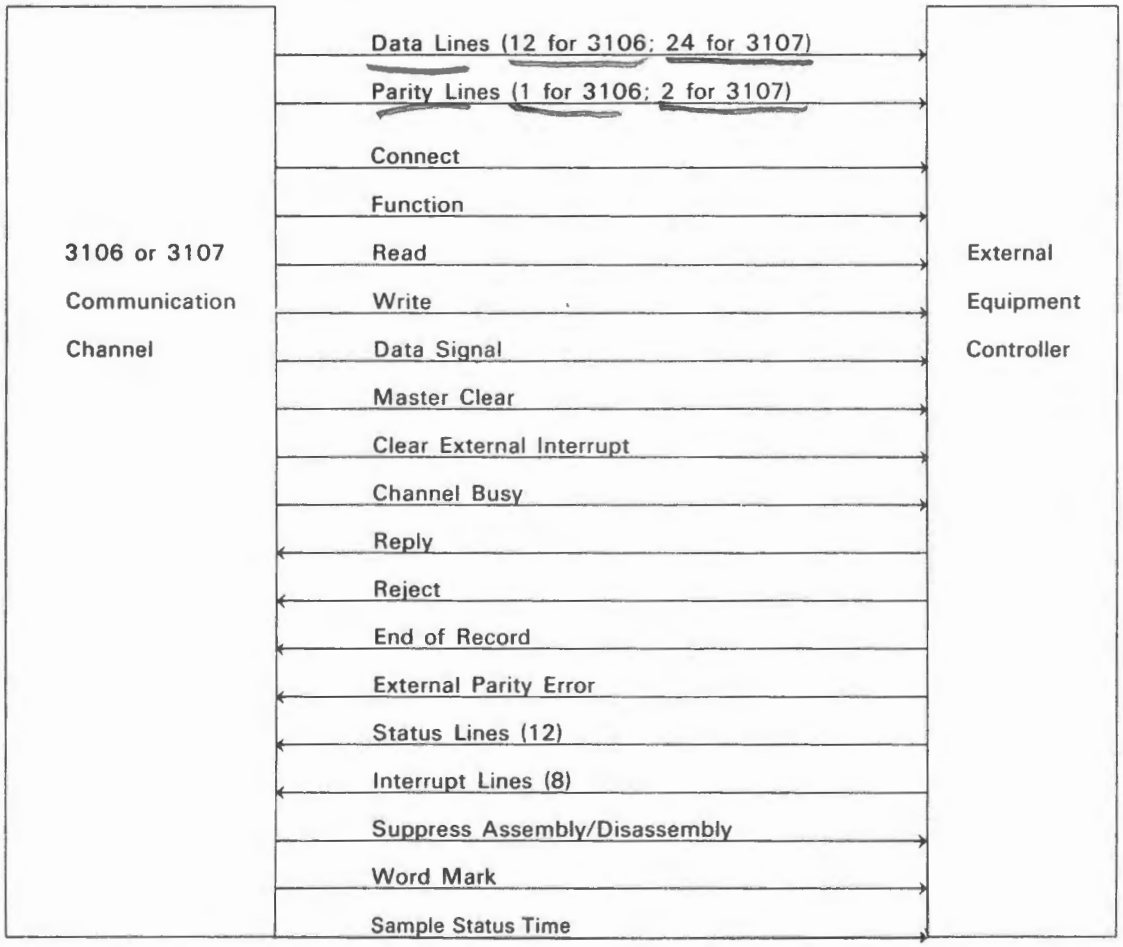


Figure 3-1. Principal Signals Between I/O Channel and External Equipment

I/O PARITY

PARITY CHECKING WITH THE 3106

The computer checks parity by one method for Connect, Function and Write operations, and by a second method for Read operations.

Connect, Function and Write

During the Connect, Function and Write operations, the Data Bus circuit of the computation section generates a parity bit and sends it to the external equipment with each 12-bit byte of data via the I/O channel. The external equipment generates a second parity bit and compares it with the parity bit from the computer. If an error exists, the external equipment sends an External Parity Error signal back to the I/O channel. This signal causes the logic within the channel to provide a "1" on sense line 0. The logic is cleared every time an attempt is made to execute a Connect, Function, Read, or Write operation with this channel. It may also be channel-cleared by the program or master-cleared by the operator. If a transmission parity error is received from a controller, the controller remains inactive until the I/O channel is cleared.

Read

During a Read operation, the external equipment generates a parity bit and sends it to the I/O channel along with each 12-bit byte of data. The I/O channel holds the parity bit while the data is forwarded to the computation section. The Data Bus circuit of the computation section generates a second parity bit and sends it back to the I/O channel. The channel compares this second signal with the parity signal which was generated by the external equipment. If an error exists, certain channel logic is set by an enable from the computation section. This logic provides a "1" on sense line 0. The channel parity logic is cleared every time an attempt is made to execute a Connect, Function, Read or Write operation with this channel. It may also be channel-cleared by the program or master-cleared by the operator. If a transmission parity error is channel-generated, it must be sensed by the INS instruction. If the error is not sensed, the next channel operation will clear the error indication.

PARITY CHECKING WITH THE 3107

The computer checks parity in a 3107 in a slightly different manner than in a 3106.

Connect, Function and Write

During the Connect, Function and Write operations, the Data Bus circuit in the computation section generates a parity bit for the lower 12-bit byte of each data word. The 3107 generates a parity bit for the upper byte. Both parity bits are sent to the external equipment via the I/O channel. The external equipment generates parity bits and compares them with the parity bits from the computer. If an error exists, the external equipment sends an External Parity Error signal back to the I/O channel where it can set the channel parity logic and provide a "1" on sense line 0. Clearing the logic occurs in the same way as it does in the 3106. If a transmission parity error is received from a controller, the controller remains inactive until the I/O channel is cleared.

Read

During a Read operation, the external equipment generates two parity bits per data word, one for each 12-bit byte, and sends them to the 3107 along with the word. The I/O channel holds the parity bit for the lower byte while it forwards the byte to the computation section. The Data Bus circuit of the computation section generates a second parity bit for this byte and sends it back to the I/O channel.

Simultaneously, the 3107 retains the parity bit for the upper byte of the data word. The I/O channel generates a second parity bit for the upper byte as it forwards the byte to the computation section.

The 3107 compares the two parity bits generated by the computer with the two parity bits generated by the external equipment. If an error exists, the channel parity logic is set by an enable from the computation section, thus providing a "1" on sense line O. Clearing the logic also occurs the same way as it does in the 3106. If a transmission parity error is channel-generated, it must be sensed by the INS instruction. If the error is not sensed, the next channel operation will clear the error indicator.

AUTO LOAD/AUTO DUMP

The Auto Load/Auto Dump feature allows the programmer 32_{10} storage addresses in which to store two short routines. These routines are used generally to receive and transmit data to external equipment. Assuming the routines are already in storage, the operator can initiate these operations with the AUTO LOAD and AUTO DUMP switches on the console.

PRELIMINARY CONSIDERATIONS

Addresses 77640 through 77677 are normally protected from being written into. To enter Auto Load or Auto Dump routines, the operator presses the ENTER AUTO PROGRAM switch on the console, enters the routine, then master clears the computer. Before pressing the AUTO LOAD or AUTO DUMP switches, the operator must first master clear the computer.

AUTO LOAD

The AUTO LOAD switch automatically sets (P) to address 77640. This group of 16 instructions may be used to bring in a program from a magnetic tape unit or other peripheral device. The last instruction in this routine should be a jump to the first address of the newly stored program.

AUTO DUMP

The AUTO DUMP switch automatically sets (P) to address 77660. This group of 16 instructions is most often used to output a block of data to a magnetic tape unit or other peripheral equipment. The last instruction in this routine may be a jump to any storage area.

3100 AUTO LOAD/AUTO DUMP INTERIM SUBROUTINE

The following 3100 Auto Load/Auto Dump subroutine may be used in lieu of a special routine for a particular installation. This routine may be used for selecting any peripheral device on any available channel. Refer to the I/O instructions in Section 7 for a complete description of the parameter designators.

Parameters:

B^1 = Return address (jump to (B^1) on channel Busy)
B^2 = Connect code (ch,x)
B^3 = Function code (ch,x)
A = Read/Write address (ch, P,m)
Q = Terminal address +1 (n)

Where: ch = I/O channel
x = Connect or Function code
P = Read/Write parameters (if used, i.e. B, N or H and INT)
m,n = Certain addresses (refer to designators in Section 7)

If any parameters are not used, they are assigned the following values:

B^1 = 00000
B^2 = 00000
B^3 = 00010
A = 00000000
Q = 00000000

SLS = Set new parameters
SJ1 = Repeat last parameters (overrides SLS)
$\overline{SJ1}$ • SLS = Rewind tape unit 0 and read/write at location 00000. Jump to location 00000 on Channel Busy
(B^1) = 77625 for Repeat
(B^1) = 77644 for Stop

Addresses 77625 through 77637 are used during the execution of the subroutine.

ADDRESS	INSTRUCTION	ADDRESS	INSTRUCTION
77640	23377222	77660	23377223
77641	00177646	77661	01077641
77642	72377124	77662	47377627
77643	24377330	77663	47277625
77644	00007476	77664	47177636
77645	77604360	77665	01077625
77646	43377144	77666	77000000
77647	00177665	77667	01077625
77650	14700000	77670	77100000
77651	77700000	77671	01077627
77652	40077632	77672	00000000
77653	13000030	77673	00000000
77654	44077631	77674	01077631
77655	05300001	77675	77600XXX*
77656	14300010	77676	01077634
77657	01077662	77677	01000000

*XXX The maximum number of I/O channels in the system (Pause Mask).

SATELLITE CONFIGURATIONS

Figure 3-2 shows three possible Satellite configurations that utilize one or more 3100 Computer Systems.

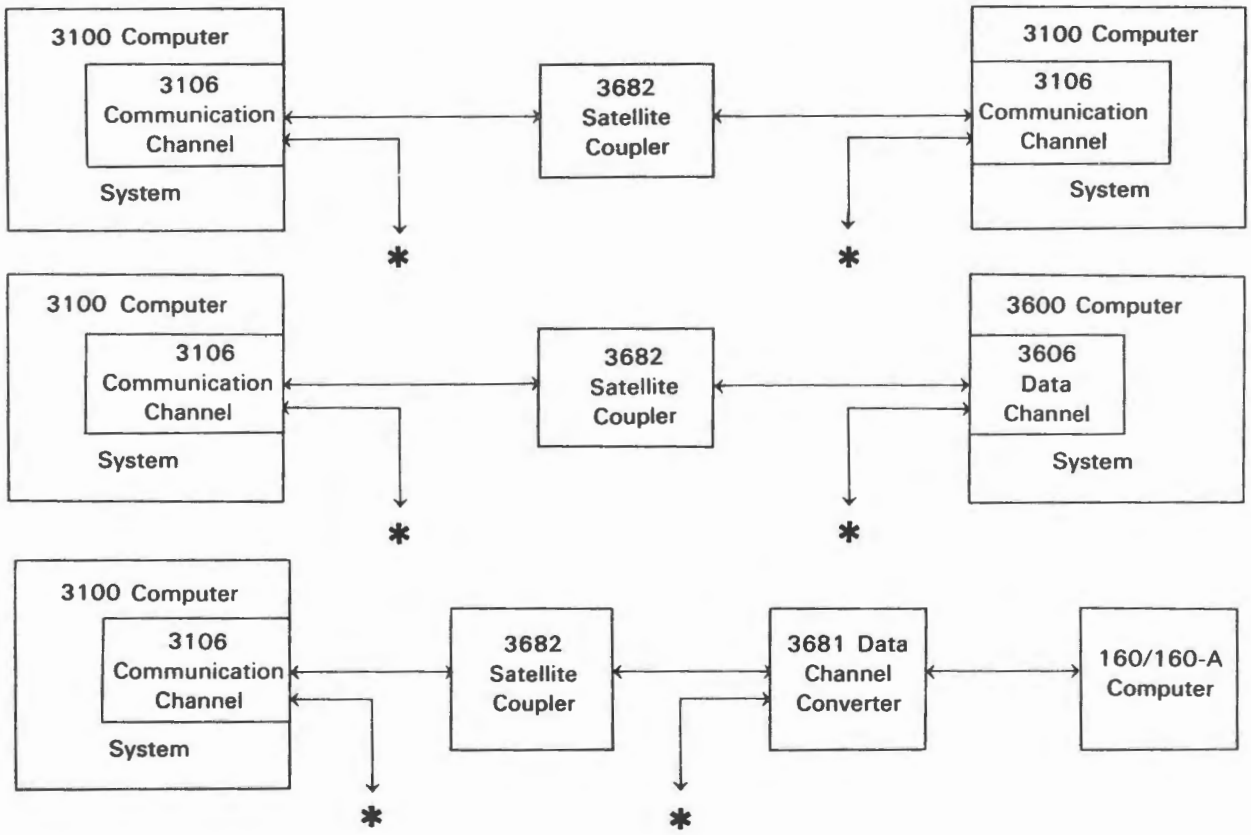


Figure 3-2. Satellite Configurations

*NOTE: May be connected to seven additional external equipments.

Section 4
INTERRUPT SYSTEM

GENERAL INFORMATION

The Interrupt Control section of the 3100 Computer is capable of testing for the existence of certain internal and external conditions without having these tests in the main program. Examples of these conditions are internal faults and external equipment end-of-operation. Near the end of each RNI cycle, a test is made for interruptible conditions. If one of these conditions exists, execution of the main program halts, the contents of the Program Address register are stored, and an interrupt routine is initiated. This interrupt routine, initially stored in memory, performs the necessary functions for the existing condition and then jumps back to the last unexecuted step in the main program. The instruction being read when the interrupt is recognized is executed when the main program is resumed.

There are four categories of interrupts in the 3100 Computer: Internal Condition interrupts, Input/Output (I/O) interrupts, Trapped Instruction interrupts and a special Power Failure interrupt. The store operations required for all four types of interrupts occur regardless of the state or selection of the storage protection feature described in Section 2.

An additional manual interrupt is set by a switch on either the computer or typewriter console. This interrupt is not masked since this switch is pressed only when an interrupt is desired. The interrupt is recognized if the interrupt system is enabled. The interrupt condition is automatically cleared after the interrupt is recognized.

INTERRUPT CONDITIONS

INTERNAL INTERRUPTS

Any one of six internal conditions may cause an interrupt during the execution of a program. These conditions and their descriptions follow.

Arithmetic Overflow Fault

The Arithmetic Overflow fault is set when the capacity of the adder is exceeded. Its capacity, including sign, is 24 or 48 bits for 24-bit precision and 48-bit precision, respectively.

Divide Fault

The Divide fault sets if a quotient, including sign, exceeds 24 or 48 bits for 24-bit precision and 48-bit precision, respectively. Therefore, attempts to divide by too small a number, including positive and negative zero, result in a Divide fault. A Divide fault also occurs when a floating point divisor is either equal to zero or not in floating point format. The results in the A, Q, and E registers are insignificant if a fault occurs. A Divide fault can be correctly sensed only after the current instruction has been executed.

Exponent Overflow/Underflow Fault

During all floating point arithmetic operations, exponential overflow occurs if the exponent exceeds $+1777_8$ or exponential underflow will occur if the exponent is less than -1777_8.

BCD Fault

A BCD Fault is set if:

1. The lower 4 bits of any character, except the least significant, exceeds 11_8 (9_{10}). Characters are tested for legality only during the LDE, ADE, and SBE instructions. In all cases, if the value 11_8 (9_{10}) is exceeded, the value zero is used for that character.
2. The upper 2 bits of any character, except the least significant, do not equal zero.
3. An attempt is made to set (load) the D register with 15_8, 16_8 or 17_8.

Search/Move Interrupt

The Search/Move control may be programmed to generate an interrupt during a 71 or 72 instruction for either of the following conditions:

1. Completion of an equality or inequality search.
2. Completion of a block move.

Real-Time Clock Interrupt

The Real-Time Clock interrupt is generated when the clock reaches a prespecified time that has been stored in register 32 of the Register File.

TRAPPED INSTRUCTION INTERRUPTS

A translator within the 3100 Computer detects and traps the 55-70 instructions if the appropriate option is not present in the system. Although they are not true interrupts, trapped instructions are processed like interrupts once they have been detected. A conventional interrupt always takes priority over a trapped sequence. The following operations take place when a trapped instruction is recognized:

1. P + 1 is stored in the lower 15 bits of address 00010.
2. The upper 6 bits of F are stored in the lower 6 bits of address 00011; the upper 18 bits remain unchanged.
3. Program control is transferred to address 00011 and an RNI cycle is executed.

Further information on trapped instructions may be found in the General Information paragraph of Section 7.

POWER FAILURE INTERRUPT

If source power to a 3100 Computer is removed, the failure is detected and the computer program is interrupted; this interrupt is necessary to prepare for a controlled shutdown and prevent the loss of data. This operation requires 16 ms for detection, and up to 4 ms for processing a special Power Failure interrupt routine.

The Power Failure interrupt overrides any other interrupt (internal or I/O), as well as the trap sequence, regardless of the state of the interrupt control. Since this interrupt overrides all others, the address where the present contents of P are stored and the address to which program control is transferred must be different from that for a normal interrupt. When a Power Failure interrupt occurs, the machine stores the contents of P in the lower 15 bits of address 00002 and transfers program control to address 00003.

The normal interrupt system is disabled during a power failure sequence; i.e., the hardware simulates the execution of a DINT (77.73) instruction.

I/O INTERRUPTS

I/O Channel Interrupts

Any of the four possible I/O channels may be programmed to generate an interrupt for either of the following conditions:

1. Reaching the end of an input or output block.
2. Receiving an End of Record (Disconnect) signal from an external device.

I/O Equipment Interrupt

The I/O equipment interrupt is set when an interrupt signal is received from any of eight peripheral equipment controllers connected to any of the four possible I/O channels (there may be a total of 32 interrupt lines). The interrupt remains set until the computer directs the originating device to cancel it with a function code.

Associated Processor Interrupt

In a system of two or more processors (computers), each processor may interrupt the processor to its left by executing an IAPR (77.57) instruction. The interrupting processor must interrupt via its storage modules 0 and 1, which are storage modules 2 and 3 of the processor being interrupted. This interrupt is not masked and becomes cleared as soon as it is recognized.

INTERRUPT MASK REGISTER

The programmer can choose to honor or ignore an interrupt by means of the Interrupt Mask register. All but two of the normal interrupt conditions are represented by the 12 Interrupt Mask register bits. The manual interrupt and the associated processor interrupt are not masked. The mask is selectively set with the SSIM (77.52) instruction and selectively cleared by the SCIM (77.53) instruction. See Table 4-1 for Interrupt Mask register bit assignments.

The contents of the Interrupt Mask register may be transferred to the upper 12 bits of the A register for programming purposes with the COPY (77.2) or CINS (77.3) instructions.

TABLE 4-1. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Mask Bit	Mask Codes	Interrupt Conditions Represented
00	0001	I/O Channel 0 (Includes interrupts
01	0002	1 generated within the
02	0004	2 channel and external
03	0010	3 equipment interrupts.)
04	0020	(Not used)
05	0040	(Not used)
06	0100	(Not used)
07	0200	(Not used)
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow & divide faults
11	4000	Search/Move completion

INTERRUPT CONTROL

A program can recognize, sense, and clear interrupts, and enable or disable interrupt control through the use of certain instructions.

ENABLING OR DISABLING INTERRUPT CONTROL

Instruction EINT (77.74) enables the interrupt system and the DINT instruction (77.73) disables it. After recognizing an interrupt and entering the interrupt sequence, other interrupts are disabled automatically. When leaving the interrupt subroutine, the interrupt must again be enabled by the EINT instruction, if awaiting interrupts or subsequent interrupts are to be recognized by the system. After executing an EINT, one more instruction may be performed before the interrupt enable takes effect.

To prevent extraneous interrupts from occurring, the Interrupt Mask register must not be set or cleared while the interrupt system is enabled.

INTERRUPT PRIORITY

An order of priority exists between the various interrupt conditions. As soon as an interrupt becomes active, the computer scans the priority list until it reaches an interrupt that is active. The computer processes this interrupt and the scanner returns to the top of the list where it waits for another active interrupt to appear. Table 4-2 lists the order of priority.

TABLE 4-2. INTERRUPT PRIORITY

Priority	Type of Interrupt
1	Arithmetic overflow or divide fault
2	Exponent overflow/underflow or BCD fault
3-34	External I/O interrupts*
35-38	I/O channel interrupts**
39	Search/move interrupt
40	Real-time clock interrupt
41	Manual interrupt
42	Associated processor interrupt

SENSING INTERRUPTS

The programmer may selectively sense interrupts, independent of the Interrupt Mask register, by using the INTS (77.4) instruction. Sensing the presence of internal faults automatically clears them. Channel interrupt lines that represent channels not present in the system are always sensed as being active. However, the Interrupt Mask register bits representing these missing channels may never be set; therefore, no interrupt can ever occur.

CLEARING INTERRUPTS

I/O equipment interrupts may be cleared by:

- Pressing the EXTERNAL CLEAR button on the console.
- Pressing the entry keyboard MC button.
- Executing an IOCL (77.51) instruction, or
- Reselecting or disabling the interrupt with a function code, SEL (77.1) instruction.

Within a program, I/O channel interrupts must be selectively cleared by the INCL (77.50) or IOCL (77.51) instructions.

*There are eight interrupt lines on each of the four possible I/O channels, or 32 lines in all. On any given channel, a lower numbered line has priority over a higher numbered line. Likewise, a lower numbered channel has priority over a higher numbered channel. Example: line 0 of channel 0 has highest priority of all external I/O interrupts, line 0 of channel 1 has second highest, and line 7 of channel 3 has the lowest.

**A lower numbered I/O channel interrupt has priority over a higher numbered I/O channel interrupt.

The Real-time Clock, Arithmetic, and Search/Move Completion interrupts may be cleared by:

- Sensing, after which the interrupts are automatically cleared.
- Executing an INCL (77.50) instruction, or
- Pressing the MC or INTERNAL CLEAR buttons.

In the INCL instruction, x represents the contents of the Interrupt Mask register. Even though the Interrupt Mask register bits usually represent both I/O channel and I/O equipment interrupts, an INCL instruction clears only internal I/O channel interrupts. Prior to clearing a channel interrupt with an INCL instruction, the program must clear the I/O equipment interrupt with a function code SEL (77.1) instruction. The manual and associated processor interrupts are automatically cleared after they are recognized by the computer during an RNI cycle.

INTERRUPT PROCESSING

Four conditions must be met before a normal interrupt can be processed:

1. With the exception of the Manual interrupt and the Associated Processor interrupt, a bit representing the interrupt condition must be set to "1" in the Interrupt Mask register.
2. The interrupt system must have been enabled.
3. An interrupt-causing condition must exist.
4. The interrupt scanning logic (Refer to Table 4-2) must reach the level of the active interrupt on the priority list.

When an active interrupt has met the above conditions, the following takes place:

1. The instruction in progress proceeds until the point is reached in the RNI cycle where an interrupt can be recognized. At this time the count in P has not been advanced nor has any operation been initiated. When an interrupt is recognized, the address of the current unexecuted instruction in P is stored in address 00004.
2. A number representing the interrupt-causing condition is stored in the lower 12 bits of address 00005 without modifying the upper bits. Table 4-3 lists the octal codes which are stored for each interrupt condition.
3. Program control is transferred to address 00005 and an RNI cycle is executed.

TABLE 4-3
REPRESENTATIVE INTERRUPT CODES

Conditions	Codes
External interrupt	*00LCh
I/O channel interrupt	010Ch
Real-time clock interrupt	0110
Arithmetic overflow fault	0111
Divide fault	0112
Exponent overflow fault	0113
BCD fault	0114
Search/move interrupt	0115
Manual interrupt	0116
Associated processor interrupt	0117

*L = line 0-7 and Ch = channel designator, 0-3

Section 5
CONSOLES AND POWER CONTROL PANEL

The 3100 Integrated Console and the 3101 and 3200 optional desk consoles enable a computer operator to control and observe the operation of the computer. This section describes the operator's controls and the significance of the visual indicators. Also included in this section is a view of the Power Control Panel and a description of its operation.

CONSOLES

Basically, the optional desk consoles are identical to the Integrated Console. Operator switches and keyboard controls on the consoles differ only in their physical placement. This section is concerned primarily with the characteristics of the Integrated Console. Refer to Figure 5-1.

REGISTER DISPLAYS

Communication Register

Data entered into any of the operational registers (except the E_D register) must first pass through the Communication register. Starting with the uppermost digit, data is entered into the Communication register by first depressing a register switch and then depressing the numeric keyboard swithces. An Active Digit indicator light is located directly above each group of three binary lights which represent the octal digits in the Communication register. Whenever a numeric keyboard switch is depressed, the binary lights beneath the illuminated Active Digit indicator will display the newly entered contents of that digit position. At the same time, the Active Digit indicator for that group goes off and the one for the next lower numbered octal digit is illuminated. When data is to be entered into the B^1, B^2, B^3, or P registers, the Active Digit indicator automatically starts at the fifth digit position of the Communication register.

Depressing the TRANSFER switch causes the data to be transferred from the Communication register to the designated register. Depressing the TRANSFER switch again results in transferring all zeros to the register.

E Register

The E register is displayed as either E_U and E_L or E_D. Whenever the E register is being displayed, the A and Q registers cannot be displayed and vice versa. The register(s) currently displayed is denoted by the illumination of one of the three register display indicators located above each of the two arithmetic register displays.

Figure 5-2 illustrates specific digit functions when the E_U E_L register is displayed on the Integrated Console. Figure 5-3 illustrates the digit functions when the E_D register is displayed.

NOTE

The E_D register may be entered directly with any of the 10 numeric keyboard characters. As each digit is entered, the preceding digit is shifted one digit position left, increasing its significance. Each succeeding entry shifts the digits one position left and inserts the newly entered digit into the lowest order position. After a maximum of 13 digits have been entered (including the overflow digit) the uppermost characters are shifted end-off as additional characters are entered. The E_U E_L register cannot be entered into by a keyboard operation. Appropriate inter-register transfer instructions must be utilized for entry into this register. Refer to the BCD section of Appendix B for an illustration of the E_D register as displayed on the optional 3200 console.

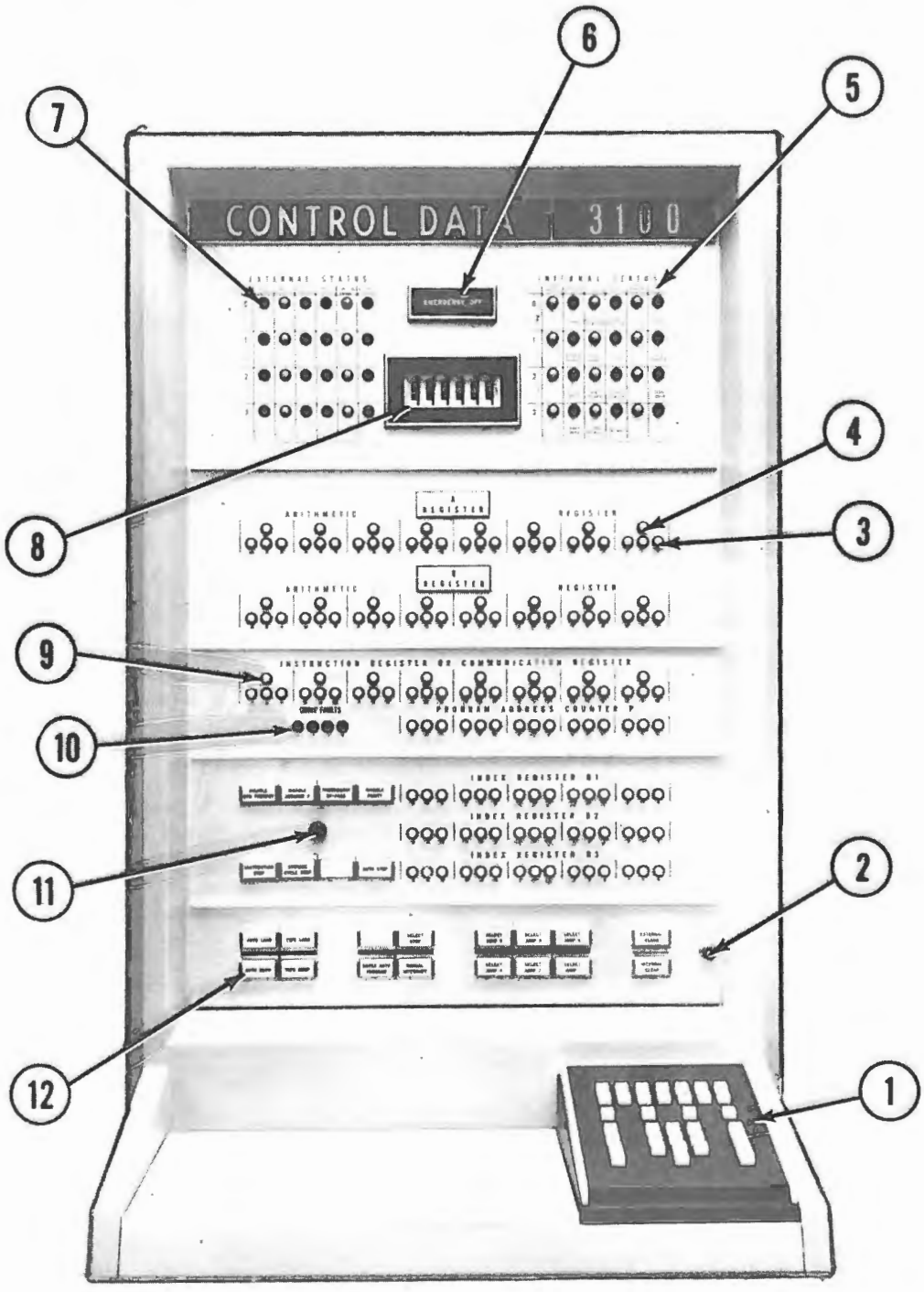


Figure 5-1. Front View of Integrated Console.

1. Detachable keyboard.
2. Panel retaining thumb screw.
3. Binary register digits.
4. BCD bit (for indicating 8's and 9's when option is present).
5. Internal status indicators.
6. Emergency power cutoff switch.
7. External status indicators.
8. Thumbwheel breakpoint switch.
9. Active digit indicator.
10. Drive fault indicators.
11. Adjustable auto-step control.
12. Various operator/maintenance pushbutton switches.

NOTE

In some 3100 systems, the A and Q register name panels are replaced with indicator lights. These indicators include the E register even though the arithmetic options may not be present in the system.

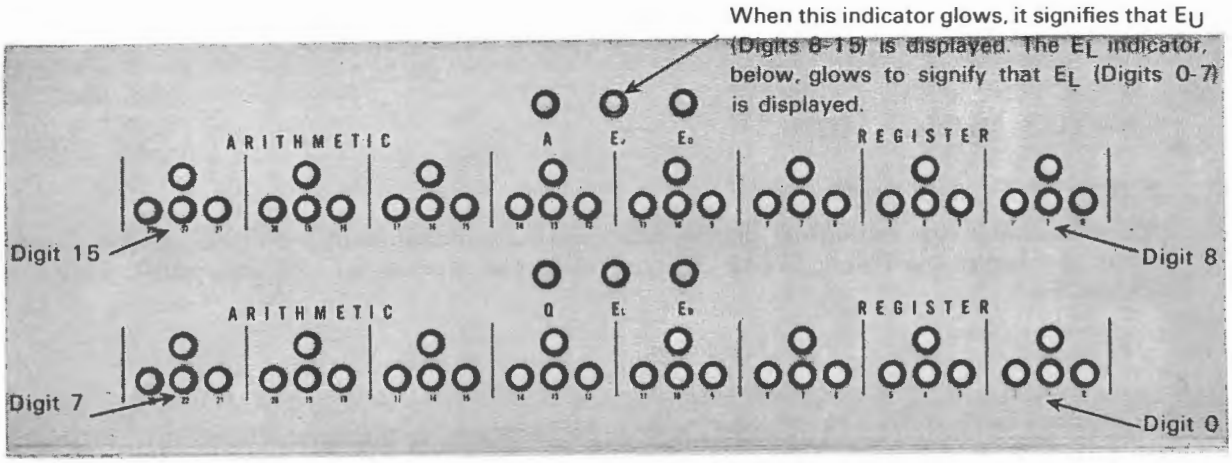


Figure 5-2. E_U E_L Register Display

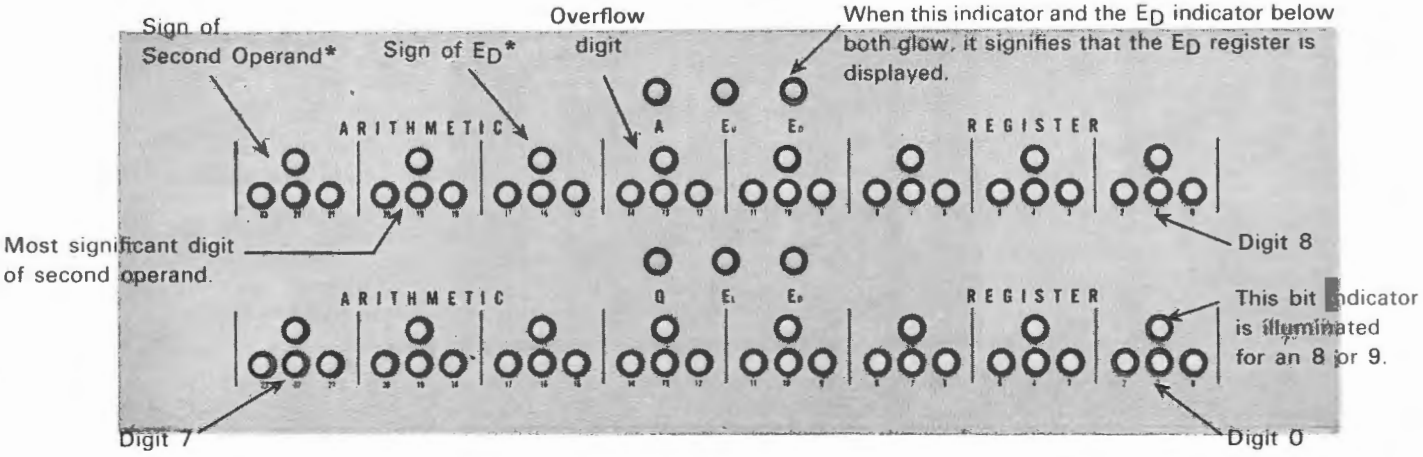


Figure 5-3. E_D Register Display

*A "+" sign is indicated by 10_8 displayed in the sign digit.
A "−" sign is indicated by 11_8 displayed in the sign digit.

Other Registers

The A, Q, P, B^1, B^2 and B^3 registers, described in the System Description Section of this manual, are displayed on the Integrated Console in binary form.

CONSOLE LOUDSPEAKER

The console loudspeaker and its associated volume control are mounted underneath the console table. The loudspeaker receives its input from the upper 3 bits of the A register. An audible sound is produced when one or more of these bits are toggled at an audio rate. Loudspeaker volume is controlled by rotating the volume control.

STATUS INDICATORS

External Status Indicators

The external status indicators display the existing conditions of I/O channels 0-3. Conditions displayed are Read, Write, Reject, Connect, Function, and Interrupt. Refer to Figure 5-4.

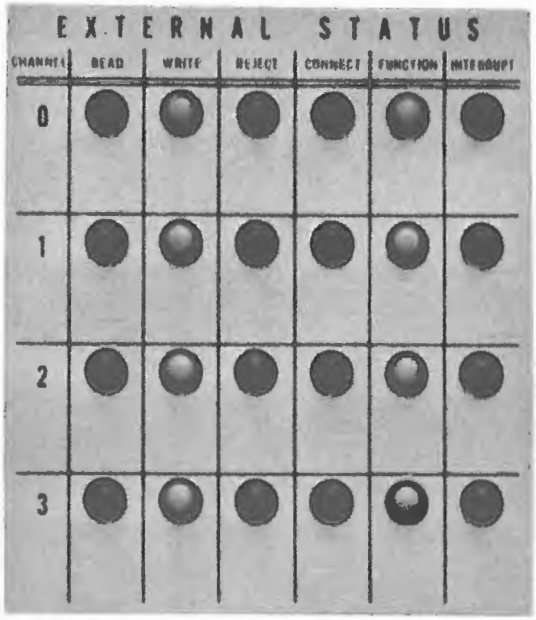


Figure 5-4. External Status Indicators

Internal Status Indicators

Six columns of internal status indicators are located on the display section of the consoles. Refer to Figure 5-5. When the particular indicator is glowing, the condition or fault described on page 5-5 exists:

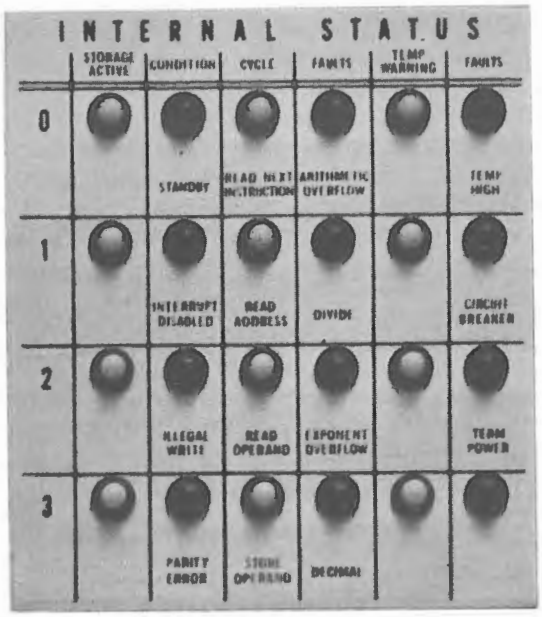


Figure 5-5. Internal Status Indicators

1. STORAGE ACTIVE 0-1-2-3

   The Storage Active lights indicate the storage area currently being referenced. Digit 0 glows when the first 4K or 8K of storage is referenced. In expanded 3100 systems, digit 1 indicates that the second 8K storage section is referenced, digit 2 the third 8K section, and digit 3 glows when the fourth 8K section is referenced.

2. CONDITIONS

   STANDBY – Indicates that the main power switch is on but the individual logic supplies are still off.

   INTERRUPT DISABLED – Indicates the interrupt system has been disabled by executing the DINT (77.73) instruction.

   ILLEGAL WRITE – Glows whenever an attempt is made to write into the area of storage currently being protected by the storage protect switches. This indicator will also glow if an attempt is made to write into the Auto Load or Auto Dump storage areas. This condition is cleared by executing an INS (77.3) instruction or performing a Master Clear.

   PARITY ERROR – Indicates that a parity error has occurred in storage. When the error is detected, this indicator glows and program execution stops. Performing a Master Clear clears the condition. Transmission parity errors do not affect this indicator.

3. CYCLE (RNI-RAD-ROP-STO)

   These indicators represent the four program cycles: Read Next Instruction, Read Address, Read Operand, and Store Operand. They are lit while the respective cycles are in progress.

4. FAULTS

   This column of indicators represents the four arithmetic fault conditions:

   ARITHMETIC OVERFLOW – The arithmetic overflow fault is set when the capacity of the adder is exceeded. Its capacity, including sign, is 24 or 48 bits for 24-bit precision and 48-bit precision, respectively.

DIVIDE – The divide fault sets if a quotient, including sign, exceeds 24 or 48 bits for 24-bit precision and 48-bit precision, respectively. Therefore, attempts to divide by too small a number, including positive and negative zero, result in a divide fault. During floating point division, a divide fault occurs if division by zero or by a number that is not in floating point format is attempted. If the divisor is not properly normalized a divide fault may also occur. Refer to Appendix B for a description of normalization.

EXPONENT OVERFLOW/UNDERFLOW – This fault indicator glows when either an exponent overflow ($>+1777_8$) or an exponent underflow ($<-1777_8$) condition exists.

DECIMAL – A decimal (BCD) fault is set if:

- The lower 4 bits of any character except the least significant exceed 11_8 (9_{10}). Characters are tested for legality only during the LDE, ADE and SBE instructions. In all cases, if the value 11_8 (9_{10}) is exceeded, the value zero will be used for that character.
- The upper 2 bits of any character except the least significant do not equal zero.
- An attempt is made to load the D register with 15_8, 16_8 or 17_8.

5. TEMPERATURE WARNING

   If the upper temperature limit of the normal operating range within a section of the computer is exceeded, a corresponding TEMP WARNING indicator glows. The indicators correspond to computer sections illustrated in Figure 5-6.

6. FAULTS

   This column of indicators represents abnormal operating conditions.

   TEMPERATURE HIGH – If the TEMP WARNING indicators are glowing and an absolute temperature is exceeded, the computer will automatically shut off logic power. The TEMP HIGH indicator for the particular computer section continues to glow until the temperature drops below the absolute limit. Secondary power must be manually re-applied before normal operation can resume. If the THERMOSTAT BYPASS console switch is on, the TEMP HIGH indicator glows and the temperature protection feature is defeated.

   CIRCUIT BREAKER – This indicator glows if the circuit breakers governing any of the internal power supplies are off.

   TERMINATOR POWER – If output power from the internal terminator power supplies fails, this indicator glows.

Temperature Indicator **2**	Temperature Indicator **1**	Temperature Indicator **0**	Temperature Indicator **3**	
8K Storage and I/O Logic	Block Control, Interrupt, 4K Storage Optional Arithmetic and I/O Logic	Main Control and Arithmetic Logic	16K Storage and I/O Logic	Power Control Panel

Figure 5-6. Temperature Warning Designations for an Expanded 3100 Computer, Front View.

SWITCHES

Switches associated with a 3100 Computer are classified as console switches and keyboard switches. Console switches include the following:

- The EMERGENCY OFF switch.
- A group of operator/maintenance switches on the console main-frame.
- The Breakpoint switch assembly (Figure 5-8).

Keyboard Switches

The console keyboard switches are used for entering data manually into the computer and for controlling its operation. A front view of the keyboard appears in Figure 5-7 and Table 5-1 describes the function of the keyboard switches.

Console Switches

EMERGENCY OFF SWITCH — This red rectangular momentary switch is used to remove power from the whole computer system in case of a fire or other emergency. It should not be used for a normal power shutdown. Refer to the SOURCE POWER OFF switch description in the Power Control Panel description of this section.

OPERATOR/MAINTENANCE SWITCHES — Table 5-2 describes the operator/maintenance switches located on the console main-frame.

BREAKPOINT SWITCH ASSEMBLY — The Breakpoint switch is a six-section, eight-position, thumb-wheel switch. The left-hand wheel selects the operating mode, and the other five wheels specify a register number or storage address. There are four mode positions on the mode selector switch with an OFF position between each mode; these modes are BPI, BPO, REG, and STO.

BPI and BPO Modes: The address on the S Bus is continually compared with the instruction or operand address specified by the Breakpoint digit switches. When the selector switch is set to BPI, the computer stops if these values become equal during an RNI (Read Next Instruction) sequence. When the mode selector switch is set to BPO, the computer stops if these values become equal during an ROP (Read Operand) or STO (Store) sequence.

REG and STO Modes: In these two modes, the operator may either monitor the contents of a register location or storage address specified by the thumb-wheel digit switches, or he may store a word in these locations. To monitor a storage location:

1. Set the mode selector to REG (register file location) or STO (storage).
2. Set the Breakpoint switch to the desired register number or storage address.
3. Press the READ STO switch on the keyboard.
4. Adjust the Auto Step control to vary the display rate.

The register or storage contents are repeatedly displayed in the Communication register at the selected repetition rate until another keyboard button is pressed to release READ STO. To write a word in storage:

1. Set the mode selector to REG or STO.
2. Set the Breakpoint switch to the desired register number or storage location.
3. Press the WRITE STO switch on the keyboard.
4. Enter data into the Communication register by depressing the numeric switches and finally the TRANSFER switch.

The data is entered into the desired storage location or Register File location at the end of the instruction that is currently being executed by the computer. Pressing any other register or mode selector switch releases WRITE STO operation.

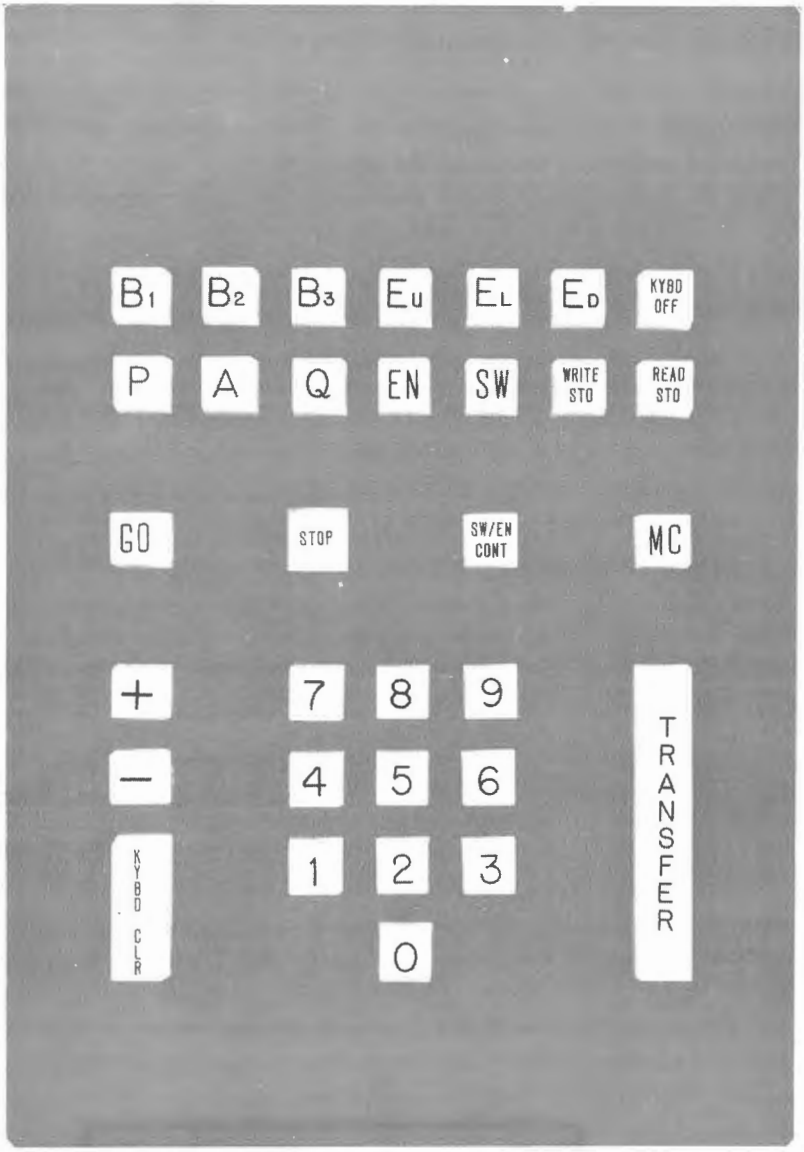


Figure 5-7. Console Keyboard

NOTE

The upper two rows of keyboard switches are mechanically linked together. This feature prevents more than one switch from being active at any one time.

TABLE 5-1. KEYBOARD SWITCH FUNCTIONS

SWITCH NAME	ILLUMINATED	DESCRIPTION
B^1 to B^3	Yes	Enables data to be manually entered into Index registers B^1 B^2, or B^3 from the keyboard.
P	Yes	Enables an address to be manually entered from the keyboard into the P register.
A	Yes	Causes both A and Q to be displayed, but permits entry only into A.
Q	Yes	Causes both A and Q to be displayed, but permits entry only into Q.
E_U*	Yes	Causes E_U and E_L to be displayed. Manual entry is not possible.
E_L*	Yes	Same as E_U.
E_D*	Yes	Causes E_D to be displayed and enables manual entry directly into this register. Refer to E_D register description.
KYBD OFF (Keyboard Off)	Yes	Deactivates all keyboard controls.
EN (Enter)	Yes	Permits data to be manually entered into storage while the computer is stopped. First address of sequence must be previously entered into P. Pressing the TRANSFER switch advances P.
SW (Sweep)	Yes	Permits unexecuted instructions to be read from consecutive storage locations. First address of sequence must be first entered into P. Pressing the TRANSFER switch advances P.
WRITE STO (Write Storage)	Yes	Permits keyboard entry into the storage location specified by the thumb-wheel switches. Entry occurs each time the TRANSFER switch is pressed whether the computer is in the GO mode or stopped.
READ STO (Read Storage)	Yes	Permits the contents of the storage register location specified by the thumb-wheel switches to be displayed. The display rate is determined by the Auto-Step control.
KYBD CLR (Keyboard Clear)	Yes	Clears the Communication register.
GO	Yes	Starts the program execution at the address specified by the P register. Not used for Sweep or Enter operations.
SW/EN CONT (Sweep/Enter Continuous)	Yes	Enables Sweep or Enter operations to proceed continuously through storage without pressing the TRANSFER switch.
STOP	Yes	Stops the computer at the end of the current instruction.
TRANSFER	No	Transfers data in the Communication register to a selected register or storage location.
MC (Master Clear)	No	Performs both an internal and external clear. Disabled when GO switch is depressed and the computer is in the GO mode.
0 through 7	No	These switches, when pressed one at a time, allow entry of that particular digit into the Communication register in the binary digit position denoted by the active digit indicator.
8 and 9	No	Depressing either of these switches permits entry of that digit directly into the E_D register. The option must be present in the system and the E_D register selection switch depressed.
+ or − (Plus or Minus)	No	Depressing either of these switches permits entry into the sign of E_D digit (refer to Figure 5-2) in the E_D register. These switches may be depressed at any time during the numeric entry of E_D. The sign of E_D may be changed by depressing the opposite sign switch.

*Depressing any of the switches associated with the arithmetic options when the optional logic is not present produces equivocal results.

TABLE 5-2. CONSOLE MAIN-FRAME SWITCHES

SWITCH NAME	FUNCTION
MANUAL INTERRUPT	Forces the computer into an interrupt routine if the computer is in the GO mode. If the computer is stopped when the switch is pressed, it will go into an interrupt routine as soon as the GO switch is depressed.
SELECT STOP 1	Stops the computer when the SLS (77.70) instruction is read.
SELECT JUMP (1 through 6)	Switches are depressed in accordance with programs utilizing the selective jump (SJ1-6) instruction.
ENTER AUTO PROGRAM	Allows the operator to enter the Auto Load and Auto Dump storage areas (addresses 77640 to 77677) with different data.
EXTERNAL CLEAR	Master clears all external equipments and the I/O channels.
INTERNAL CLEAR	Master clears internal conditions and registers.
DISABLE STO PROTECT	Disables the protection feature of the 15 storage protect switches. This switch has no effect on the protected Auto Load and Auto Dump storage areas.
DISABLE ADVANCE P	Prevents the P register from being incremented. When the GO switch on the keyboard is depressed, the same instruction is repeated.
THERMOSTAT BYPASS	Allows computation to proceed regardless of unfavorable temperatures within the computer.
DISABLE PARITY	Prevents recognition of parity errors from all storage modules.
INSTRUCTION STEP	Enables the operator to step through the program instruction by instruction. An instruction is executed each time the switch is depressed.
BCD STEP	Enables the operator to step through a BCD instruction one sequence at a time.
STORAGE CYCLE STEP	Enables the operator to step through an instruction one storage cycle at a time, i.e. RNI, RAD, ROP, or STO.
AUTO STEP	Permits instructions to be executed in a slow speed GO mode. The speed is regulated by the auto-step speed control on the console. There are approximately 3 to 50 instructions executed per second.
AUTO LOAD	If the computer has been master cleared and the Auto Load switch is depressed, the computer will automatically jump to address 77640 and execute the instruction stored there. Refer to Auto Load/Auto Dump in Section 3.
TYPE LOAD	Permits the operator to enter a block of data from the typewriter. The data is defined by the lower bounds in register 23 and upper bounds in register 33 of the Register File. Refer to the Typewriter Section for additional information.
AUTO DUMP	This switch performs the same function as the Auto Load switch with the exception of jumping to address 77660.
TYPE DUMP	Similar to the Type Load operation, this switch causes a block of data to be printed by the typewriter. The data in storage is defined by registers 23 and 33.

Examples of Keyboard Switch Functions

1. To enter data into the A register:
   a. Depress the A register switch.
   b. Enter all eight digits of the Communication register by depressing the appropriate numeric key switches.*
   c. Depress the TRANSFER switch.
   d. Depress the KEYBOARD OFF switch.
2. To enter data into the Q register:

   Depress the Q register switch and repeat steps *b* through *d* of example 1.
3. To enter the Program Address Counter (P register) with a specific address:
   a. Depress the P register switch.
   b. Enter the lower five digits of the Communication register by depressing the appropriate numeric key switches.
   c. Depress the TRANSFER switch.
   d. Depress the KEYBOARD OFF switch.
4. To enter an operand at a specific address**:
   a. Perform step 3.
   b. Depress the EN switch.
   c. Enter all eight digits of the Communication register by depressing the appropriate numeric key switches.
   d. Depress the TRANSFER switch.
   e. The count in the Program Address Counter has now incremented by one. If data is to be entered into this memory location, repeat steps *c* and *d* for as many succeeding entries as required.
   f. Depress the KEYBOARD OFF switch when all data has been entered into the successive group of memory locations.
5. To read an operand from a specific storage address:
   a. Perform step 3.
   b. Depress the SW switch.
   c. Depress the TRANSFER switch.
   d. The contents of the specified storage address are now displayed in the Communication register. (The Program Address Counter is not incremented when the TRANSFER switch is initially depressed.)
   e. If the TRANSFER switch is depressed again, the Program Address Counter is incremented by one, and the contents of the new address are displayed.
   f. Depress the KEYBOARD OFF switch when all the desired memory locations within a successive group have been examined.
6. To enter zeros or another operand into all storage locations:

NOTE

Step 5 only permits the operator to examine the contents of specific storage locations. The instructions are not executed during this operation.

*If all eight digit positions of the Communication register are not entered before the Transfer switch is depressed, zeros will be entered into the remaining digit positions.

**The breakpoint switch may be used in lieu of this operation. Refer to example d, Figure 5-8.

a. Depress the EN switch.
b. Enter all eight digits of the Communication register by depressing the appropriate numeric key switches.
c. Depress the SW/EN CONT switch.
d. Depress the STOP switch.
e. Depress the KEYBOARD OFF switch.

7. The following procedure is applicable for sweeping storage during certain maintenance routines:
   a. Depress the SW switch.
   b. Depress the SW/EN CONT switch. The switch remains depressed until the STOP switch is depressed.
   c. Depress the STOP switch.
   d. Depress the KEYBOARD OFF switch.

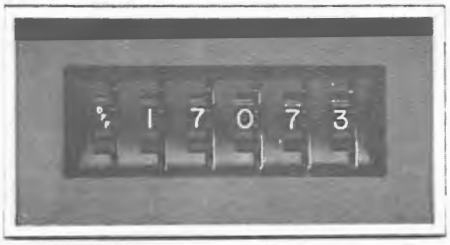
Examples of Console Switch Functions

1. To enter a special routine into the Auto Load storage area:
   a. Depress the MC (Master Clear) keyboard switch.
   b. Holding down the keyboard STOP switch, depress the AUTO LOAD switch. Release both switches. The P register should now read 77640. (Holding the STOP switch down, prevents the computer from entering the GO mode and executing the previous Auto Load routine.)
   c. Depress the ENTER AUTO PROGRAM switch.
   d. Depress the keyboard EN switch.
   e. Enter the first instruction of the new routine at address 77640 by depressing the appropriate numeric key switches.
   f. Depress the keyboard TRANSFER switch.
   g. Repeat steps *e* and *f* for addresses 77641 through 77657.
   h. Depress the MC switch. This clears the registers and cancels the ENTER AUTO PROGRAM function.
   i. Depress the KEYBOARD OFF switch.
2. To enter a special routine into the Auto Dump storage area:

   Repeat steps *a* through *i* of example 1 using the AUTO DUMP switch and filling the storage area covered by addresses 77660 through 77677.
3. To execute the Auto Load routine:
   a. Depress the keyboard MC switch.
   b. Depress the AUTO LOAD switch. The computer automatically executes the Auto Load routine and stops when a stop or halt instruction is recognized. The Auto Load function is automatically cleared when the computer stops.
4. To execute the Auto Dump routine:

   Perform steps *a* and *b* in example 3 but use the AUTO DUMP switch instead of the AUTO LOAD switch.
5. To execute a program at an Auto Step rate:
   a. Set the P register to the first address of the program to be executed.
   b. Depress the AUTO STEP switch.
   c. Adjust the AUTO STEP display rate control.
   d. Depress the AUTO STEP switch again to cancel the function and stop program execution. The only way to exit from the Auto Step mode is to depress the AUTO STEP switch again. In the Auto Step mode, halt and jump instructions are executed but the computer will not stop. Neither will program execution be affected by depressing the STOP switch. The computer will continue cycling through memory until the AUTO STEP switch is again depressed.

EXAMPLE A

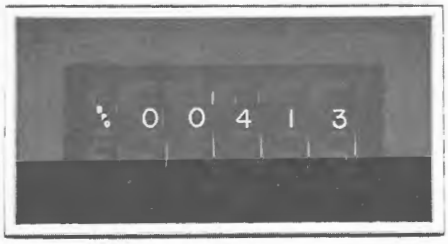


The Breakpoint switch is inoperative whenever an OFF designator is displayed. An OFF designator separates the REG, STO, BPJ and BPO positions.

EXAMPLE B

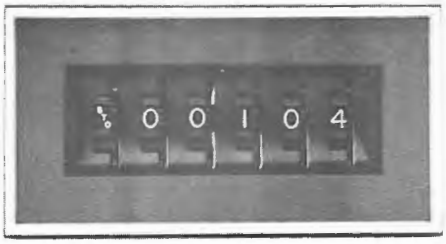
During the normal execution of a program, the computer stops when an RNI is attempted at memory location 05443. A jump to this location also causes the computer to stop. If the program references memory location 05443 for an operand, the computer ignores the Breakpoint switch.

EXAMPLE C



The computer stops only when an attempt is made to read or store an operand at address 00413.

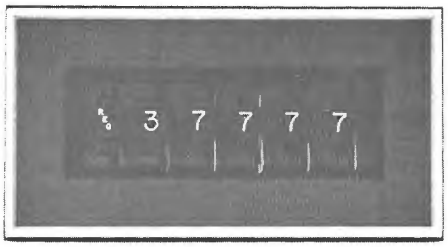
EXAMPLE D



If the WRITE STO switch on the keyboard is depressed and data has been entered into the Communication register, the data is transferred to memory location 00104 when the TRANSFER switch is depressed.

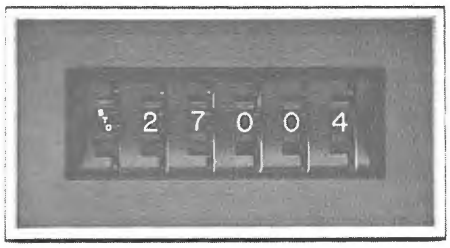
Figure 5-8. Breakpoint Switch Examples

EXAMPLE E



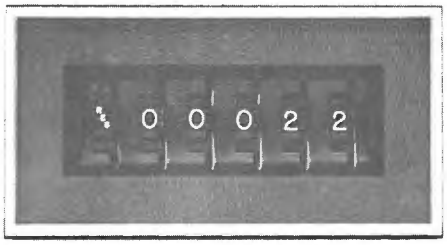
If the WRITE STO switch on the keyboard is depressed and data has been entered into the Communication register, the data will be transferred to register 77 when the TRANSFER switch is depressed. (Only the lower two digits are recognized when the designator switch is in the REG position. The programmer must use caution when writing into the Register File to prevent destruction of other data. Refer to Section 1, Table 1-3.)

EXAMPLE F



If the READ STO switch on the keyboard is depressed, the contents of memory location 27004 are displayed in the Communication register at a repetition rate determined by the Step Rate control. (If the memory location depicted by the Breakpoint switch exceeds the storage capacity of the system, the computer selects the address that corresponds to the storage capacity of the system.)

EXAMPLE G



If the READ STO switch on the keyboard is depressed, the contents of register 22 are displayed in the Communication register at a repetition rate determined by the Step Rate control. (Only the lower two digits are of consequence when the REG designator is displayed. In this case register 22, the real time clock, is being referenced.)

Figure 5-8. Breakpoint Switch Examples (Cont.)

POWER CONTROL PANEL

Power for the 3100 Computer System is controlled by the Power Control Panel, mounted on the right side of the main cabinet assembly. The switches, circuit breakers, indicators and meters associated with the panel are shown in Figure 5-9. Refer to Part 5 of the 3100 Customer Engineering manual for detailed maintenance information concerning the Power Control Panel.

SWITCHES

Table 5-3 lists the switches and their functions. Refer to Section 2 for a description of the Storage Address Protection switches.

ELAPSED TIME METERS

Two elapsed time meters and a key-operated, two-position switch are located on the control panel. Turning the key-operated Maintenance Mode switch to ON connects the Maintenance Time meter to the computer to record the amount of time the computer is used during a maintenance period. Removing the key connects the Operating Time meter to the computer to record normal operating time. Customers renting the computer are often billed according to the time recorded on this meter. The sum of the times recorded on both meters indicates the total computer running time. Only one of the two meters can operate at any one time. Either meter is active for a minimum of one second when a storage cycle occurs.

TABLE 5-3. POWER CONTROL PANEL SWITCH FUNCTIONS

SWITCH NAME	FUNCTION
CONTROL POWER	When this switch is depressed, the Blower switch and Peripheral Group switches can be activated.
BLOWERS ON	Depressing this switch turns on cabinet blowers, power supply blowers and furnishes power for the peripheral equipment blowers. This switch must be on before the power supplies can be activated. The Control Power switch must be on before this switch can be activated.
POWER SUPPLIES ON	When this switch is depressed and the Control Power and Blowers switches are on, the motor generators are turned on. These sets furnish operating power for the logic power supplies.
PERIPHERAL GROUP I ON	If the Control Power switch is on and this switch is depressed, operating power is sent to all the equipment connected to the Peripheral Group I power distribution bus.
PERIPHERAL GROUP II ON	If the Control Power switch is on and this switch is depressed, operating power is sent to all of the equipment connected to the Peripheral Group II power distribution bus.

NOTES

1. **The switches are active only when main power is present at the control panel and the applicable circuit breakers are closed (ON position). The individual circuit breakers are located directly below the switch panel.**
2. **Except for the Blowers switch, the OFF switches remove power immediately. If the Power Supplies OFF and the Blowers OFF switches are depressed in close succession, an automatic five minute delay will keep the blowers operating. The Power Supplies OFF switch must be depressed a minimum of half a second.**

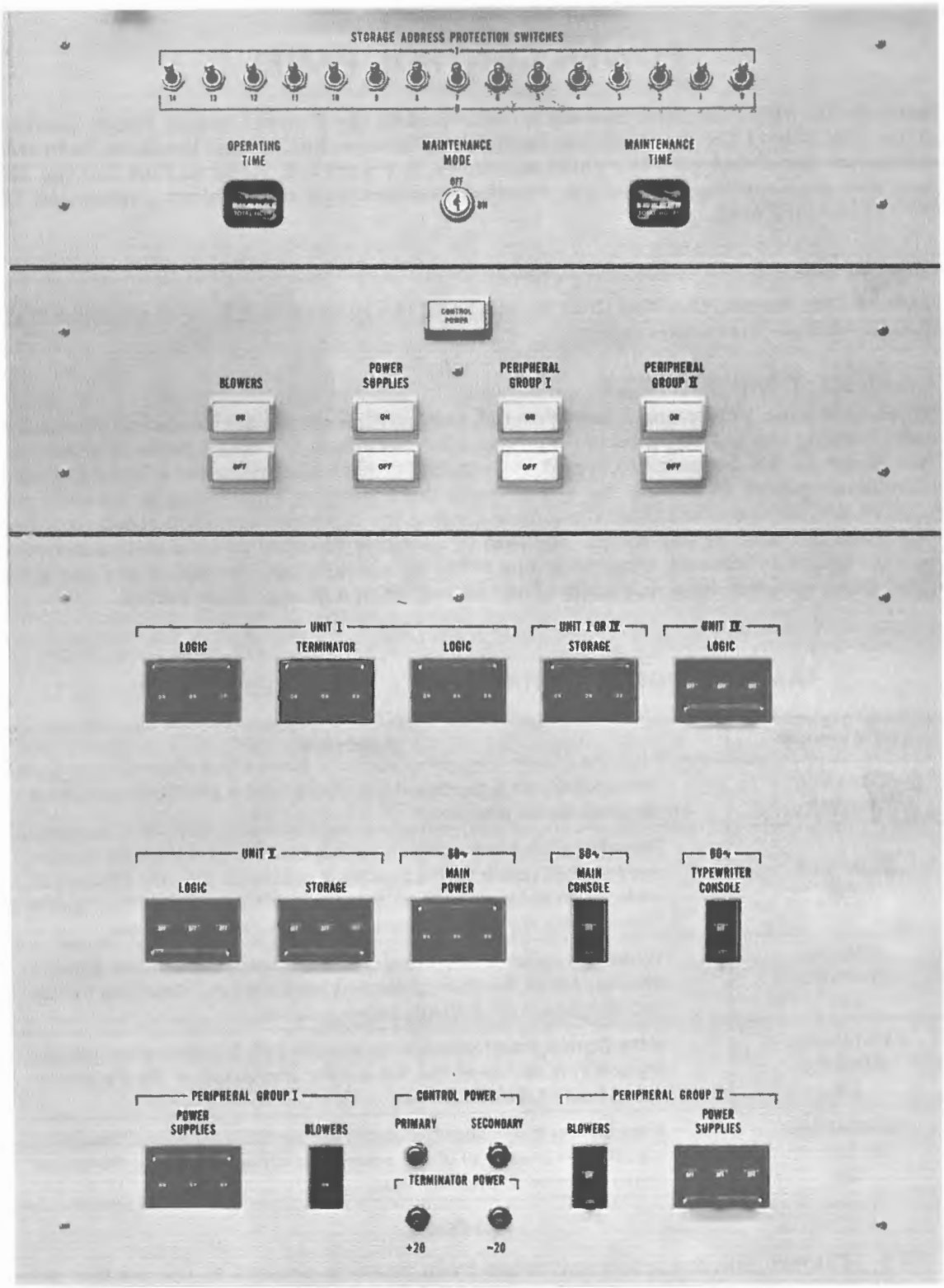


Figure 5-9. Power Control Panel

Section 6

TYPEWRITER DESCRIPTION

The 3192 Console Typewriter (Figure 6-1) is an on-line input-output (I/O) device; *i.e.* it requires no connection to a communication channel and no function codes are issued. The typewriter receives output data directly from storage via the lower 6 bits of the Data Bus. Inputs to storage are handled in the same manner.

The console typewriter consists of an electric typewriter and a typewriter control panel mounted on a desk console.

Figure 6-1. 3192 Console Typewriter

Used in conjunction with block control and the Register File, the typewriter may be used to enter a block of internal binary-coded characters into storage and to print out data from storage. The two storage addresses that define the limits of the block must be stored in the register file prior to an input or output operation. Register 23* contains the initial character address of the block, and register 33 contains the last character address, plus one. Because the initial character address is incremented for each storage reference, it always shows the address of the character currently being stored or dumped. Output operations occur at the rate of 15 characters per second. Input operations are limited by the operator's typing speed. In systems using the optional 3101 desk console, the 3192 is an integral part of the console.

OPERATION

The general order of events when using the console typewriter for an input or output operation is:

1. Set tabs, margins and spacing. Turn on typewriter.
2. Clear.
3. Check status.
4. Type out or type in.

SET TABS, MARGINS, AND SPACING

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

CLEAR

There are three types of clears which may be used to clear all conditions (except Encode Function) existing in the typewriter control. These are:

- Internal Clear or a Master Clear.
  This signal clears all external equipments, the communication channels, the typewriter control, and sets the typewriter to lower case.
- Clear Channel, Search/Move Control, or Type Control instruction (77.51). This instruction selectively clears a channel, the S/M control, or, by placing a "1" in bit 08 of the instruction, the typewriter control, and sets the typewriter to lower case.
- Clear Switch on typewriter.
  This switch clears the typewriter control and sets the typewriter to lower case.

STATUS CHECKING

The programmer may wish to check the status of the typewriter before proceeding. This is done with the Pause instruction. Status response is returned to the computer via two status lines.

The typewriter control transmits two status signals that are checked by the Busy Comparison Mask using the Pause instruction. These status signals are:

Bit 09 Type Finish
Bit 10 Type Repeat

An additional status bit appears on sense line 08. This code is Type Busy, and is transmitted by block control in the computation section when a typewriter operation has been selected. If the programmer is certain of the status of the typewriter, this operation may be omitted.

*The upper nine bits of registers 23 and 33 should be "0"

TYPE IN AND TYPE LOAD

The Set Type In instruction or pressing the TYPE LOAD switch on the console or typewriter permits the operator to enter data directly into storage from the typewriter. When the TYPE LOAD indicator on the console or typewriter glows, the operator may begin typing. The Encode Function switch must be depressed to enable backspace, tab, carriage return, and case shifts to be transmitted to the computer during a typewriter input operation.

Input is in character mode only. As each character is typed, the information is transmitted via the Data Bus to the storage address specified by block control. This address is incremented as characters are transmitted. When the current address equals the terminating address, the TYPE LOAD indicator goes off and the operation is terminated. Data is lost if the operator continues typing after the TYPE LOAD indicator goes off.

TYPE OUT AND TYPE DUMP

The typewriter begins to type out when the computation section senses a Set Type Out instruction or the operator presses the TYPE DUMP switch on the console or typewriter. Single 6-bit characters are sent from storage to the typewriter via the lower 6 bits of the Data Bus. When the current address equals the terminating address, the TYPE DUMP indicator goes off and the operation is terminated.

During a Type Out operation, the keyboard is locked to prevent loss of data in the event a key is accidentally pressed.

CONSOLE SWITCHES AND INDICATORS

Figure 6-2 shows the switch arrangement of the typewriter control panel. The function of each switch appears in Table 6-1. A rocker switch on the typewriter unit is used to apply power to the typewriter motor.

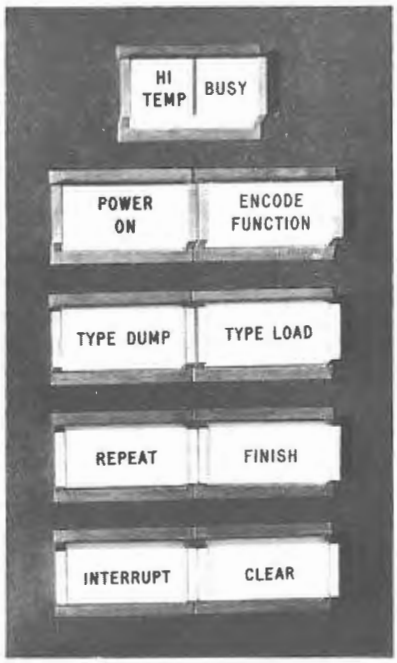


Figure 6-2. Typewriter Control Panel.

TABLE 6-1. CONSOLE TYPEWRITER SWITCHES AND INDICATORS

Name	Switch (S) Indicator (I)	Description
HIGH TEMP	I	This indicator glows when the ambient temperature within the typewriter cabinet exceeds 110° F.
BUSY	I	This indicator shows that the TYPE LOAD or TYPE DUMP switch has been pressed and the operation is in progress.
POWER ON	I	This indicator shows that power is applied to the typewriter.
TYPE DUMP	S & I	This switch is in parallel with the TYPE DUMP switch on the console and causes the computer to send data to the typewriter for print-out. It is a momentary contact switch that is illuminated until the last character in the block has been printed or the CLEAR button is pressed.
TYPE LOAD	S & I	This switch is in parallel with the TYPE LOAD switch on the console and allows the computer to receive a block of input data from the typewriter. The TYPE LOAD indicator remains on until either the FINISH, REPEAT or CLEAR button is pressed, or until the last character of the block has been stored. If the program immediately reactivates the typewriter, it may appear that the light does not go off.
REPEAT	S & I	This switch is pressed during a Type Load operation to indicate that a typing error occurred. This switch deactivates busy sense line 10 (see PAUS instruction). If the computer does not respond, this light remains on.
FINISH	S & I	This switch is pressed during a Type Load operation to indicate that there is no more data in the current block. This action is necessary if the block that the operator has entered is smaller than the block defined by registers 23 and 33. This switch also deactivates busy sense line 09. If the computer does not respond, this light remains on.
INTERRUPT	S & I	This switch is in parallel with the MANUAL INTERRUPT switch on the console and is used to manually interrupt the computer program.
ENCODE FUNCTION	S & I	This switch enables the typewriter to send to storage the special function codes for backspace, tab, carriage return, upper-case shift, and lower-case shift.
CLEAR	S & I	This switch clears the typewriter controls and sets the typewriter to lower case but does not cancel Encode Function.

CHARACTER CODES

Table 6-2 lists the internal BCD codes, typewriter printout and upper- or lower-case shift that applies to the console typewriter. All character transmission between the computation section and the typewriter is in the form of internal BCD. The typewriter logic makes the necessary conversion to the machine code.

NOTE

Shifting to upper case (57) or lower case (32) is not necessary except on keyboard letters where both upper and lower cases are available. The standard type set for the 3192 has two sets of upper case letters and no lower case letters. This eliminates the need for specifying a case shift.

TABLE 6-2. CONSOLE TYPEWRITER CODES

Print-out	Case	Internal BCD Code
0	L	00
1	L	01
2	L	02
3	L	03
4	L	04
5	L	05
6	L	06
7	L	07
8	L	10
9	L	11
±	U	12
=	L	13
"	U	14
:	U	15
;	L	16
?	U	17
+	U	20
A	U or L	21
B	U or L	22
C	U or L	23
D	U or L	24
E	U or L	25
F	U or L	26
G	U or L	27
H	U or L	30
I	U or L	31
(Shift to LC)		32
.	U and L	33
)	U	34
'	L	35
@	U	36
!	L	37

Print-out	Case	Internal BCD Code
-	L	40
J	U or L	41
K	U or L	42
L	U or L	43
M	U or L	44
N	U or L	45
O	U or L	46
P	U or L	47
Q	U or L	50
R	U or L	51
° (degree)	U	52
$	U	53
*	U	54
#	U	55
%	U	56
(Shift to UC)		57
(Space)		60
/	L	61
S	U or L	62
T	U or L	63
U	U or L	64
V	U or L	65
W	U or L	66
X	U or L	67
Y	U or L	70
Z	U or L	71
&	U	72
,	U and L	73
(	U	74
(Tab)		75
(Backspace)		76
(Carriage return)		77

Section 7
INSTRUCTIONS

GENERAL INFORMATION

INSTRUCTION WORD FORMATS

The standard 3100 machine coded instruction is 24 bits in length and generally classified into one of two formats: word or character oriented.

Word oriented instructions are the most common of the instruction formats. Fifteen bits are allocated for an unmodified storage address, operand, or shift count. Indirect addressing is usually available. Figure 7-1 illustrates a word oriented instruction and the significance of the first 15 bits when they represent an unmodified word address 'm'.

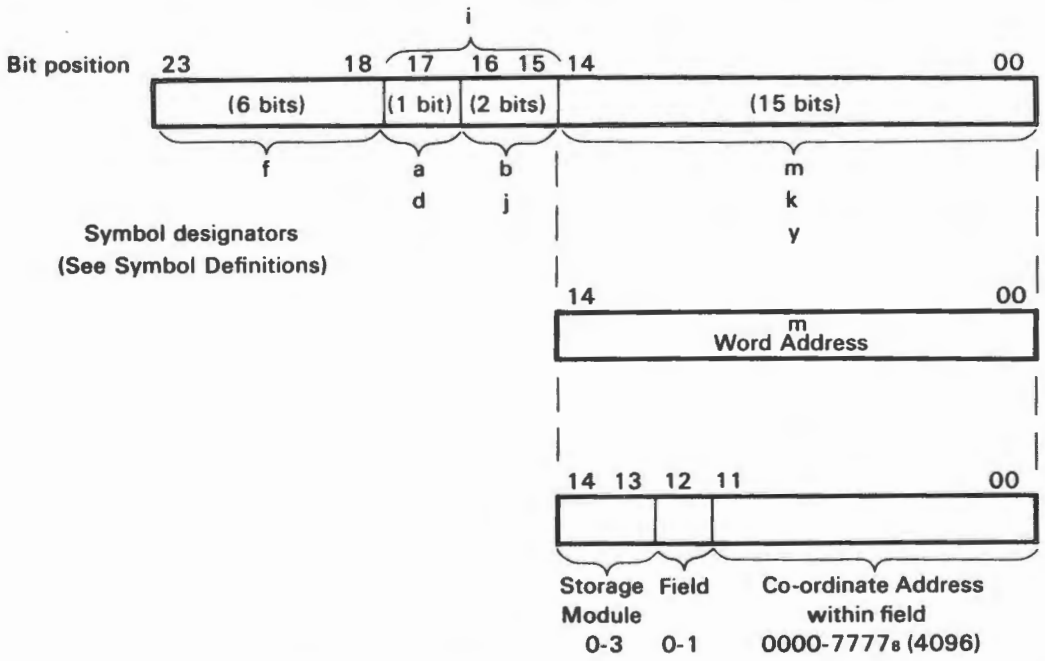


Figure 7-1. Word-Addressed Instruction Format

Character oriented instructions allocate 17 bits for unmodified character addresses or extended operands. Indirect addressing is not available for these instructions; however, address modification is permissible by referencing a specific index register. Figure 7-2 illustrates the format of a character oriented instruction word and the significance of the first 17 bits when they represent an unmodified character address 'r'.

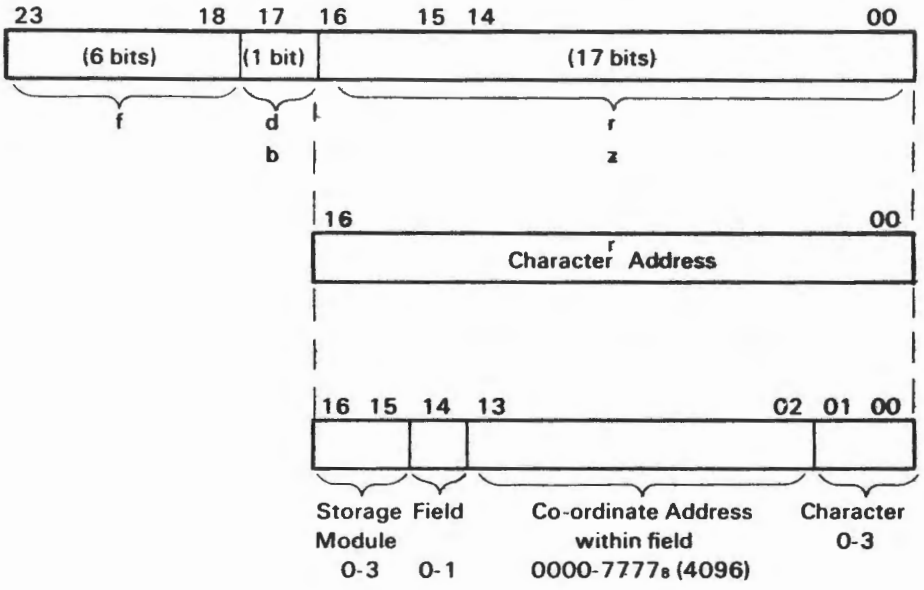
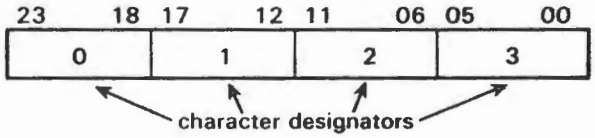


Figure 7-2. Character-Addressed Instruction Format

Characters in a data word are always specified in the following manner:



WORD ADDRESSES VS. CHARACTER ADDRESSES

It is often desirable to convert a word address and character position to its corresponding character address or vice versa. The following procedure is a technique used for this purpose:

To convert a word address to a character address:

- Octally multiply the word address by four. (During program execution, this operation is simulated by a left shift of two binary places.)
- Add the character position to the product.

The sum will be the character address.

EXAMPLE: Given: Word address 12442, character position 2
Find: Corresponding character address.

1. 12442
   x4
   52210
2. +2
   52212 = character address

To convert a character address to a word address:

- Octally divide the character address by four.

The quotient will be the word address and the remainder is the character position. No remainder indicates character zero.

EXAMPLE: Given: Character address 03442
Find: Word address and character position.

```
   00710
 4|03442
   34
    4
    4
```

2 = remainder = character position 2.

NOTE

Octal multiplication and division tables may be found in the appendix section of this manual.

Instruction word formats that differ from word and character orientation are described in the instruction listing.

SYMBOL DEFINITIONS

The following designators are used throughout the list of instructions. Additional special symbols are used in Search/Move and certain I/O instructions and are defined where they are used.

a = addressing mode designator (a=0, direct addressing; a=1, indirect addressing)
b = index designator (unless otherwise stated)
c = denotes a character code or field
ch = denotes an I/O channel (0-3)
d = special operation designator (see individual instructions).
f = function code (6 bits, octal 00 to 77)
H = instruction modifier for INPC or OUTC indicating 6 or 12 bit I/O operation
i = interval designator (decrement quantity)
j = jump, stop, or skip condition designator (see individual instructions)
k = shift count (unmodified)
m = word execution address (unmodified)
n = same as m, but the word address of the second operand
r = character execution address (unmodified)
s = same as r, but the character address of the second operand
S = instruction modifier denoting sign extension
  S present, bit 17="1", sign extended
  S absent, bit 17="0", sign not extended
v = a specific register (00-77) within the Register File.
x = connect code or interrupt mask
y = 15-bit operand
z = 17-bit operand
//////// = indicates zeros should be loaded into a particular area of an instruction.

INDEXING AND ADDRESS MODIFICATION

In some instructions, the execution address 'm' or 'r', or the shift count 'k' may be modified by adding to them the contents of an index register, B^b. The 2-bit designator 'b' specifies which of the three index registers is to be used. Symbols representing the respective modified quantities are M, R, and K.

$M = m + (B^b)$
$R = r + (B^b)$ the sign of B^b is extended to bit 16 (2^{17}-1)
$K = k + (B^b)$

In each case, if b=0, then M=m, R=r and K=k.

ADDRESSING MODES

Three modes of addressing are used in the computer: No Address, Direct Address, and Indirect Address.

No Address

This mode is used when an operand 'y' or a shift count 'k' is placed directly into the lower portion of an instruction word. Symbols 'a' and 'b' are not used as addressing mode and index designators with any of the no address instructions.

Direct Address

The direct addressing mode is used in any instruction in which an operand address 'm' is stored in the lower portion of the initial instruction word. This mode is specified by making 'a' equal to 0. In many instructions, address 'm' may be modified (indexed) by adding to it the contents of register B^b, $M = m + (B^b)$.

Indirect Address

It is possible to use indirect addressing only with instructions that require an execution address 'm'. For applicable instructions, indirect addressing is specified by making 'a' equal to 1. Several levels (or steps) of indirect addressing may be used to reach the execution address; however, execution time is delayed in direct proportion to the number of steps. The search for a final execution address continues until 'a' equals 0. It is important to note that direct or indirect addressing and address modification are two distinct and independent steps. In any particular instruction, one may be specified without the other. Figure 7-3 shows the indirect addressing routine for a 3100 Computer.

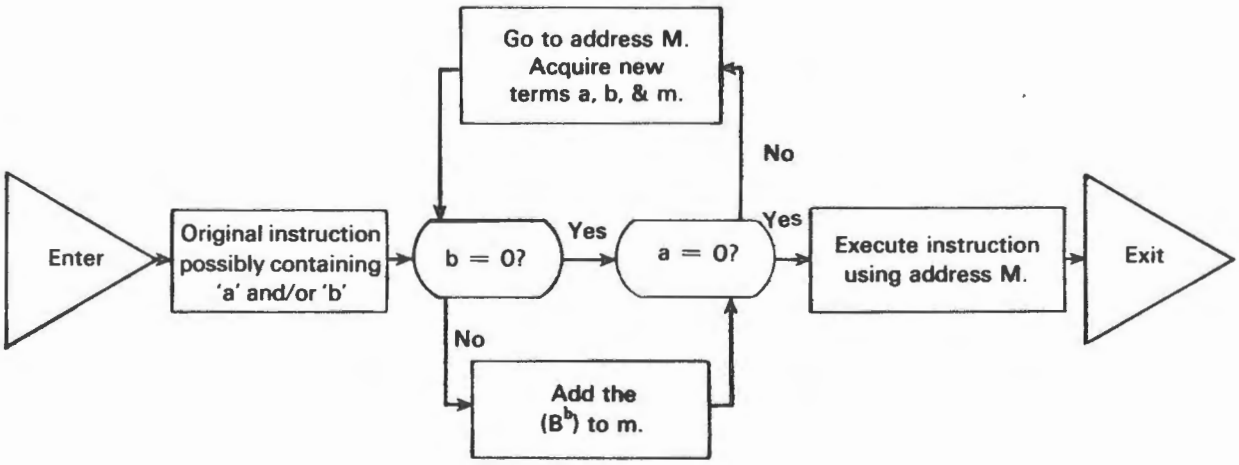


Figure 7-3. Indexing and Indirect Addressing Routine Flow Chart

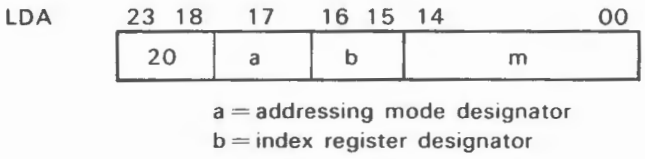
NOTE

Unless it is otherwise stated, indirect addressing follows the above routine throughout the list of instructions.

INDEXING AND ADDRESSING MODE EXAMPLES

The following examples utilize the LDA (20) instruction; however, the process applies to any of the instructions with an 'a' and/or 'b' designator.

LDA	23 18	17	16 15	14 00
	20	a	b	m



a = addressing mode designator
b = index register designator

EXAMPLE 1
(ADDRESS MODIFICATION — (indexing) ONLY)

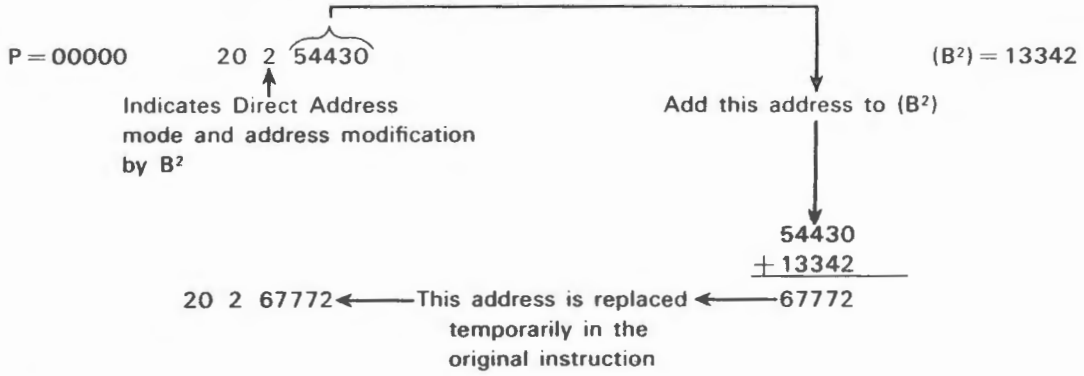
P = 00000 20 2 54430 (B²) = 13342

Indicates Direct Address mode and address modification by B²

Add this address to (B²)

```
  54430
+ 13342
-------
  67772
```

20 2 67772 ← This address is replaced temporarily in the original instruction ← 67772



LDA with the 24-bit quantity stored at address 67772

```
P =  67771   --------
     67772   77700000 → This quantity is loaded into the A register
     67773   --------
```

EXAMPLE 2
(INDIRECT ADDRESSING ONLY)

P = 00001 20 4 54430

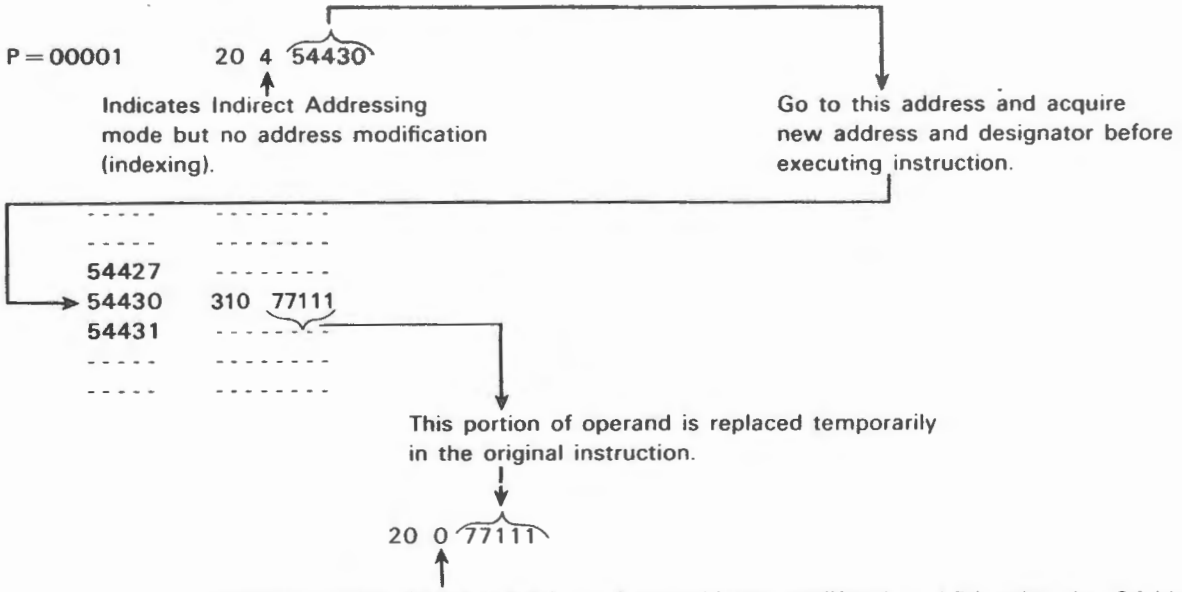
Indicates Indirect Addressing mode but no address modification (indexing).

Go to this address and acquire new address and designator before executing instruction.

```
  -----   --------
  -----   --------
  54427   --------
→ 54430   310 77111
  54431   --------
  -----   --------
  -----   --------
```

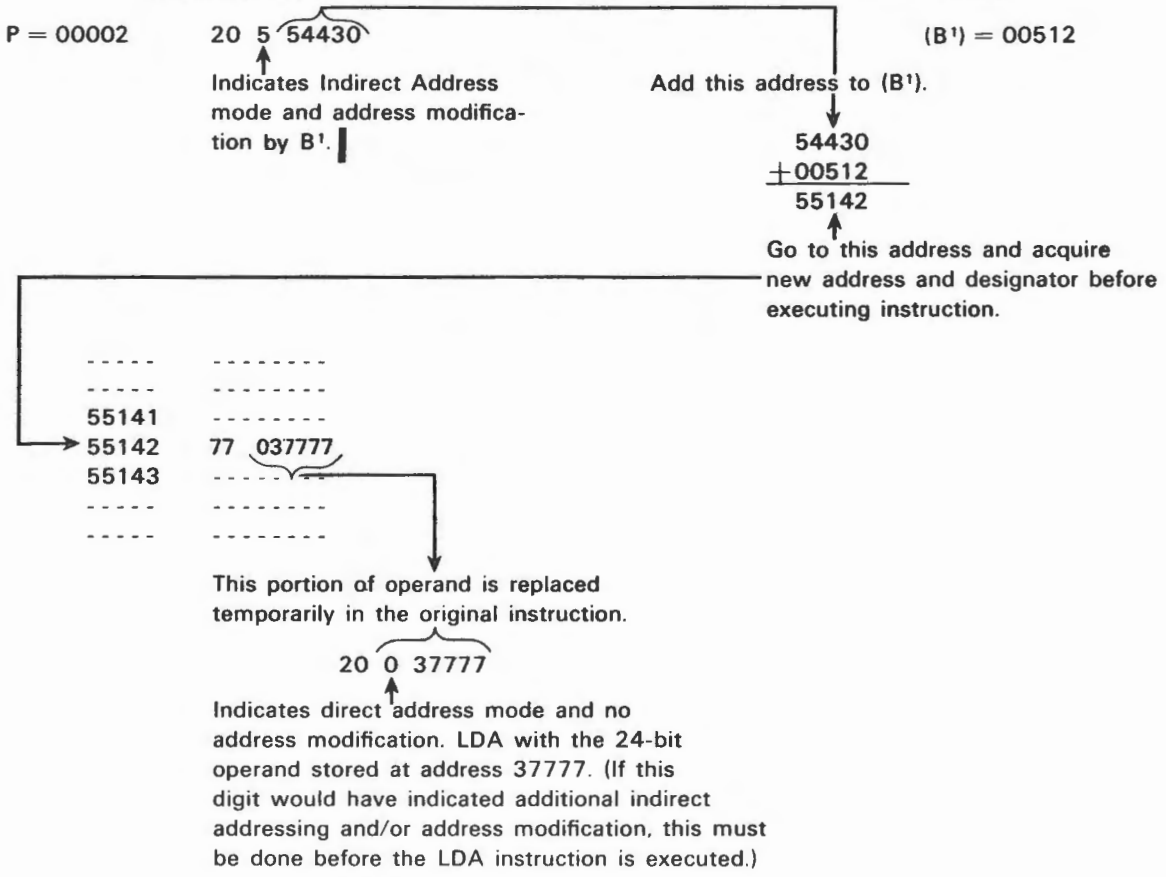
This portion of operand is replaced temporarily in the original instruction.

20 0 77111



Indicates Direct Address mode and no address modification. LDA with the 24-bit operand stored at address 77111. (If this digit would have indicated additional indirect addressing and/or address modification this must be done before the LDA instruction is executed.)

EXAMPLE 3
(INDIRECT ADDRESSING AND ADDRESS MODIFICATION)



I/O Channel Considerations

The 3100 Computer may have a maximum of four I/O channels. To preserve programming compatibility with the 3200 Computer, eight instruction bits are allocated for channel selection; however, only the first four bits are utilized in 3100 Systems. If channel 4, 5, 6 or 7 is addressed in a 3100 System, it is recognized as a nonexistent channel.

Trapped Instructions

The instructions appearing in Table 7-1 are executed by the Utility System under the control of SCOPE. The Basic Utility software system also is capable of executing these instructions.

The computer detects the 55-70 instructions as they appear in the F register and traps them if the BCD and Floating Point 48-bit Precision hardware is absent. Trapped instructions are processed as interrupts once they are detected. A conventional interrupt always takes priority over the trap sequence. The following operations occur when a trapped instruction is detected:

1. P + 1 is stored in the lower 15 bits of address 00010.
2. The upper 6 bits of F are stored in the lower 6 bits of address 00011; the upper 18 bits remain unchanged.
3. Program control is transferred to address 00011 and an RNI cycle is executed.

TABLE 7-1.
LIST OF TRAPPED INSTRUCTIONS

Operation Field		Interpretation
55	- - - -	I.R.T., 48-bit precision
56	MUAQ	Multiply AQ, 48-bit precision
57	DVAQ	Divide AQ, 48-bit precision
60	FAD	Floating point add
61	FSB	Floating point subtract
62	FMU	Floating point multiply
63	FDV	Floating point divide
64	LDE	Load E_D
65	STE	Store E_D
66	ADE	Add to E_D
67	SBE	Subtract from E_D
70	SFE	Shift E_D
	EZJ,EQ	E_D zero jump, $E_D = 0$
	EZJ,LT	E_D zero jump, $E_D < 0$
	EOJ	E_D overflow jump
	SET	Set D register

INSTRUCTION LIST

Each group of instructions is introduced with an index and, whenever necessary, a group description. Individual instructions are all presented in the same basic format:

- Heading, which includes the assembly language mnemonic and instruction name
- Machine code instruction format
- Instruction description
- Comments (when necessary)
- Approximate instruction execution time (add 1.75 usec for each step of indirect addressing)

The abbreviation, RNI, is used throughout the list of instructions to indicate the Read Next Instruction sequence. This is a sequence of steps taken by the control section to advance the computer to its next program step. For an extensive description of this sequence, consult the 3100 Customer Engineering Manual, publication number 60111200.

Table 7-2 identifies the instructions and indicates on which page explicit instruction descriptions may be found. Table 7-3 is a summary of the instruction execution times. In addition to these tables, three additional tables are provided at the end of this manual for cross reference of the instruction list.

TABLE 7-2. INSTRUCTION SYNOPSIS AND INDEX

MNEMONIC	INSTRUCTION	PAGE
ADA, I	add to A	7-38
ADAQ, I	add to AQ	7-40
ADE	add to E	7-47
AEU	transmit (A) to E upper	7-29
AIA	transmit (A) + (B^b) to A	7-26
ANA, S	logical product (AND) of y and (A)	7-18
ANI	logical product (AND) of y and (B^b)	7-18
ANQ, S	logical product (AND) of y and (Q)	7-18
AQA	transmit (A) + (Q) to A	7-26
AQE	transmit (AQ) to E	7-29
AQJ, EQ	compare A with Q: jump if (A) = Q	7-36
NE	compare A with Q: jump if (A) ≠ Q	7-36
GE	compare A with Q: jump if (A) ≥ Q	7-36
LT	compare A with Q: jump if (A) < Q	7-36
ASE, S	skip next instruction, if (A) = y	7-13
ASG, S	skip next instruction, if (A) ≥ y	7-14
AZJ, EQ	compare A with zero: jump if (A) = 0	7-35
NE	compare A with zero: jump if (A) ≠ 0	7-35
GE	compare A with zero: jump if (A) ≥ 0	7-35
LT	compare A with zero: jump if (A) < 0	7-35
CILO	channel interrupt lockout	7-69
CINS	copy internal status	7-62
CLCA	clear channel activity	7-69
CON	connect	7-70
COPY	copy external status	7-60
CPR, I	within limits test	7-53
CTI	set console typewriter input	7-71
CTO	set console typewriter output	7-71
DINT	disable interrupt control	7-67
DVA, I	divide AQ (48 by 24)	7-39
DVAQ, I	divide AQE (96 by 48)	7-42
EAQ	transmit (E upper) to A and (E lower) to Q	7-29
ECHA, S	enter A with 17-bit character address	7-15
EINT	enable interrupt control	7-67
ELQ	transmit (E lower) to Q	7-29
ENA	enter A	7-15
ENI	enter index	7-15
ENQ	enter Q	7-15
EOJ	jump to m on E overflow	7-49
EUA	transmit (E upper) to A	7-29
EXS	sense external status	7-60
EZJ, EQ	compare E with zero; jump if E = 0	7-49
LT	compare E with zero; jump if E < 0	7-49
FAD, I	floating add to AQ	7-43
FDV, I	floating divide AQ	7-44
FMU, I	floating multiply AQ	7-44
FSB, I	floating subtract from AQ	7-44

TABLE 7-2. INSTRUCTION SYNOPSIS AND INDEX (CONTINUED)

MNEMONIC	INSTRUCTION	PAGE
HLT	unconditional stop; read next instruction from location m	7-30
IAI	transmit $(B^b) + (A)$ to B^b	7-26
IAPR	interrupt associated processor	7-66
IJD	index jump; decrement index	7-34
IJI	index jump; increment index	7-33
INA	increasé A	7-16
INAC, INT	character-addressed input to A	7-80
INAW, INT	word-addressed input to A	7-82
INCL	clear interrupt	7-65
INI	increase index	7-16
INPC, INT, B, H	character-addressed input to storage	7-72
INPW, INT, B, N	word-addressed input to storage	7-74
INQ	increase Q	7-16
INS	sense internal status	7-62
INTS	sense interrupt	7-61
IOCL	clear I/O, typewriter, and S/M	7-63
ISD	index skip; decrement index	7-19
ISE	skip next instruction, if $(B^b) = y$	7-13
ISG	skip next instruction, if $(B^b) \geq y$	7-14
ISI	index skip; increment index	7-19
LACH	load A character	7-20
LCA, I	load A complement	7-21
LCAQ, I	load AQ complement (double precision)	7-21
LDA, I	load A	7-20
LDAQ, I	load AQ (double precision)	7-21
LDE	load E	7-48
LDI, I	load index	7-22
LDL, I	load ogical	7-21
LDQ, I	load Q	7-22
LPA, I	logical product with A	7-37
LQCH	load Q character	7-22
MEQ	masked equality search	7-54
MOVE, INT	move *l* characters from r to s	7-58
MTH	masked threshold search	7-55
MUA, I	multiply A	7-39
MUAQ, I	multiply AQ	7-42
OTAC, INT	character-addressed output from A	7-84
OTAW, INT	word-addressed output from A	7-86
OUTC, INT, B, H,	character-addressed output from storage	7-76
OUTW, INT, B, N	word-addressed output from storage	7-78
PAUS	pause	7-64
QEL	transmit (Q) to E lower	7-29
QSE, S	skip next instruction, if $(Q) = y$	7-13
QSG, S	skip next instruction, if $(Q) \geq y$	7-14
RAD, I	replace add	7-38
RTJ	return jump	7-32

TABLE 7-2. INSTRUCTION SYNOPSIS AND INDEX (CONTINUED)

MNEMONIC	INSTRUCTION	PAGE
SACH	store character from A	7-23
SBA, I	subtract from A	7-39
SBAQ, I	subtract from AQ	7-40
SBCD	set BCD fault	7-67
SBE	subtract from E	7-47
SCA, I	selectively complement A	7-37
SCAQ	scale AQ	7-52
SCHA, I	store 17-bit character address from A	7-25
SCIM	selectively clear interrupt mask	7-66
SEL	select function	7-70
SET	set D to value of y	7-46
SFE	shift E	7-49
SFPF	set floating point fault	7-67
SHA	shift A	7-50
SHAQ	shift AQ	7-52
SHQ	shift Q	7-52
SJ1	jump if key 1 is set	7-31
SJ2	jump if key 2 is set	7-31
SJ3	jump if key 3 is set	7-31
SJ4	jump if key 4 is set	7-31
SJ5	jump if key 5 is set	7-31
SJ6	jump if key 6 is set	7-31
SLS	selective stop	7-31
SQCH	store character from Q	7-24
SRCE, INT	search character equality.	7-56
SRCN, INT	search character inequality	7-56
SSA, I	selectively set A	7-37
SSH	storage shift	7-50
SSIM	selectively set interrupt mask	7-66
STA, I	store A	7-23
STAQ, I	store AQ	7-24
STE	store E	7-48
STI, I	store index	7-25
STQ, I	store Q	7-24
SWA, I	store 15-bit word address from A	7-25
TAI	transmit (A) to B^b	7-27
TAM	transmit (A) to high speed memory	7-28
TIA	transmit (B^b) to A	7-27
TIM	transmit (B^b) to high speed memory	7-28
TMA	transmit (high speed memory) to A	7-28
TMI	transmit (high speed memory) to B^b	7-28
TMQ	transmit (high speed memory) to Q	7-27
TQM	transmit (Q) to high speed memory	7-27
UCS	unconditional stop	7-31
UJP, I	unconditional jump	7-32
XOA, S	exclusive OR y and (A)	7-17
XOI	exclusive OR y and (B^b)	7-17
XOQ, S	exclusive OR y and (Q)	7-17

TABLE 7-3. SUMMARY OF INSTRUCTION EXECUTION TIMES, μsec.

INSTRUCTION MNEMONIC	EXECUTION TIME	INSTRUCTION MNEMONIC	EXECUTION TIME
ADA	3.5	INQ	1.8
ADAQ	5.2	INS	2.2
ADE	16.1*	INTS	2.3
AEU	1.8-6.1*	IOCL	1.8
AIA	1.8	ISD	2.6
ANA	1.8	ISE	2.6
ANI	1.8	ISG	2.6
ANQ	1.8	ISI	2.6
AQA	1.8		
AQE	1.8-6.1*	LACH	3.5
AQJ	2.6	LCA	3.5
ASE	2 6	LCAQ	5.2
ASG	2.6	LDA	3.5
AZJ	2.6	LDAQ	5.2
CILO	1.8	LDE	10.8*
CINS	2.2	LDI	3.2
CLCA	1.8	LDL	3.2
CON	2.3***	LDQ	3.2
COPY	2.2	LPA	3.2
CPR	3.5 or 4.3	LQCH	3.2
CTI	1.8		
CTO	1.8	MEQ	6.3 + 6.3n
DINT	1.8	MOVE	7.0 + 10.4n
DVA	15.0	MTH	6.3 + 6.3n
DVAQ	31.5*	MUA	11.5-15.0
		MUAQ	22.4-29.4*
EAQ	1.8-6.1*		
ECHA	1.8	OTAC	7.0
EINT	1.8	OTAW	7.0
ELQ	1.8-6.1	OUTC	7.0 + 7.0n
ENA	1.8	OUTW	7.0 + 7.0n
ENI	1.8		
ENQ	1.8	PAUS	1.8 us-40 ms
EOJ	1.8*		
EUA	1.8-6.1	QEL	1.8-6.1
EXS	2.2	QSE	2.6
EZJ	1.8*	QSG	2.6
FAD	14-16.8*	RAD	5.2
FDV	28.0*	RTJ	3.6
FMU	19.6-25.2*		
FSB	14-16.8*		
		SACH	3.5
HLT	1.8	SBA	3.5
		SBAQ	5.2
IAI	1.8	SBCD	1.8
IAPR	**	SBE	16.1*
IJD	2.6	SCA	3.5
IJI	2.6	SCAQ	2.6-5.1
INA	1.8	SCHA	3.5
INAC	7.0***	SCIM	1.8
INAW	7.0***	SEL	2.3***
INCL	1.8	SET	1.8*
INI	1.8	SFE	1.8-6.3*
INPC	7.0 + 7.0n	SFPF	1.8
INPW	7.0 + 7.0n	SHA	1.8-3.8

n = number of words sea

* = Trapped instruction in computers without the appropriate optional hardware package.

** = Dependent upon interrupt response.

*** = Dependent upon a variable signal response time from an external source of equipment.

TABLE 7-3. SUMMARY OF INSTRUCTION EXECUTION TIMES, μsec. (CONTINUED)

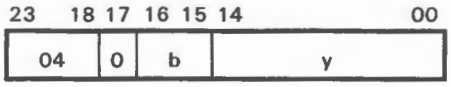
INSTRUCTION MNEMONIC	EXECUTION TIME	INSTRUCTION MNEMONIC	EXECUTION TIME
SHAQ	1.8-3.7	TAI	1.8
SHQ	1.8-3,8	TAM	4.3
SJ1-6	1.8	TIA	1.8
SLS	1.8	TIM	4.3
SQCH	3.5	TMA	4.3
SRCE	7.0 + 7.0n	TMI	4.3
SRCN	7.0 + 7.0n	TMQ	4.3
SSA	3.5	TQM	4.3
SSH	5.2		
SSIM	1.8	UCS	—
STA	3.5	UJP	1.8
STAQ	5.2		
STE	10.8*	XOA	1.8
STI	3.5	XOI	1.8
STQ	3.5	XOQ	1.8
SWA	3.5		

n = number of words
* = Trapped instruction in computers without the appropriate optional hardware package.
** = Dependent upon interrupt response.
*** = Dependent upon a variable signal response time from an external source of equipment.

REGISTER OPERATIONS WITHOUT STORAGE REFERENCE

Operation	Field	Address Field	Interpretation
ASE, S	04	y	Skip next instruction if (A) = y
QSE, S		y	Skip next instruction if (Q) = y
ISE		y, b	Skip next instruction if (B^b) = y
ASG, S	05	y	Skip next instruction if (A) ≥ y
QSG, S		y	Skip next instruction if (Q) ≥ y
ISG		y, b	Skip next instruction if (B^b) ≥ y
ENA, S	14	y	Enter A with y
ECHA, S	11	r	Enter A with 17-bit character address
ENQ, S	14	y	Enter Q with y
ENI		y, b	Enter index with y
INA, S	15	y	Increase A by y
INQ, S		y	Increase Q by y
INI		y, b	Increase index by y
XOA, S	16	y	Exclusive OR of A and y
XOQ, S		y	Exclusive OR of Q and y
XOI		y, b	Exclusive OR of index and y
ANA, S	17	y	AND of A and y
ANQ, S		y	AND of Q and y
ANI		y, b	AND of index and y
ISI	10	y, b	Index skip, incremental
ISD		y, b	Index skip, decremental

JSE Skip Next Instruction if (B^b) = y



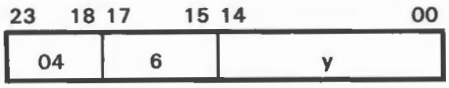
b = index register designator

(Approximate execution time: 2.6 μsec.)

Instruction Description: If (B^b) = y, skip to address P + 2; if not, RNI from address P + 1.

Comments: If b = 0, y is compared to zero.

ASE Skip Next Instruction if (A) = y

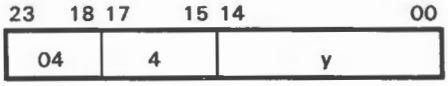


(Approximate execution time: 2.6 μsec.)

Instruction Description: If (A) = y, skip to address P + 2; if not, RNI from address P + 1.

Comments: Only the lower 15 bits of A are used for this instruction.

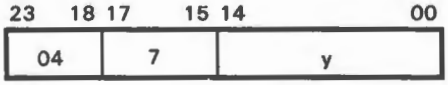
ASE,S Skip Next Instruction if (A) = y



(Approximate execution time: 2.6 μsec.)

Instruction Description: Same as ASE except the sign of y is extended. All 24 bits of A are recognized.

QSE Skip Next Instruction if (Q) = y

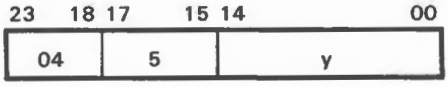


(Approximate execution time: 2.6 μsec.)

Instruction Description: If (Q) = y, skip to address P + 2; if not, RNI from address P + 1.

Comments: Only the lower 15 bits of Q are used for this instruction.

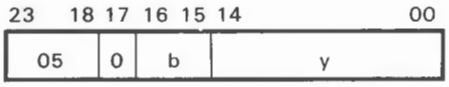
QSE,S Skip Next Instruction if (Q) = y



(Approximate execution time: 2.6 μsec.)

Instruction Description: Same as QSE except the sign of y is extended. All 24 bits of Q are recognized.

ISG Skip Next Instruction if (B^b) ≥ y



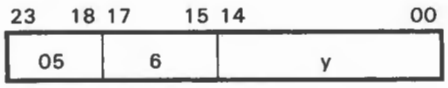
b = index register designator

(Approximate execution time: 2.6 μsec.)

Instruction Description: If (B^b) are equal to or greater than y, skip to address P + 2; if not, RNI from address P + 1.

Comments: If b = 0, y is compared to zero.

ASG Skip Next Instruction if (A) ≥ y

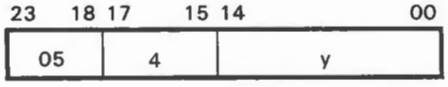


(Approximate execution time: 2.6 μsec.)

Instruction Description: If (A) are equal to or greater than y, skip to address P + 2; if not, RNI from address P + 1.

Comments: (A_{L15}) and y are considered 15-bit positive numbers.

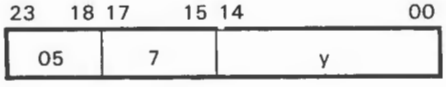
ASG,S Skip Next Instruction if (A) ≥ y



(Approximate execution time: 2.6 μsec.)

Instruction Description: Same as ASG except the sign of y is extended. All 24 bits of A are recognized. Positive zero (00000000) is recognized as greater than negative zero (77777777).

QSG Skip Next Instruction if (Q) ≥ y

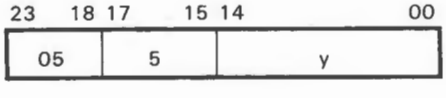


(Approximate execution time: 2.6 μsec.)

Instruction Description: If (Q) are equal to or greater than y, skip to address P + 2; if not, RNI from address P + 1.

Comments: (Q_{L15}) and y are considered 15-bit positive numbers.

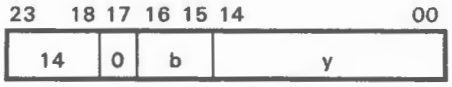
QSG,S Skip Next Instruction if (Q) ≥ y



(Approximate execution time: 2.6 μsec.)

Instruction Description: Same as QSG except the sign of y is extended. All 24 bits of Q are recognized. Positive zero (00000000) is recognized as greater than negative zero (77777777).

ENI Enter Index with y



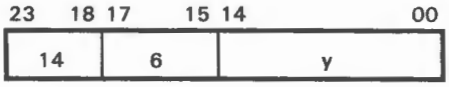
b = index register designator

(Approximate execution time: 1.8 μsec.)

Instruction Description: Clear index register B^b and enter y directly into it.

Comments: If b = 0, this is a no-operation instruction.

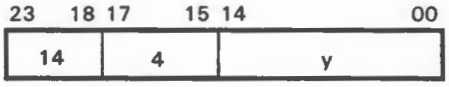
ENA Enter A with y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Clear the A register and enter y directly into A.

ENA,S Enter A with y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Same as ENA except the sign of y is extended.

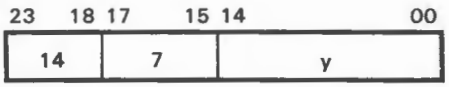
ECHA Enter Character Address into A

d = 0 for no sign extension
d = 1 for sign extension

(Approximate execution time: 1.8 μsec.)

Instruction Description: Clear A; then enter a 17-bit operand z (usually a character address) into A.

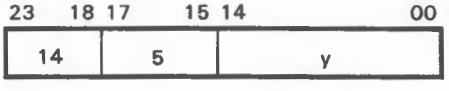
ENQ Enter Q with y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Clear the Q register and enter y directly into Q.

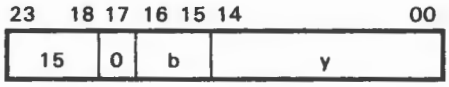
ENQ,S Enter Q with y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Same as ENQ except the sign of y is extended.

INI Increase Index by y



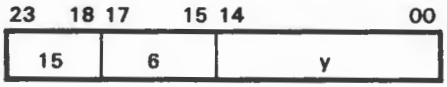
b = index register designator

(Approximate execution time: 1.8 μsec.)

Instruction Description: Add y to (B^b).

Comments: If b = 0, this is a no-operation instruction. Signs of y and B^b are extended.

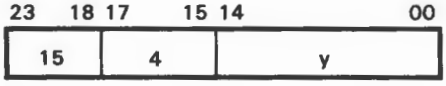
INA Increase A by y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Add y to (A).

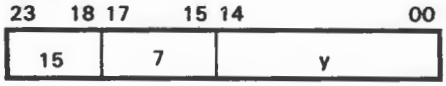
INA,S Increase A by y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Same as INA except the sign of y is extended.

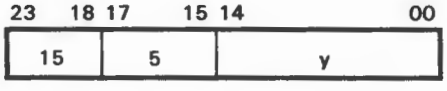
INQ Increase Q by y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Add y to (Q).

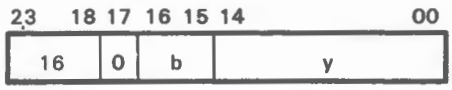
INQ,S Increase Q by y



(Approximate execution time: 1.8 μsec.)

Instruction Description: Same as INQ except the sign of y is extended.

XOI EXCLUSIVE OR of B^b and y



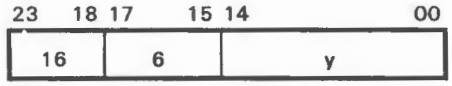
b = index register designator

(Approximate execution time: 1.8 μsec.)

Instruction Description: **Enter the selective complement (the EXCLUSIVE OR function) of y and (B^b) back into the same index register.**

Comments: **If b = 0, this is a no-operation instruction.**

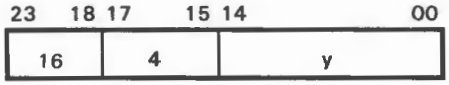
XOA, EXCLUSIVE OR of A and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Enter the selective complement (the EXCLUSIVE OR function) of y and (A) back into the A register.**

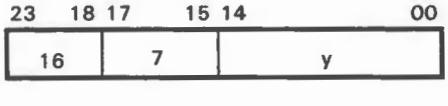
XOA,S EXCLUSIVE OR of A and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Same as XOA except the sign of y is extended.**

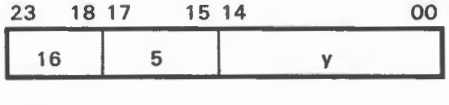
XOQ EXCLUSIVE OR of Q and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Enter the selective complement (the EXCLUSIVE OR function) of y and (Q) back into the Q register.**

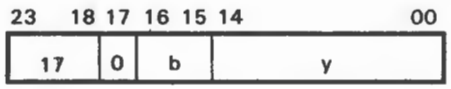
XOQ,S EXCLUSIVE OR of Q and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Same as XOQ except the sign of y is extended.**

[redacted] AND of B^b and y



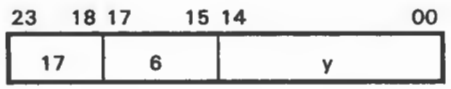
b = index register designator

(Approximate execution time: 1.8 μsec.)

Instruction Description: **Enter the logical product (the AND function) of y and (B^b) back into the same index register.**

Comments: **If b=0, this is a no-operation instruction.**

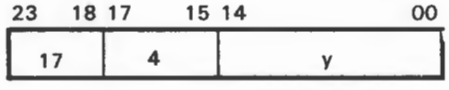
[redacted] AND of A and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Enter the logical product (the AND function) of y and (A) back into the A register.**

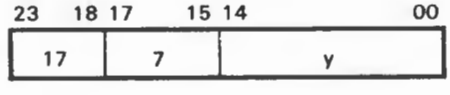
[redacted] AND of A and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Same as ANA except the sign of y is extended.**

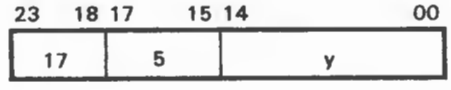
[redacted] AND of Q and y



(Approximate execution time: 1.8 μsec.)

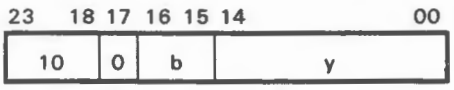
Instruction Description: **Enter the logical product (the AND function) of y and (Q) back into the Q register.**

[redacted] AND of Q and y



(Approximate execution time: 1.8 μsec.)

Instruction Description: **Same as ANQ except the sign of y is extended.**

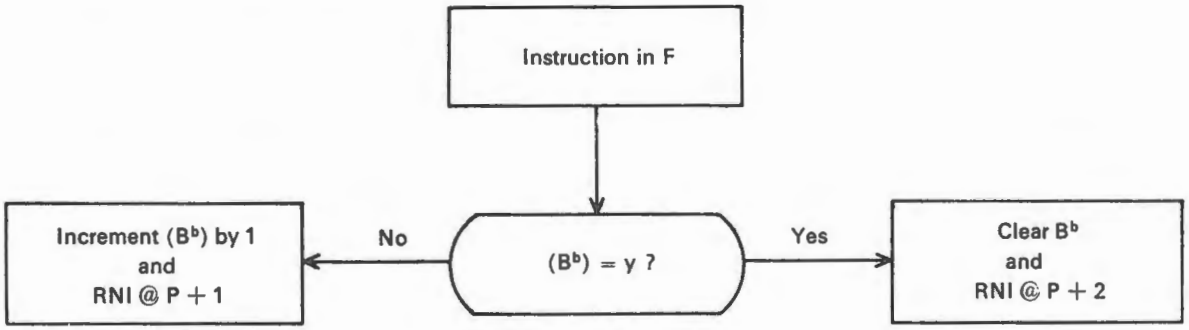


b = index register designator

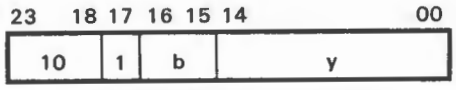
(Approximate execution time: 3.0 μsec.)

Instruction Description: If $(B^b) = y$, clear B^b and skip to address P + 2; if not, add one to (B^b) and RNI from address P + 1.

Comments: The 10.0 instruction is a SSH (storage shift) instruction, described later in this chapter. Positive zero (00000) and negative zero (77777) form an equal comparison.



ISD Index Skip, Decremental

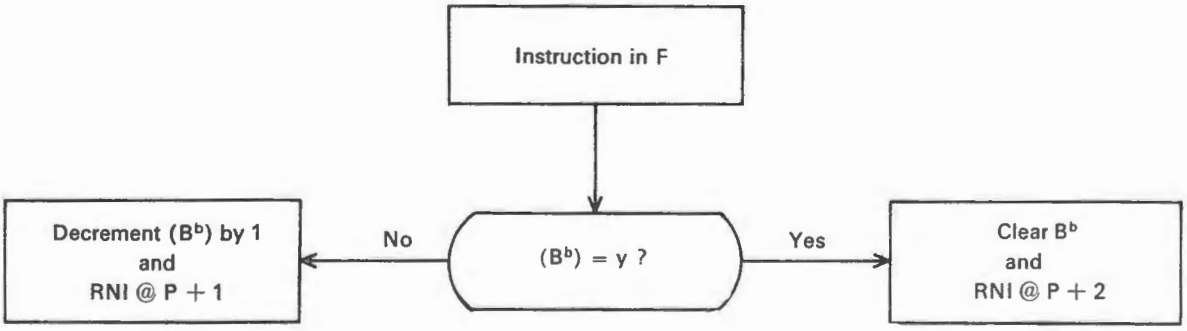


b = index register designator

(Approximate execution time: 3.0 μsec.)

Instruction Description: If $(B^b) = y$, clear B^b and skip to address P + 2; if not, subtract one from (B^b) and RNI from address P + 1.

Comments: Positive zero (00000) and negative zero (77777) form an equal comparison.



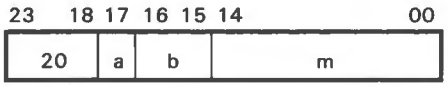
LOAD

Operation Field	Address Field	Interpretation
LDA,I 20	m,b	Load A
LACH 22	r,B[1]	Load A, Character
LCA,I 24	m,b	Load A, Complement
LDL,I 27	m,b	Load A, Logical
LDAQ,I 25	m,b	Load AQ
LCAQ,I 26	m,b	Load AQ, Complement
LDQ,I 21	m,b	Load Q
LQCH 23	r,B[2]	Load Q, Character
LDI,I 54	m,b	Load Index

NOTE

The LDE instruction is described in the BCD section of the instructions.

LDA Load A



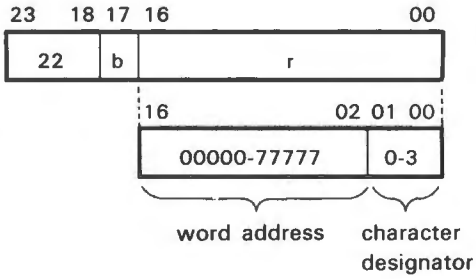
(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Load A with a 24-bit quantity from the storage address specified by M.

Comments: Indirect addressing and address modification may be used.

LACH Load A, Character

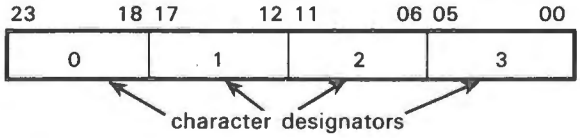


(Approximate execution time: 3.5 μsec.)

If b = 1, r is modified by index register B^1; $R = r + (B^1)$.
If b = 0, r is not modified (r = R).

Instruction Description: Load bits 00 through 05 of A with the character from storage specified by character address R. The A register is cleared prior to the load operation.

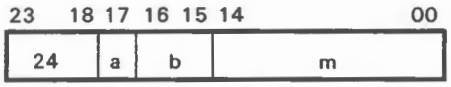
Comments: Indirect addressing may not be used. Characters are specified in storage as follows:



NOTE

Since the sign of B^b is extended during character address modification, it is possible to only reference within ± $16{,}383_{10}$ characters.

LCA Load A, Complement



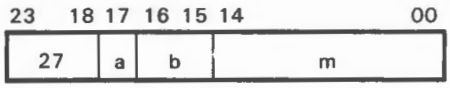
(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Load A with the complement of a 24-bit quantity from storage address M.

Comments: Indirect addressing and address modification may be used.

LDL Load A, Logical

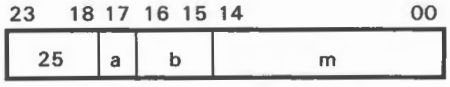


(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator

Instruction Description: Load A with the logical product (the AND function) of (Q) and the 24-bit quantity from storage address M.

LDAQ Load AQ



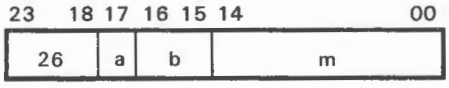
(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Load the A and Q registers with the 24-bit quantities from addresses M and M +1, respectively.

Comments: Addresses 77776 and 77777 should be used only if it is desirable to have M and $\overline{M+1}$ as non-consecutive addresses, since one's complement arithmetic is used to form M + 1.

LCAQ Load AQ, Complement



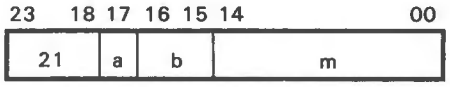
(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Load registers A and Q with the complement of the 24-bit quantities from addresses M and M+1, respectively.

Comments: Addresses 77776 and 77777 should be used only if it is desirable to have M and $\overline{M+1}$ as non-consecutive addresses, since one's complement arithmetic is used to form M + 1.

LDQ Load Q



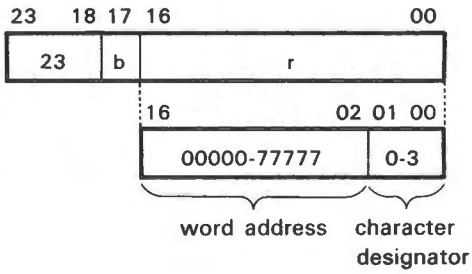
(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load Q with a 24-bit quantity from storage address M.

Comments: Indirect addressing and address modification may be used.

LQCH Load Q, Character



(Approximate execution time: 3.5 μsec.)

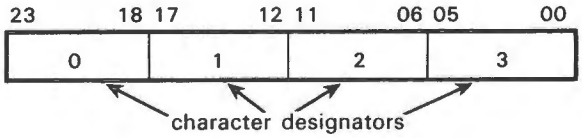
If b = 1, r is modified by index register B^2; R = r + (B^2).
If b = 0, r is not modified (r = R).

NOTE

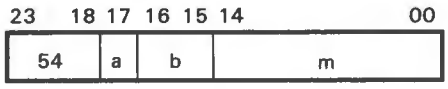
Since the sign of B^b is extended during character address modification, it is possible to only reference within ± $16{,}383_{10}$ characters.

Instruction Description: Load bits 00 through 05 of Q with the character from storage specified by character address R. The Q register is cleared prior to the load operation.

Comments: Indirect addressing may not be used. Characters are specified in storage as follows:



LDI Load Index



(Approximate execution time: 3.8 μsec.)

a = addressing mode designator
b = index register designator
m = storage address (indexing not permitted)

Instruction Description: Load the specified index register, B^b, with the lower 15 bits of the operand stored at address m.

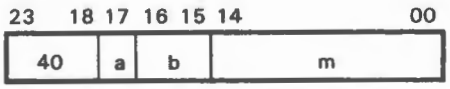
Comments: Indirect addressing may be used but address modification is *not* possible. During indirect addressing only a and m are inspected. Symbol b from the initial instruction specifies which index register is to be loaded with the lower 15-bits from the storage address.

STORE

Operation Field		Address Field	Interpretation
STA,I	40	m,b	Store A
SACH	42	r,B²	Store A, character
STAQ,I	45	m,b	Store AQ
STQ,I	41	m,b	Store Q
SQCH	43	r,B¹	Store Q, character
STI,I	47	m,b	Store index
SWA,I	44	m,b	Store 15-bit word address
SCHA	46	m,b	Store 17-bit character address

NOTE

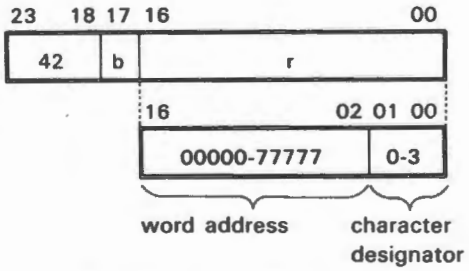
The STE instruction is described in the BCD instruction section.



(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: **Store (A) at the storage address specified by M. The (A) remains unchanged.**

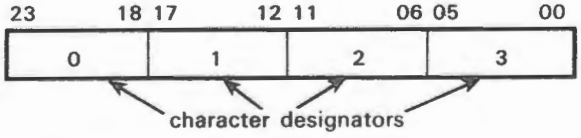


(Approximate execution time: 3.5 μsec.)

If b = 1, r is modified by index register B²; R = r + (B²).
If b = 0, r is not modified (r = R).

Instruction Description: **Store the contents of bits 00 through 05 of the A register in the specified character address. All of (A) and the remaining three characters in storage remain unchanged.**

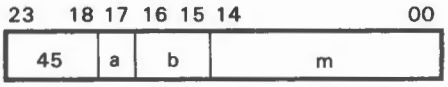
Comments: **Indirect addressing may not be used. Characters are specified in storage as follows:**



NOTE

Since the sign of B^b is extended during character address modification, it is possible to only reference within ± 16,383₁₀ characters.

STAQ Store AQ



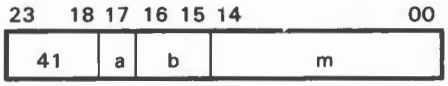
(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Store (A) and (Q) in the storage locations specified by address M and M + 1, respectively. The (A) and (Q) remains unchanged.

Comments: Addresses 77776 and 77777 should be used only if it is desirable to have M and M + 1 as non-consecutive addresses, since one's completment arithmetic is used to form M + 1.

STQ Store Q

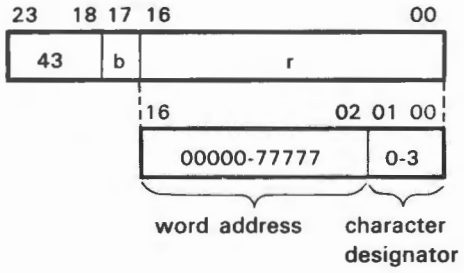


(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Store (Q) at the storage address specified by M. The (Q) remains unchanged.

SQCH Store Q Character

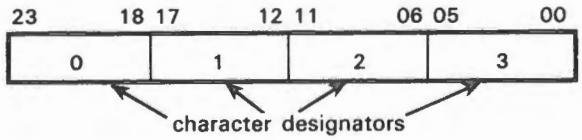


(Approximate execution time: 3.5 μsec.)

If b = 1, r is modified by index register B^1; R = r + (B^1).
If b = 0, r is not modified. (r = R)

Instruction Description: Store the contents of bits 0 through 5 of the Q register in the specified character address. All of (Q) and the remaining three characters in storage remain unchanged.

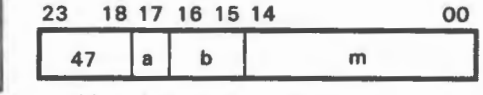
Comments: Indirect addressing may *not* be used. Characters are specified in storage as follows:



NOTE

Since the sign of B^b is extended during character address modification, it is possible to reference only within ± 16,383$_{10}$ characters.

STI Store Index



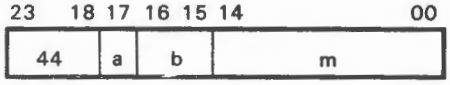
(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address (indexing not permitted)

Instruction Description: **Store the contents of the specified index register, B^b, in the lower 15 bits of storage address m. The upper 9 bits of m and all of (B^b) remain unchanged.**

Comments: **Indirect addressing may be used, but address modification is not possible. During indirect addressing only a and m are inspected. The b designator from the initial instruction specifies the index register that will have its contents stored. If b=0, zeros are stored in the lower 15 bits of m.**

SWA Store Word Address

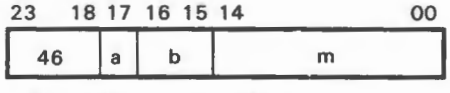


(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: **Store the lower 15 bits of (A) in the designated address M. The upper 9 bits of M and all of (A) remain unchanged.**

SCHA Store Character Address



(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: **Store the lower 17 bits of (A) in the address designated by M. The upper 7 bits of M and all of (A) remain unchanged.**

INTER-REGISTER TRANSFER, 24-BIT PRECISION

Operational Field	Address Field	Interpretation
AQA 53		Transfer (A) + (Q) to A
AIA	b	Transfer (A) + (B^b) to A
IAI	b	Transfer (B^b) + (A) to B^b
TIA	b	Transfer (B^b) to A
TAI	b	Transfer (A) to B^b
TMQ	v	Transfer (Register v) to Q
TQM	v	Transfer (Q) to Register v
TMA	v	Transfer (Register v) to A
TAM	v	Transfer (A) to Register v
TMI	v,b	Transfer (Register v) to B^b
TIM	v,b	Transfer (B^b) to Register v

General Instruction Description

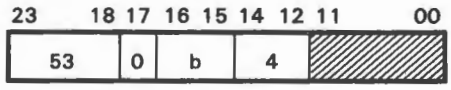
The 53 instruction is used to move data between the A and Q registers, the index registers, and the Register File. The contents of the transferring register remain unchanged.

AQA Transfer (A) + (Q) to A

(Approximate execution time: 1.8 μsec.)

Comments: (Q) remains unchanged. Bits 00 through 11 should be loaded with zeros. Arithmetic Overflow is not sensed.

AIA Transfer (A) + (B^b) to A

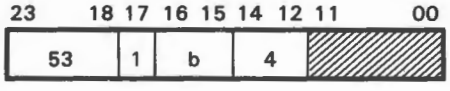


b = index register designator

(Approximate execution time: 1.8 μsec.)

Comments: The sign of (B^b) is extended prior to the addition. Bits 00 through 11 should be loaded with zeros. Arithmetic Overflow is not sensed.

IAI Transfer (A) + (B^b) to B^b

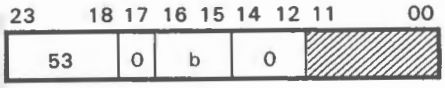


b = index register designator

(Approximate execution time: 1.8 μsec.)

Comments: The sign of the original (B^b) is extended prior to the addition. The upper 9 bits of the sum are lost when the sum is transferred to the index register. Bits 00 through 11 should be loaded with zeros.

TIA Transfer (B^b) to A

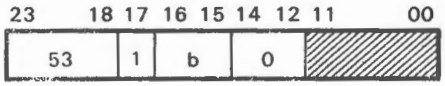


b = index register designator

(Approximate execution time: 1.8 μsec.)

Comments: No sign extension on B^b. Prior to the transfer, (A) is cleared. If b = 0, zeros are transferred to A. Bits 00 through 11 are loaded with zeros.

TAI Transfer (A) to B^b

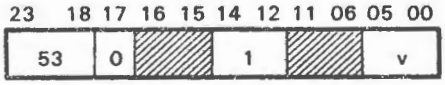


b = index register designator

(Approximate execution time: 1.8 μsec.)

Comments: The (A) remains unchanged. If b = 0, this becomes a no-operation instruction. Bits 00 through 11 should be loaded with zeros.

TMQ Transfer (Register v) to Q

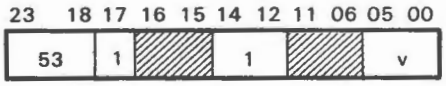


v = Register File number, $00\text{-}77_8$

(Approximate execution time: 4.3 μsec.)

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros.

TQM Transfer (Q) to Register v

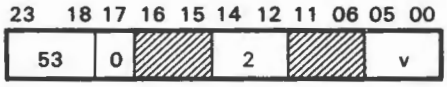


v = register file number, $00\text{-}77_8$

(Approximate execution time: 4.3 μsec.)

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros.

TMA Transfer (Register v) to A

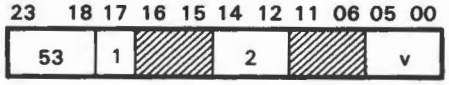


v = register file number, 00-77$_8$

(Approximate execution time: **4.3** μsec.)

Comments: **Bits 06 through 11, 15 and 16 should be loaded with zeros.**

TAM Transfer (A) to Register v

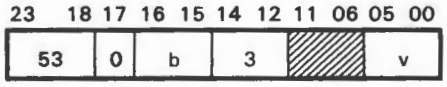


v = register file number, 00-77$_8$

(Approximate execution time: 4.3 μsec.)

Comments: **Bits 06 through 11, 15 and 16 should be loaded with zeros.**

TMI Transfer (Register v) to B^b

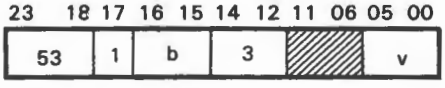


b = index register designator
v = register file number, 00-77$_8$

(Approximate execution time: 4.3 μsec.)

Comments: **Lower 15 bits of v are transferred to B^b. Bits 06 through 11 should be loaded with zeros.**

TIM Transfer (B^b) to Register v



b = index register designator
v = register file number, 00-77$_8$

(Approximate execution time: **4.3** μsec.)

Comments: **Upper nine bits of v are cleared. Bits 06 through 11 should be loaded with zeros.**

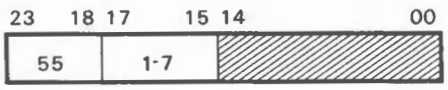
INTER-REGISTER TRANSFER, 48-BIT PRECISION

Operation Field	Address Field	Interpretation
ELQ* 55	- - - -	Transfer (E_L) to Q
QEL*	- - - -	Transfer (Q) to E_L
EUA*	- - - -	Transfer (E_U) to A
AEU*	- - - -	Transfer (A) to E_U
EAQ*	- - - -	Transfer (E) to AQ
AQE*	- - - -	Transfer (AQ) to E

*Trapped instruction if the Floating Point/Double Precision (FP/DP) arithmetic option is not present.

TRAPPED INSTRUCTIONS IF FP/DP ARITHMETIC OPTION IS NOT PRESENT

ELQ	Transfer (E_L) to Q	55.1
EUA	Transfer (E_U) to A	55.2
EAQ	Transfer (E) to AQ	55.3
QEL	Transfer (Q) to E_L	55.5
AEU	Transfer (A) to E_U	55.6
AQE	Transfer (AQ) to E	55.7



(Approximate execution time: 1.8-6.1 μsec.) option present.

Instruction Description: The 48-bit E register is split into halves-E_U and E_L. With the 55 instruction, data may be moved as a 48-bit word between E and AQ, or in halves between A and E_U or Q and E_L.

Comments: Bits 00 through 14 should be loaded with zeros. 55.0 and 55.4 are no-operation instructions, even with the option present.

STOP AND JUMPS

Operation	Field	Address Field	Interpretation
HLT	00	m	Unconditional stop; RNI from address m
SLS	77.70		Selective stop
UCS	77.77		Unconditional stop
SJ1		m	Jump if key 1 is set
SJ2		m	Jump if key 2 is set
SJ3		m	Jump if key 3 is set
SJ4		m	Jump if key 4 is set
SJ5		m	Jump if key 5 is set
SJ6		m	Jump if key 6 is set
RTJ		m	Return jump
UJP,I	01	m,b	Unconditional jump
IJI	02	m,b	Index jump; increment index
IJD		m,b	Index jump; decrement index
AZJ,EQ	03	m	Compare A with zero; jump if (A) = 0
NE			Compare A with zero; jump if (A) ≠ 0
GE			Compare A with zero; jump if (A) ≥ 0
LT			Compare A with zero; jump if (A) < 0
AQJ,EQ		m	Compare A with Q; jump if (A) = (Q)
NE			Compare A with Q; jump if (A) ≠ (Q)
GE			Compare A with Q; jump if (A) ≥ (Q)
LT			Compare A with Q; jump if (A) < (Q)

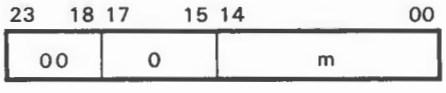
NOTE

Two additional Jump instructions, EZJ and EOJ, are described under the BCD instructions.

A Jump instruction causes a current program sequence to terminate and initiates a new sequence at a different storage location. The P register provides continuity between program steps and always contains the storage location of the current program step. When a Jump instruction occurs, a new address is entered into P. In most Jump instructions, the execution address m specifies the beginning address of the new program sequence. The word at address m is read from storage, placed in F, and the first instruction of the new sequence is executed.

Some of the Jump instructions are conditional upon a register containing a specific quantity or upon the status of the Jump key on the console. If the condition is satisfied, the jump is made to location m. If not, the program proceeds in its normal sequence to the next instruction.

HLT Halt

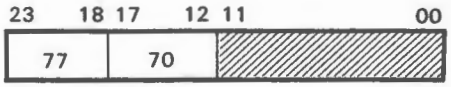


(Approximate execution time: 1.8 *usec*)

Instruction Description: **Unconditionally halt at this instruction. Upon restarting, RNI from address m.**

Comments: **Indirect addressing and address modification may not be used.**

SLS Selective Stop

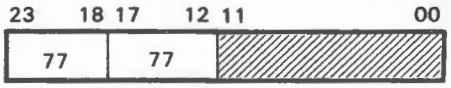


(Approximate execution time: 1.8 μsec.)

Instruction Description: **Program execution halts if the Select Stop switch on the console is set. RNI from address P + 1 when restarting.**

Comments: **Bits 00 through 11 should be loaded with zeros.**

UCS Unconditional Stop

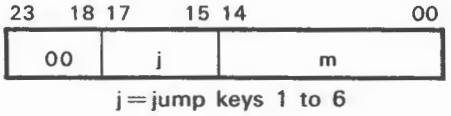


(Approximate execution time: indeterminate)

Instruction Description: **This instruction unconditionally stops the execution of the current program. RNI from address P + 1 when restarting.**

Comments: **Bits 00 through 11 should be loaded with zeros.**

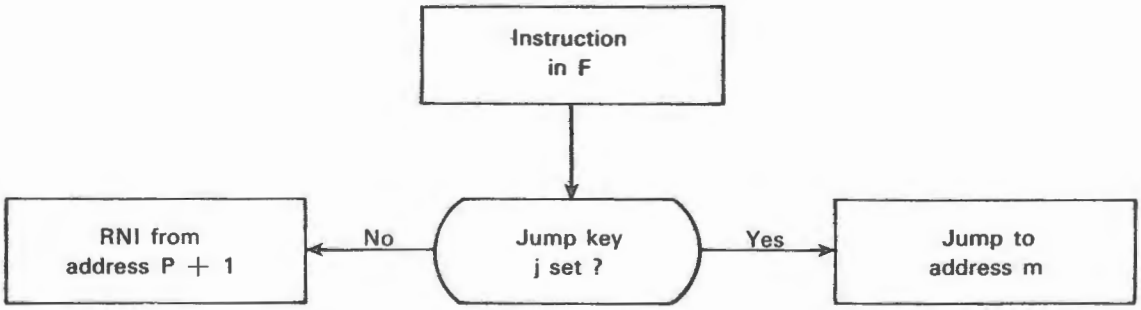
SJ6 Selective Jump



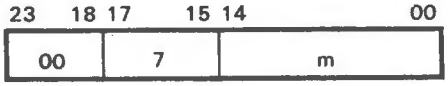
(Approximate execution time: 1.8 μsec.)

Instruction Description: **Jump to address m if Jump key j is set; otherwise, RNI from address P + 1.**

Comments: **Indirect addressing and address modification may *not* be used.**



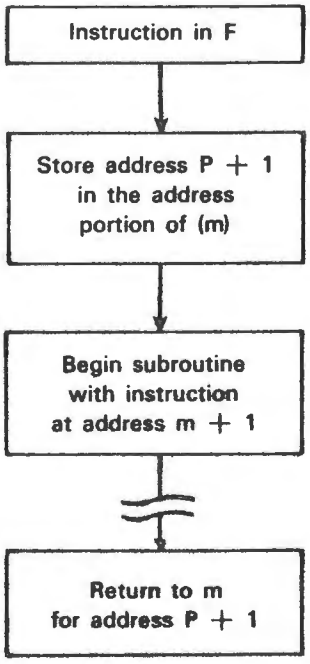
[redacted]



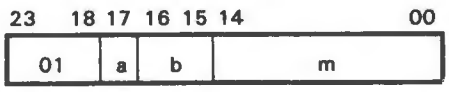
(Approximate execution time: 3.6 μsec.)

Instruction Description: **The address portion of m is replaced with the return address, P + 1. Jump to location m + 1 and begin executing instructions at that location.**

Comments: **Indirect addressing and address modification may *not* be used.**



[redacted]



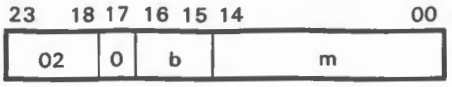
(Approximate execution time: 1.8 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Unconditionally jump to address M.**

Comments: **Indirect addressing and indexing may be used**

IJI Index Jump, Incremental



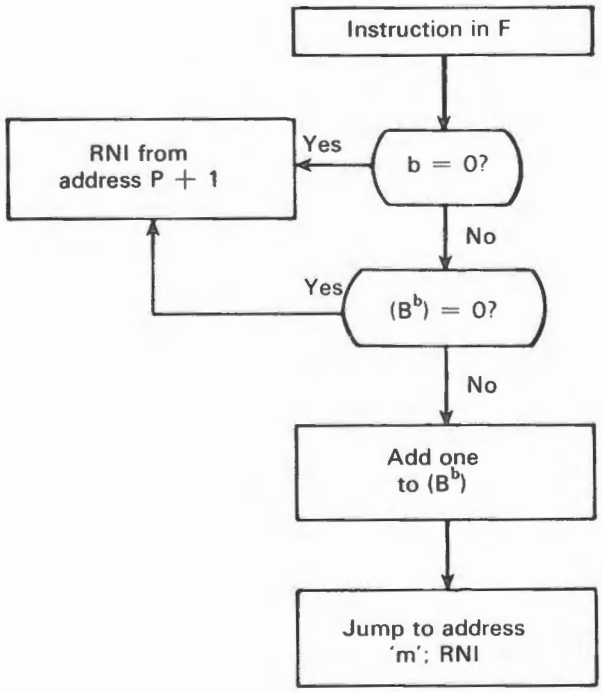
b = index register designator
m = jump address

(Approximate execution time: 2.6 μsec.)

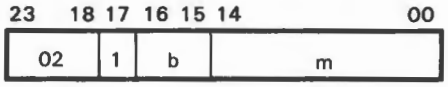
Instruction Description: If b = 1, 2, or 3, the respective index register is examined:

1. If $(B^b) = 00000$, the jump test condition is not satisfied; RNI from address P + 1.
2. If $(B^b) \neq 00000$, the jump test condition is satisfied. One is added to (B^b); jump to address m and RNI.

Comments: If b = 0, this is a no-operation instruction; RNI from address P + 1. Indirect addressing and jump address modification may *not* be used. The counting operation is done in a one's complement additive accumulator. Negative zero (77777) is not generated because the count progresses from: 77775, 77776, to 00000 (positive zero) and stops. If negative zero is initially loaded into B^b, the count progresses: 77777, 00001, 00002, etc. In this case, the counter must increment through the entire range of numbers to reach positive zero.



IJD Index Jump, Decremental



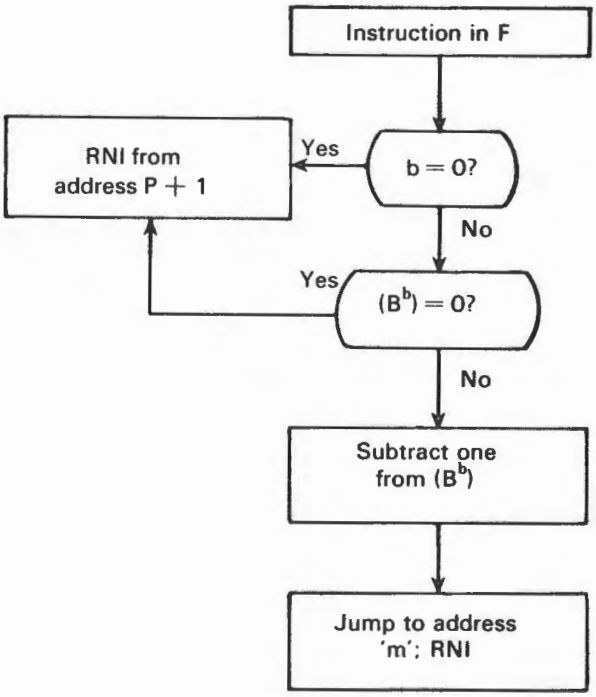
(Approximate execution time: 2.6 μsec.)

b = index register designator
m = jump address

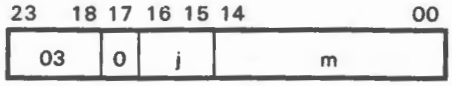
Instruction Description: If b=1, 2 or 3, the respective index register is examined:

1. If $(B^b)=00000$, the jump test condition is not satisfied; RNI from address P + 1.
2. If $(B^b)\neq 00000$, the jump test condition is satisfied. One is subtracted from (B^b); jump to address m and RNI.

Comments: If b=0, this is a no-operation instruction; RNI from address P + 1. Indirect addressing and jump address modification may *not* be used. If negative zero (77777) is initially loaded into B^b, the count will decrement through the entire range of numbers to reach 00000 before the program will RNI from P + 1.



AZJ, Condition Compare A with Zero, Jump



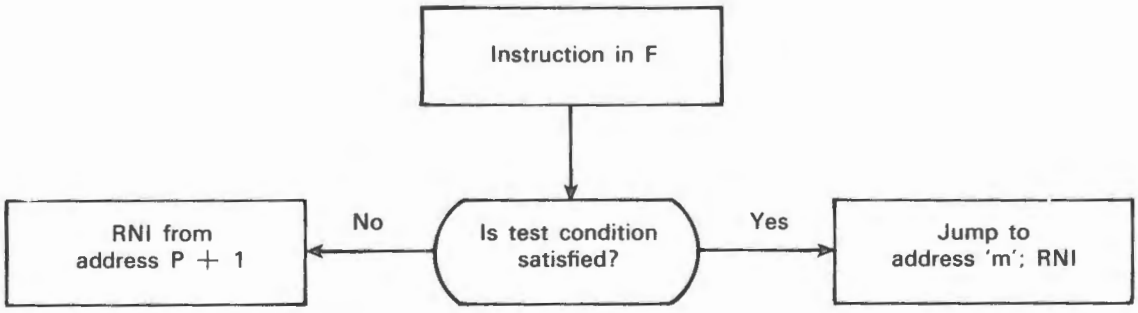
(Approximate execution time: 2.6 μsec.)

j = jump designator (0-3)
m = jump address

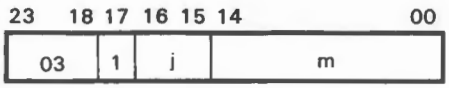
Instruction Description: **The operand in A is algebraically compared with zero for an equality, inequality, greater-than or less-than condition (see table). If the test condition is satisfied, program execution jumps to address m. If the test condition is not satisfied, RNI from address P + 1.**

Comments: **Positive zero (00000000) and negative zero (77777777) give identical results when j = 0 or 1. When j = 2 or 3, negative zero is recognized as less than positive zero. Indirect addressing and address modification may *not* be used.**

Condition Mnemonic	Jump Designator j	Test Condition
EQ	0	(A) = 0
NE	1	(A) ≠ 0
GE	2	(A) ≥ 0
LT	3	(A) < 0



AQJ, Condition Compare A With Q, Jump



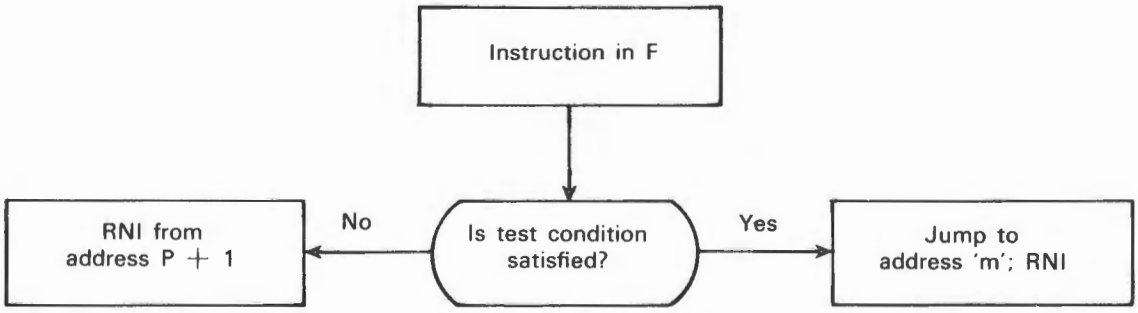
(Approximate execution time: 2.6 μsec.)

j = 0-3 jump designator (0-3)
m = jump address

Instruction Description: The quantity in A is algebraically compared with the quantity in Q for equality, inequality, greater-than or less-than condition (see table). If the test condition is satisfied, program execution jumps to address m. If the test condition is not satisfied, RNI from address P + 1.

Comments: This instruction may be used to test (Q) by placing an arbitrary value in A for the comparison. Positive and negative zero give identical results in this test when j = 0 or 1. When j = 2 or 3, negative zero is recognized as less than positive zero. Indirect addressing and address modification may *not* be used.

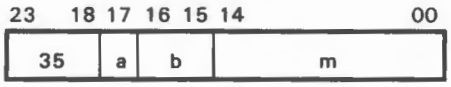
Condition Mnemonic	Jump Designator j	Test Condition
EQ	0	(A) = (Q)
NE	1	(A) ≠ (Q)
GE	2	(A) ≥ (Q)
LT	3	(A) < (Q)



LOGICAL INSTRUCTIONS WITH STORAGE REFERENCE

Operation Field		Address Field	Interpretation
SSA,I	35	m,b	Selectively set A
SCA,I	36	m,b	Selectively complement A
LPA,I	37	m,b	Logical product A

SSA Selectively Set A

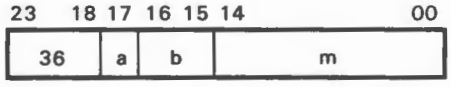


(Approximate execution time: **3.5** μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Selectively set the bits in the A register to "1's" for all corresponding "1's" in the quantity at address M.**

SCA Selectively Complement A

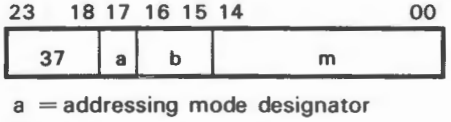


(Approximate execution time: **3.5** μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Selectively complement the bits in the A register that correspond to the set bits in the quantity at address M.**

LPA Logical Product A



(Approximate execution time: **3.5** μsec.)

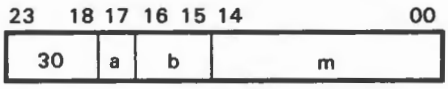
a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Replace (A) with the logical product of (A) and (M).**

ARITHMETIC, FIXED POINT, 24-BIT PRECISION

Operation Field	Address Field	Interpretation
ADA,I 30	m,b	Add to A
RAD,I 34	m,b	Replace add
SBA,I 31	m,b	Subtract from A
MUA,I 50	m,b	Multiply A
DVA,I 51	m,b	Divide A

ADA Add to A

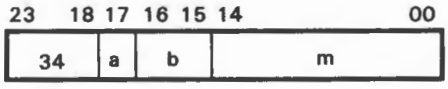


(Approximate execution time: 3.5 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Add the 24-bit operand located at address M to (A). The sum replaces the original (A).**

RAD Replace Add

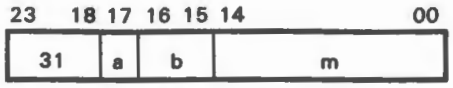


(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: **Replace the quantity at address M with the sum of (M) and (A). The original (A) remains unchanged.**

[redacted]

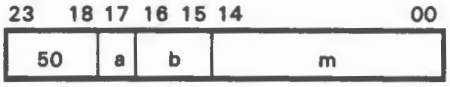


(Approximate execution time [redacted] μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 24-bit operand located at address M from (A). The difference replaces the original (A).

MUA Multiply A

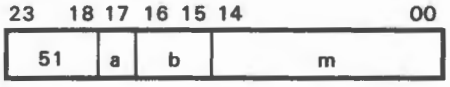


(Approximate execution time: 11.5-15.0 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply (A) by the operand located at address M. The 48-bit product is displayed in QA with the lowest order bits in A.

[redacted]



(Approximate execution time: 15.0 μsec.)

a = address mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Divide the 48-bit operand in AQ by the operand at storage address M. The quotient is displayed in A and the remainder with sign extended is displayed in Q. If a divide fault occurs, the operation halts and program execution advances to the next address. The final (A) and (Q) are meaningless if a divide fault occurs.

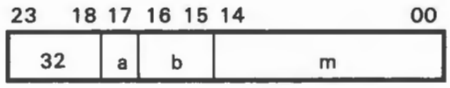
ARITHMETIC, FIXED POINT, 48-BIT PRECISION

Operation Field		Address Field	Interpretation
ADAQ,I	32	m,b	Add to AQ
SBAQ,I	33	m,b	Subtract from AQ
*MUAQ,I	56	m,b	Multiply AQ
*DVAQ,I	57	m,b	Divide AQ

*Trapped instruction if arithmetic option is not present.

This group of instructions may use indirect addressing and address modification. The A and Q registers function as a single 48-bit register with the highest order bits in A. Address 77777 is not recommended for use with this group of instructions.

ADAQ Add to AQ



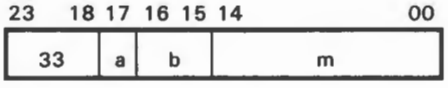
(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: **Add the 48-bit operand located in addresses M and M + 1 to (AQ). The sum is displayed in AQ.**

Comments: **The upper 24 bits of the 48-bit operand in memory are contained at address M.**

SBAQ Subtract from AQ



(Approximate execution time: 5.2 μsec.)

a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: **Subtract the 48-bit operand located in addresses M and M + 1 from (AQ). The difference is displayed in AQ.**

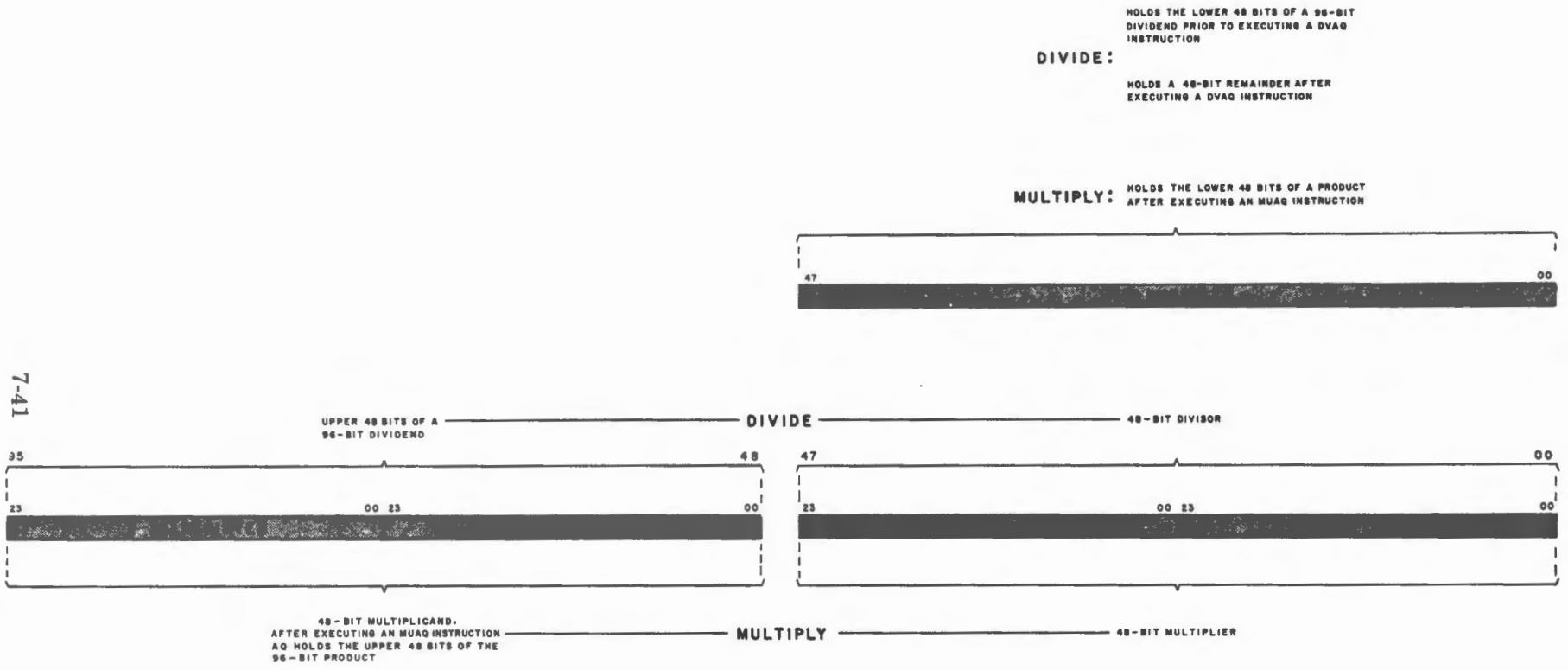
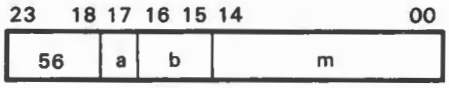


Figure 7-4. Operand Formats and Bit Allocations for MUAQ and DVAQ Instructions

MUAQ Multiply AQ



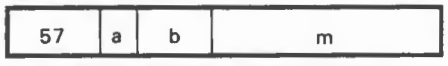
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

(Approximate execution time: 22.4-29.4 μsec.)

Instruction Description: **Multiply (AQ) by the 48-bit operand in addresses M and M + 1. The 96-bit product is displayed in AQE.**

Comments: **Refer to Figure 7-4 for operand formats.**

DVAQ Divide AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

(Approximate execution time: 31.5 μsec.) option present.

Instruction Description: **Divide (AQE) by the 48-bit operand in addresses M and M + 1. The quotient is displayed in AQ, and the remainder with its sign extended is displayed in E.**

Comments: **If a divide fault occurs, program execution advances to the next address. The final contents of AQ and E are meaningless if a divide fault occurs. Refer to Figure 7-4 for operand formats.**

ARITHMETIC, FLOATING POINT

Operational Field	Address Field	Interpretation
*FAD,I 60	m,b	FP addition to AQ
*FSB,I 61	m,b	FP subtraction from AQ
*FMU,I 62	m,b	FP multiplication of AQ
*FDV,I 63	m,b	FP division of AQ

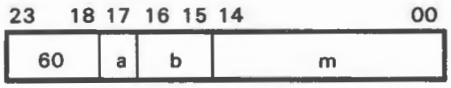
*Trapped instruction if the Floating Point/Double Precision (FP/DP) arithmetic option is not present.

GENERAL FLOATING POINT/DOUBLE PRECISION NOTE

Figure 7-5 illustrates operand format and bit allocations for floating point instructions. Refer to the Floating Point section of Appendix B for additional floating point considerations and examples.

TRAPPED INSTRUCTION IF FP/DP ARITHMETIC OPTION IS NOT PRESENT

FAD FP Addition to AQ



(Approximate execution time: 14-16.8 μsec.) option present.

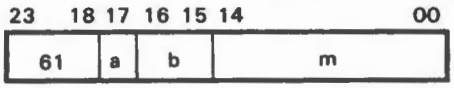
a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Add the 48-bit operand located in addresses M and M + 1 to (AQ). The rounded and normalized sum is displayed in AQ.

Comments: The higher order bits of E hold the portion of the operand that was shifted out of AQ during exponent equalization.
Refer to Figure 7-5 for operand formats.

TRAPPED INSTRUCTIONS IF FP/DP ARITHMETIC OPTION IS NOT PRESENT

FSB FP Subtraction from AQ



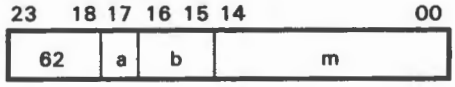
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

(Approximate execution time: 14-16.8 μsec.) option present.

Instruction Description: Subtract the 48-bit floating point operand located at storage addresses M and M + 1 from the floating point operand in AQ. The rounded and normalized difference is displayed in AQ.

Comments: The upper order bits of E hold the portion of the operand that was shifted out of AQ during the equalization of exponents. Refer to Figure 7-5 for operand formats.

FMU FP Multiplication of AQ



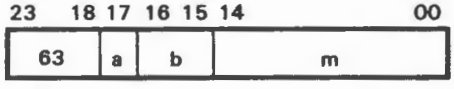
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

(Approximate execution time: 19.6-25.2 μsec.) option present.

Instruction Description: Multiply the 48-bit floating point operand in AQ by the floating point operand located at storage addresses M and M + 1. The rounded and normalized product is displayed in AQ.

Comments: Bits of 12-47 of E hold the lower 36 bits of the 72-bit unnormalized product. Refer to Figure 7-5 for operand formats.

FDV FP Division of AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

(Approximate execution time: 28.0 μsec.) option present.

Instruction Description: Divide the floating point operand in AQ by the 48-bit floating point operand located at storage addresses M and M + 1. The rounded and normalized quotient is displayed in AQ. The remainder with sign extended appears in the E register.

Comments: The sign of the remainder is the same as that of the dividend. Refer to Figure 7-5 for operand formats.

NOTE

The divisor must be properly normalized or a divide fault will result. Refer to Interrupt Conditions in Section 4.

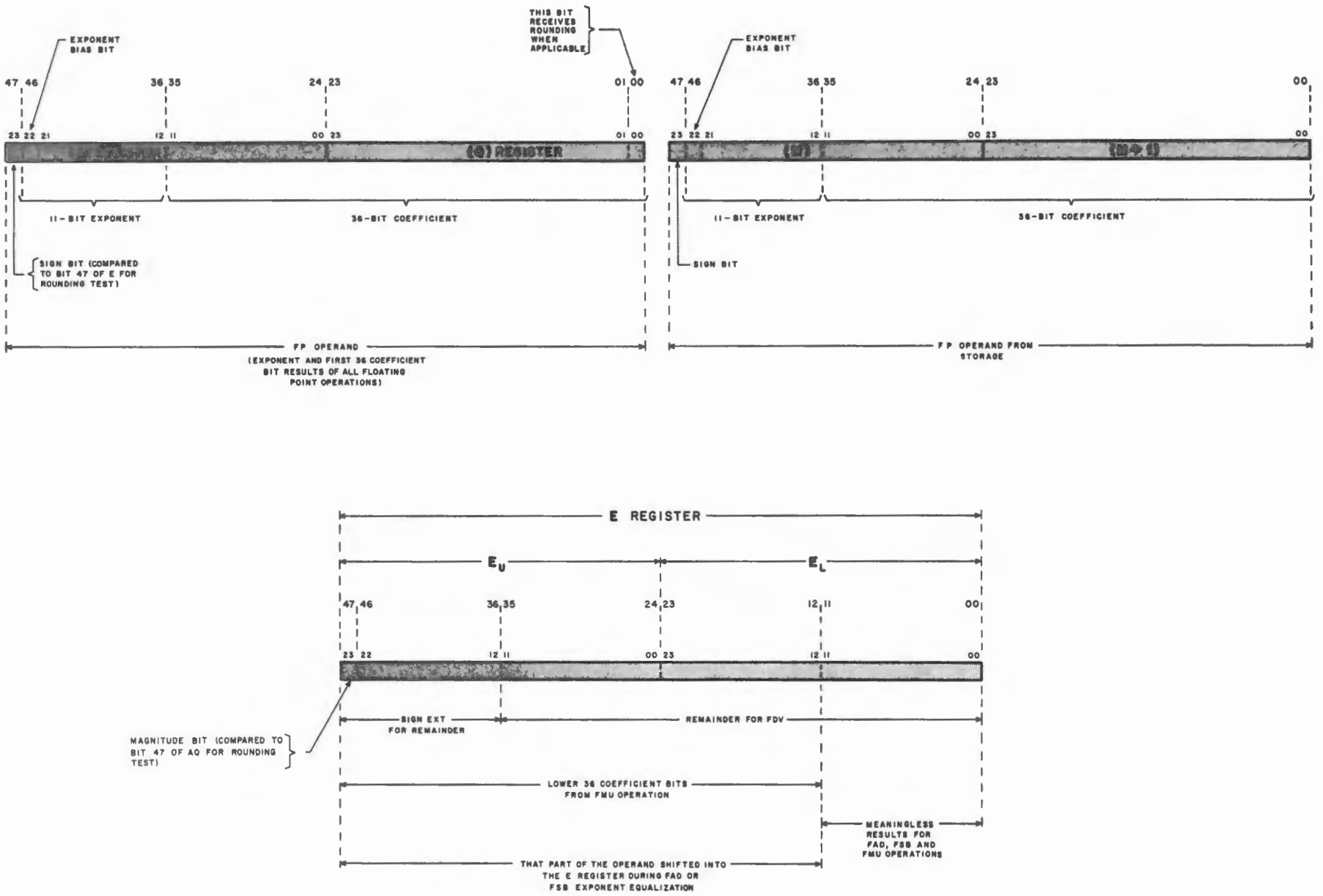


Figure 7-5. Operand Formats and Bit Allocations for Floating Point Arithmetic Instructions.

BCD

Operational Field		Address Field	Interpretation
*SET	70	y	Set D register
*ADE	66	m,b[3]	Add to E
*SBE	67	m,b[3]	Subtract from E
*LDE	64	m,b[1]	Load E
*STE	65	m,b[2]	Store E
*SFE	70	k,b	Shift E
*EZJ,EQ		m	E zero jump, E = 0
*EZJ,LT		m	E zero jump, E < 0
*EOJ		m	E overflow jump

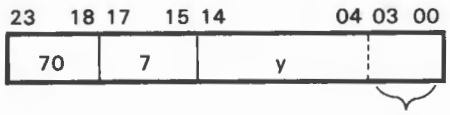
*Trapped instruction if BCD arithmetic option is not present.

GENERAL BCD INSTRUCTION NOTE

Refer to the BCD arithmetic section of Appendix B for additional BCD considerations and examples.

TRAPPED INSTRUCTIONS IF BCD ARITHMETIC OPTION IS NOT PRESENT

SET Set D Register



(Approximate execution time: 1.8 μsec.) option present.

only the lower 4 bits are recognized.

y = field length designator

Instruction Description: **Load the lower 4 bits of y into the 4-bit D register.**

Comments: **(D) remains the same until replaced by a new 4-bit operand. In LDE and STE operations dealing with equal size fields, the D register is loaded only once with a SET instruction. If y = 0, subsequent LDE, STE, ADE and SBE instructions are processed as no-ops. Refer to the BCD section of Appendix B for an example of a SET instruction execution.**

ADE Add to E_D

If B = 1, r is modified by (B^3); R = r + (B^3).
If b = 0, r is the unmodified direct address (r = R).

(Approximate execution time: 16.1 μsec.) option present.

Instruction Description: A maximum field of 12 BCD numeric characters in storage may be added to (E_D). The sum is displayed in E_D.

Comments: The characters in storage are in consecutive character positions. R specifies the most significant character (MSC) of a field. The 4-bit D register specifies field length. The (E_D) are always right justified, i.e. the lowest significant digit of the operand is always in the digit zero position.

NOTE

Since the sign of B^b is extended during character address modification, it is possible to reference only within ± $16{,}383_{10}$ characters.

SBE Subtract from (E_D)

If b = 1, r is modified by (B^3); R = r + (B^3).
If b = 0, r is the unmodified direct address (r = R).

(Approximate execution time: 16.1 μsec.) option present.

Instruction Description: A maximum field of 12 BCD characters in storage is subtracted from (E_D). The difference is displayed in E_D.

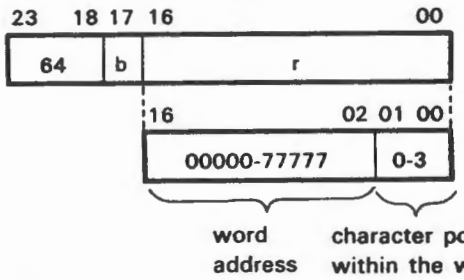
Comments: The characters in storage that comprise the subtrahend are located in consecutive character positions. R specifies the most significant character of a field. The 4-bit D register specifies the field length. The (E_D) are always right justified, i.e. the lowest significant digit of the operand is always in the digit zero position.

NOTE

Since the sign of B^b is extended during character address modification, it is possible to reference only within ± $16{,}383_{10}$ characters.

TRAPPED INSTRUCTIONS IF BCD ARITHMETIC OPTION IS NOT PRESENT

LDE Load E_D



(Approximate execution time: 10.8 μsec.) option present.

If b = 1, r is modified by (B^1); R = r + (B^1).
If b = 0, r is the unmodified direct address.

Instruction Description: Load the E_D register with a maximum field of 12 numeric BCD characters from storage.

Comments: Characters are loaded consecutively, starting with the least significant character (LSC) at address R + (D-1) and continuing until the most significant character at address R is in E_D. (E_D) is shifted right as loading progresses. The sign of the decimal operand is acquired along with the LSC. Prior to executing this instruction, the field length must be specified with a SET (70.7) instruction. The (E_D) and always right justified, i.e., the lowest significant digit of the operand is always in the digit zero position.
Refer to the BCD section of Appendix B for an LDE instruction execution example.

NOTE

Since the sign of B^b is extended during character address modification, it is possible to reference only within ± $16{,}383_{10}$ characters.

STE Store E_D

(Approximate execution time: 10.8 μsec.) option present.

If b = 1, r is modified by the (B^2); R = r + (B^2)
If b = 0, r is the unmodified direct address (r = R).

Instruction Description: Store a maximum field of 13 numeric BCD characters from the E_D register into storage.

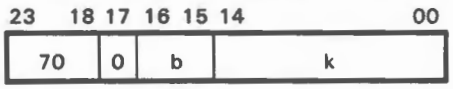
Comments: Characters are stored, beginning with the least significant character (LSC) and the sign of the stored operand is acquired with this character. (E_D) is shifted right as the Store operation progresses, end off, until the entire field of characters is stored. Prior to executing this instruction the field length must be specified with a SET (70.7) instruction.

NOTE

Since the sign of B^b is extended during character address modification, it is possible to reference only within ± $16{,}383_{10}$ characters.

TRAPPED INSTRUCTIONS IF BCD ARITHMETIC OPTION IS NOT PRESENT

SFE Shift E_D



b = index register designator
k = shift designator

(Approximate execution time: 1.8-6.3 μsec.) option present.

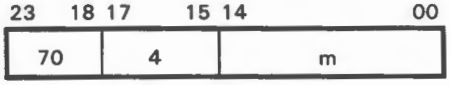
Instruction Description: This instruction shifts BCD characters within the E_D register in single character (4-bit) shifts.

Comments: k is added to (B^b) to modify the shift designator; K = k + (B^b). The sign of B^b is extended. The computer senses bits 00-03 and 23 of the sum. A left shift is performed if bit 23 is zero, and a right shift if it is one. Shifts are end-off in both directions. For a left shift, bits 00-03 specify the shift magnitude and for a right shift, the complement of the lower 4 bits of the sum specify the shift magnitude.

Examples:

If K = 00000006, shift left 6 character positions.
If K = 77777771, shift right 6 character positions.

ZJ,EQ E_D
Zero Jump, (E_D) = 0

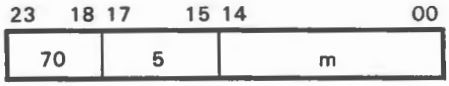


m = jump address

(Approximate execution time: 1.8 μsec.) option present.

Instruction Description: This instruction compares (E_D) with zero. If (E_D) = 0, jump to address m; if not, RNI from address P + 1.

ZJ,LT E_D
Zero Jump, (E_D) < 0

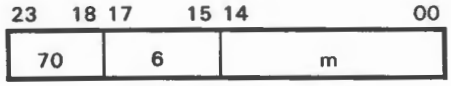


m = jump address

(Approximate execution time: 1.8 μsec.) option present.

Instruction Description: This instruction compares (E_D) with zero. If (E_D) < 0, jump to address m; if not, RNI from address P + 1.

EOJ E_D Overflow Jump



m = jump address

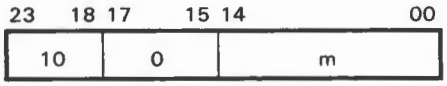
(Approximate execution time: 1.8 μsec.) option present.

Instruction Description: Jump to address m if the overflow digit (digit 13) of the E_D register receives a character indicating that E_D had overflowed. The overflow condition is also true where an ADE or SBE causes an end-off carry in the overflow digit. If overflow has not occurred, RNI from address P + 1.

STORAGE SHIFT, SEARCHES, COMPARE AND REGISTER SHIFTS

Operation Field		Address Field	Interpretation
SSH	10	m	Storage shift
SHA	12	y,b	Shift A
SHQ		y,b	Shift Q
SHAQ	13	y,b	Shift AQ
SCAQ		y,b	Scale AQ
CPR,I	52	m,b	Compare (within limits test)
MEQ	06	m,i	Masked equality search
MTH	07	m,i	Masked threshold search

SSH Storage Shift



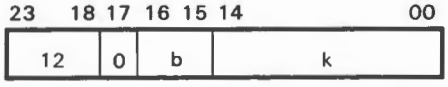
m = storage address

(Approximate execution time: 5.2 μsec.)

Instruction Description: Sense bit 23 of the quantity stored at address m. Shift (m) one place left, end around, and replace it in this same storage location. If bit 23 = "0" (positive), RNI from P + 1; if negative ("1"), RNI from P + 2.

Comments: Address modification may *not* be used.

SHA Shift A



b = index register designator
k = shift count; K = k + (B^b)

(Approximate execution time: 1.8-3.8 μsec.)

Instruction Description: (B^b) and k, with their signs extended, are added. If b = 0, the sign of k is still extended. The sign and magnitude of the 24-bit sum determine the direction and magnitude of the shift. The computer senses only bits 00-05 and 23 of the sum for this information. For left shifts, the shift magnitude is placed in k; to shift right, the complement of the shift magnitude is placed in k.

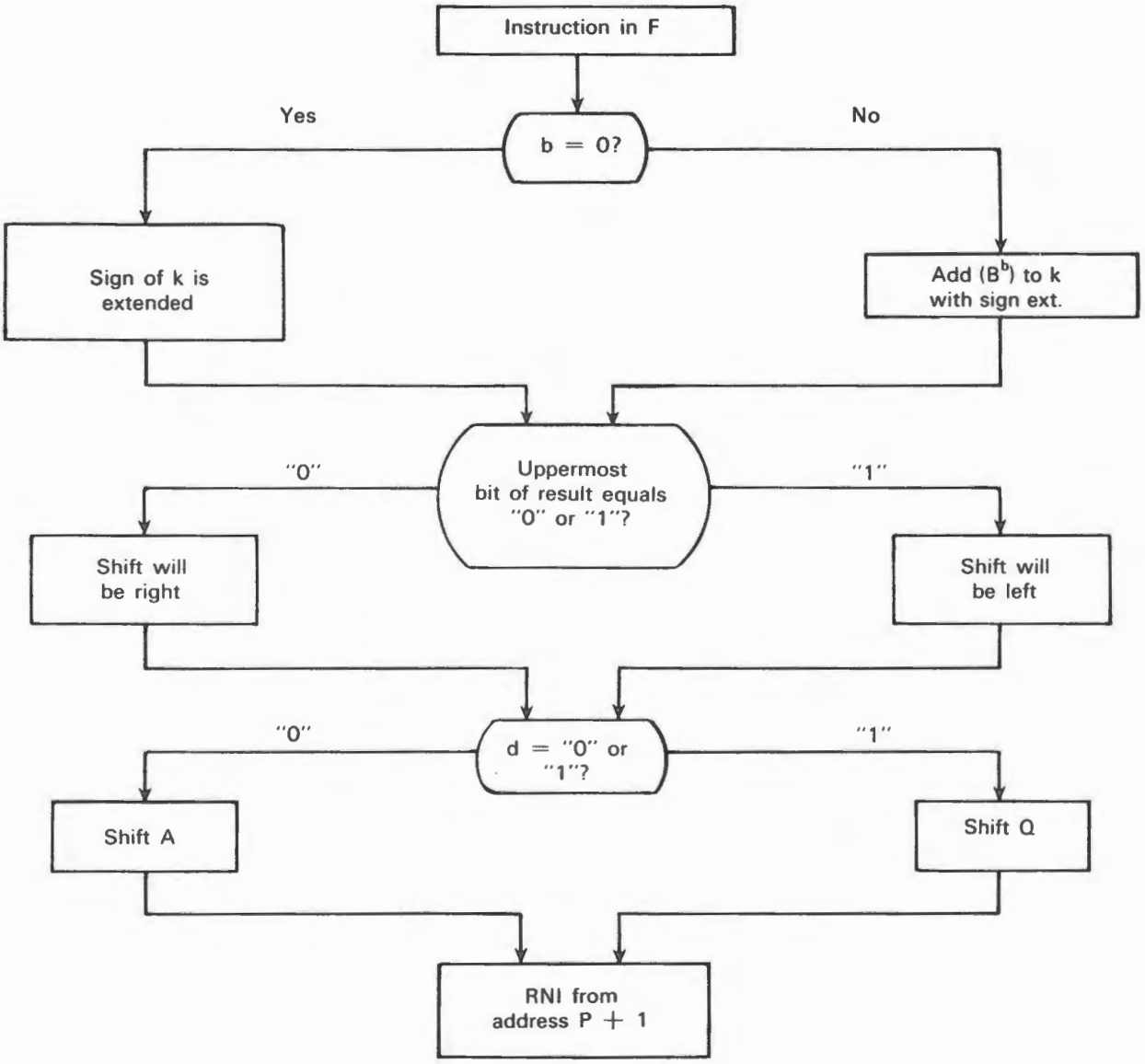
Examples: (b = 0 in both cases):

Shift left six positions: k = 00006
Shift right six positions: k = 77771

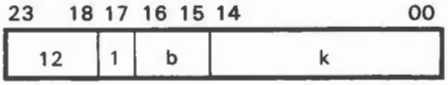
Comments: During left shifts, bits reaching the upper bit position of the A (during SHA) or Q (during SHQ) registers are carried end around. Therefore, a left shift of 24 places results in no change in (A) or (Q). A left shift that exceeds 24 places results in an effective shift of K-24 (or K-48) places.

During right shifts, the sign bit is extended and the bits are shifted end-off. A right shift of 23 or more places results in (A) or (Q) becoming all "0's" or all "1's", depending upon the original sign.

SHA/SHQ FLOW CHART



SHQ Shift Q

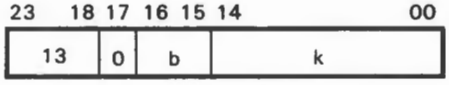


(Approximate execution time: 1.8-3.8 μsec.)

b = index register designator
k = shift count; K = k + (B^b)

Instruction Description: **Refer to SHA description.**

SHAQ Shift AQ



(Approximate execution time: 1.8-3.7 μsec.)

b = index register designator
k = shift count; K = k + (B^b)

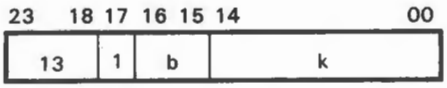
Instruction Description: **(B^b) and k, with their signs extended, are added. If b = 0, the sign of k is still extended. The sign and magnitude of the 24-bit sum determine the direction and magnitude of shift. The computer senses only bits 00-05 and 23 of the sum for this information. For a left shift, the magnitude is placed in k; to shift right, the complement of the shift magnitude is placed in k.**

Examples: (b = 0 in both cases):

Shift left three places:	**k = 00003**
Shift right three places:	**k = 77774**

Comments: **During left shifts bits reaching the upper bit position of the A register are carried end around to the lowest bit position of Q. Therefore, a left shift of 48 places results in no change in (AQ). A left shift exceeding 48 places results in an effective shift of K-48 places.**
During right shifts, the sign bit is extended and the bits are shifted end-off. A right shift of 47 or more places results in (AQ) becoming all "0's" or all "1's", depending upon the original sign.

SCAQ Scale AQ

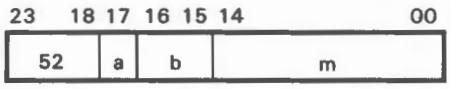


(Approximate execution time: 2.6-5.1 μsec.)

b = index register designator
K = k minus the shift count
$K \rightarrow B^b$

Instruction Description: **(AQ) is shifted left, end around, until the 2 highest order bits (46 and 47) are unequal. If (AQ) should initially equal positive or negative zero, 48_{10} shifts are executed before the instruction terminates. During scaling, the computer counts the number of shifts. A quantity K, called the residue, is equal to k minus the shift count. If b = 0, this quantity is discarded; if b = 1, 2, or 3, the residue is transferred to the designated index register.**

CPR Compare (Within Limits Test)



(Approximate execution time: 3.5 or 4.3 μsec.)

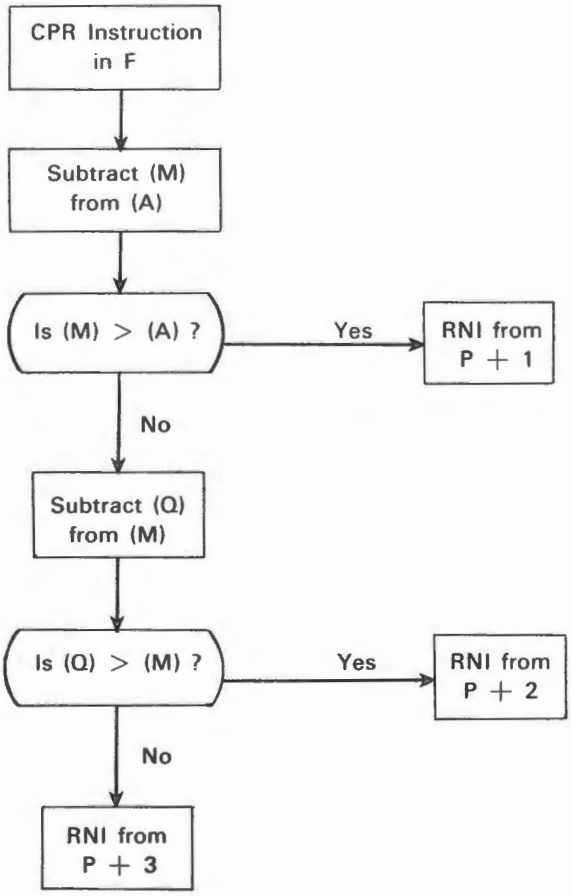
a = addressing mode designator
b = index register designator
m = storage address

Instruction Description: The quantity stored at address M is tested to see if it is within the upper limits specified by A and the lower limits specified by Q. The testing proceeds as follows:

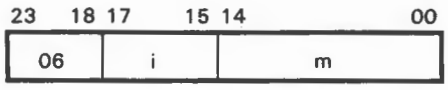
1. Subtract (M) from (A). If (M) > (A), RNI from address P + 1; if not.
2. Subtract (Q) from (M). If (Q) > (M), RNI from P + 2; if not,
3. RNI from address P + 3.

Comments: The final state of the (A) and (Q) registers remains unchanged. (A) must be ≥ (Q) initially or the test cannot be satisfied. 77777777 is not sensed as negative zero. The following table is a synopsis of the CPR test:

Test Sequence	Jump Address if Test Satisfied
(M) > (A)	P + 1
(Q) > (M)	P + 2
(A) ≥ (M) ≥ (Q)	P + 3



MEQ Masked Equality Search



(Approximate execution time: 6.3+6.3n* μsec.)

i = interval designator, 0 to 7
m = storage address

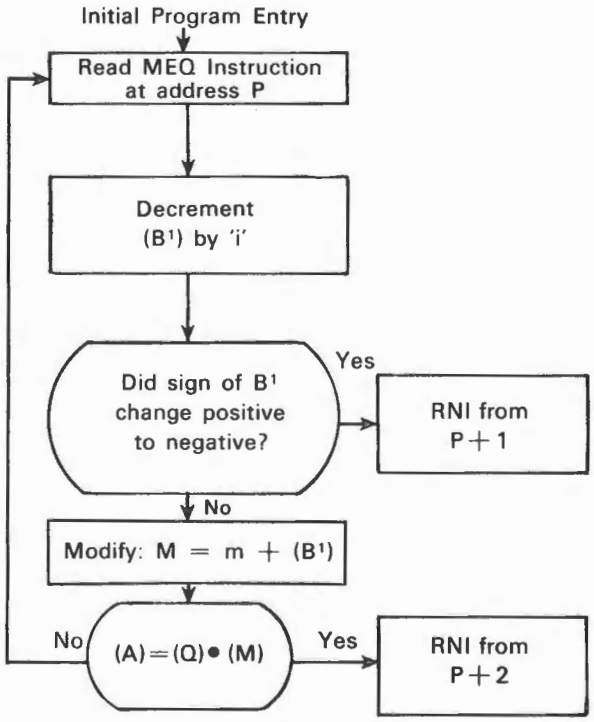
Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B^1 exclusively. m is modified just prior to step 3 in the test below.

Instruction Sequence:

1. Decrement (B^1) by i. (Refer to table below.)
2. If (B^1) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) = (Q) • (M). M = m+(B^1). If (A) =(Q) • (M), RNI from P + 2; if not,
4. Repeat the sequence.

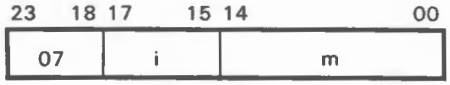
Comments: i is represented by 3 bits, permitting a decrement interval selection from 1 to 8. Address modification may *not* be used. Positive zero and negative zero are recognized as equal quantities.

Designator i	Decrement interval
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8



*n = number of words searched

MTH Masked Threshold Search



(Approximate execution time: 6.3 + 6.3n* μsec.)

i = interval designator, 0 to 7
m = storage address

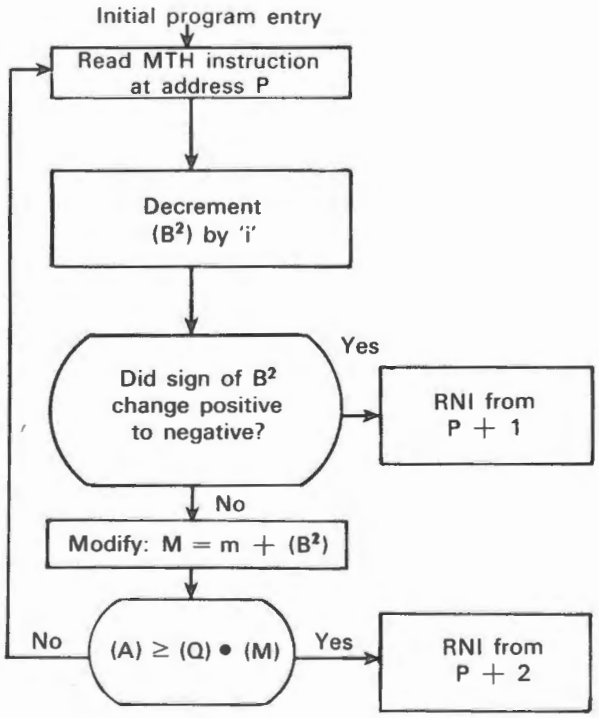
Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B^2 exclusively. m is modified just prior to step 3 in the test below.

Instruction Sequence:

1. Decrement (B^2) by "I". (Refer to table below.)
2. If (B^2) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) ≥ (Q) • (M). M = m + (B^2). If (A) ≥ (Q) • (M), RNI from P + 2; if not,
4. Repeat the sequence.

Comments: i is represented by 3 bits, permitting a decrement interval selection from 1 to 8. Address modification may *not* be used. Positive zero and negative zero are recognized as equal quantities.

Designator i	Decrement interval
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8



*n = number of words searched

SEARCH

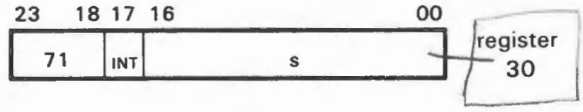
Operation Field	Address Field	Interpretation
71 SRCE,INT	c,r,s	Search for character equality
SRCN,INT	c,r,s	Search for character inequality

GENERAL SEARCH/MOVE NOTE

The SEARCH and MOVE instructions are mutually exclusive. Attempts to execute one while the other is in progress will cause a reject and a skip to address P + 2.

SRCE Search for Character Equality

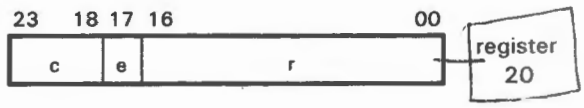
P



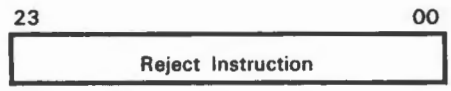
(Approximate execution time: 7.0+7.0n* μsec.)

SRCN Search for Character Inequality

P + 1



P + 2



e = "0" for SRCE, equality
e = "1" for SRCN, inequality
INT = "1" for interrupt upon completion
s = last character address of the search block, plus one
c = 00-77$_8$, BCD code of search character
r = first (current) character address of the search block

Instruction Description: This instruction initiates a search through a block of character addresses in storage looking for equality or inequality with character c. It is composed of three words, including the two main instruction words plus a reject instruction.

As a Search progresses, r is incremented until the search terminates when either a comparison occurs between the search character c and a character in storage, or until r=s. If a comparison does occur, the address of the satisfying character may be determined by inspecting r. To do this, transfer the contents of register 20 to A with instruction TMA (53 0 20020).

Register 20 of the register file is reserved for the second instruction word which contains the current character address of the search block. Register 30 is reserved for the first instruction word which contains the last character address, plus one of the search block.

Figure 7-6 is a flow chart of steps that occur during a search operation.

*n = number of characters compared with c.

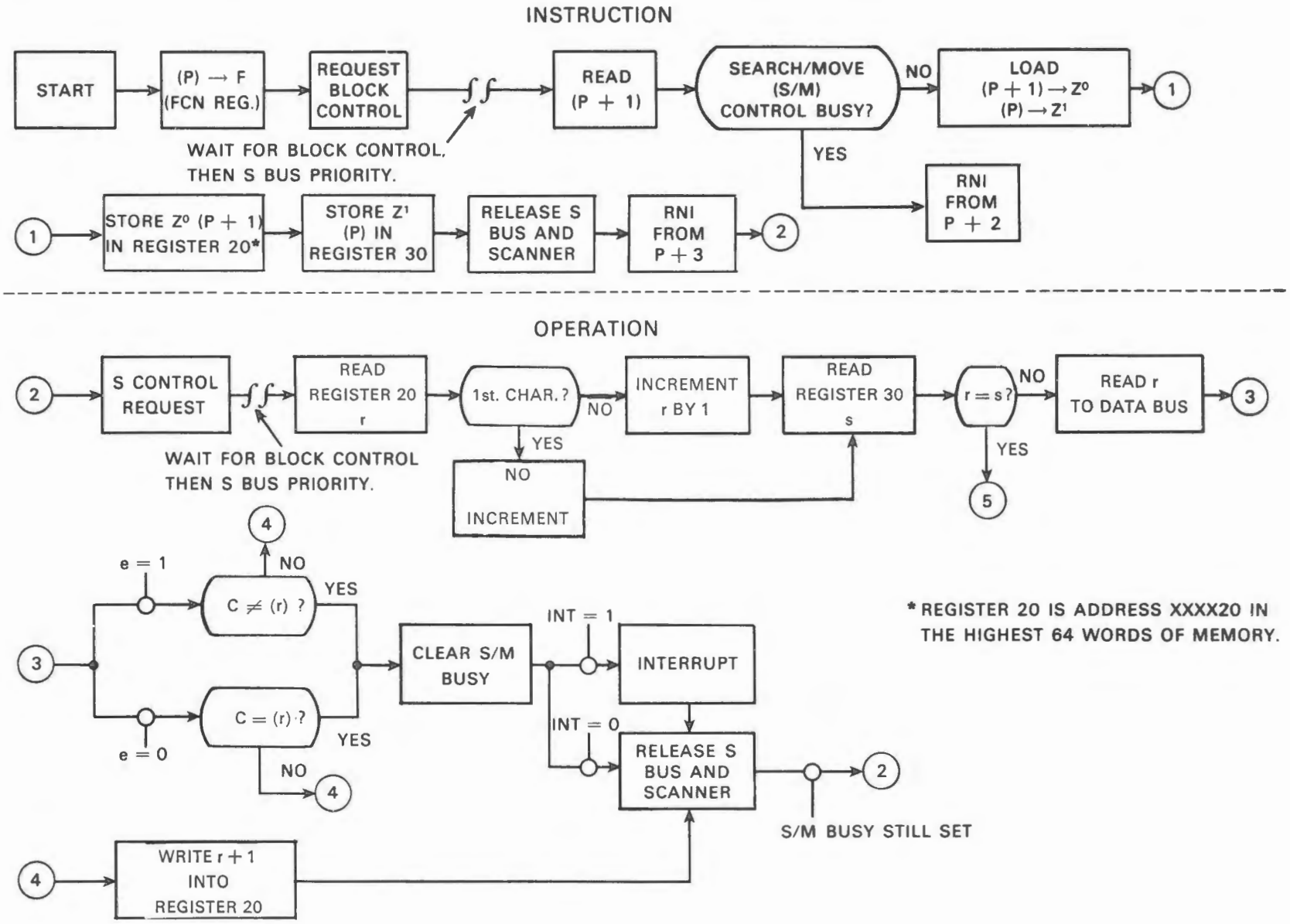
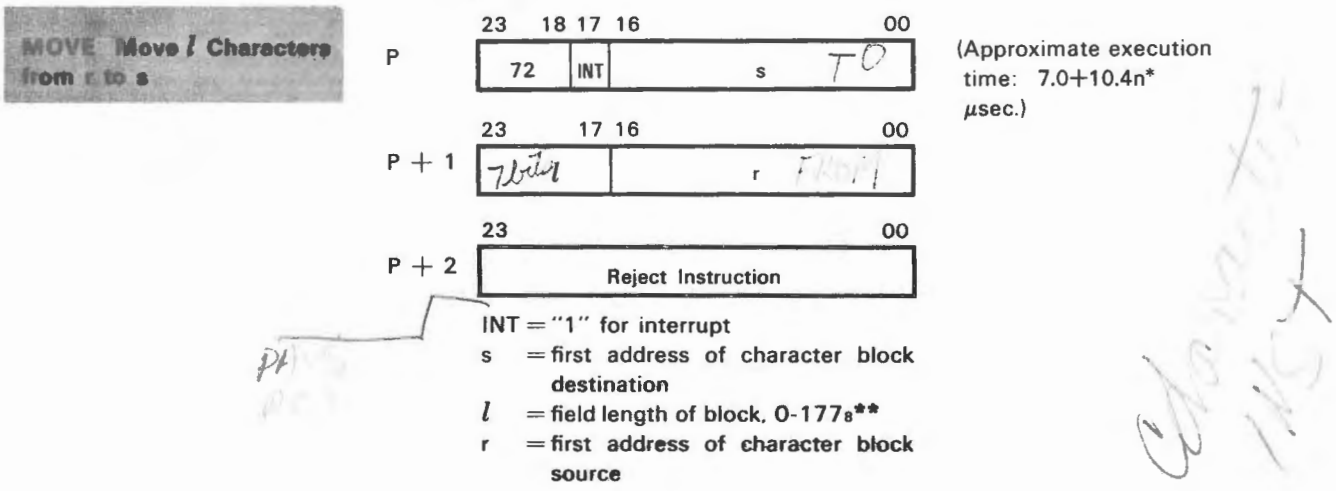


Figure 7-6. Search Operation

MOVE

Operation Field	Address Field	Interpretation
72 MOVE, INT	l,r,s	Move l characters from r to s



Instruction Description: This instruction moves a block of data, l characters long, from one area of storage to another. It is composed of three words, including the two main instruction words, plus a reject instruction.

As a Move operation progresses, r and s are incremented and l is decremented until $l = 0$. 128 characters or 32 words may be moved. When bits 00 and 01 of r and s are "0", and the field length is a multiple of four characters, data is moved word by word. This reduces move time by 75% over a character by character move.

Register 21 of the Register File is reserved for the second instruction word which contains the first address of the character block source. Register 31 is reserved for the first instruction word which contains the first address of the character block destination.

Figure 7-7 is a flow chart of steps that occur during a Move operation.

*n = number of words or characters that are moved.
** = 1-177₈ represents a field length of 1 to 127 characters; 0 represents a field length of 128 characters.

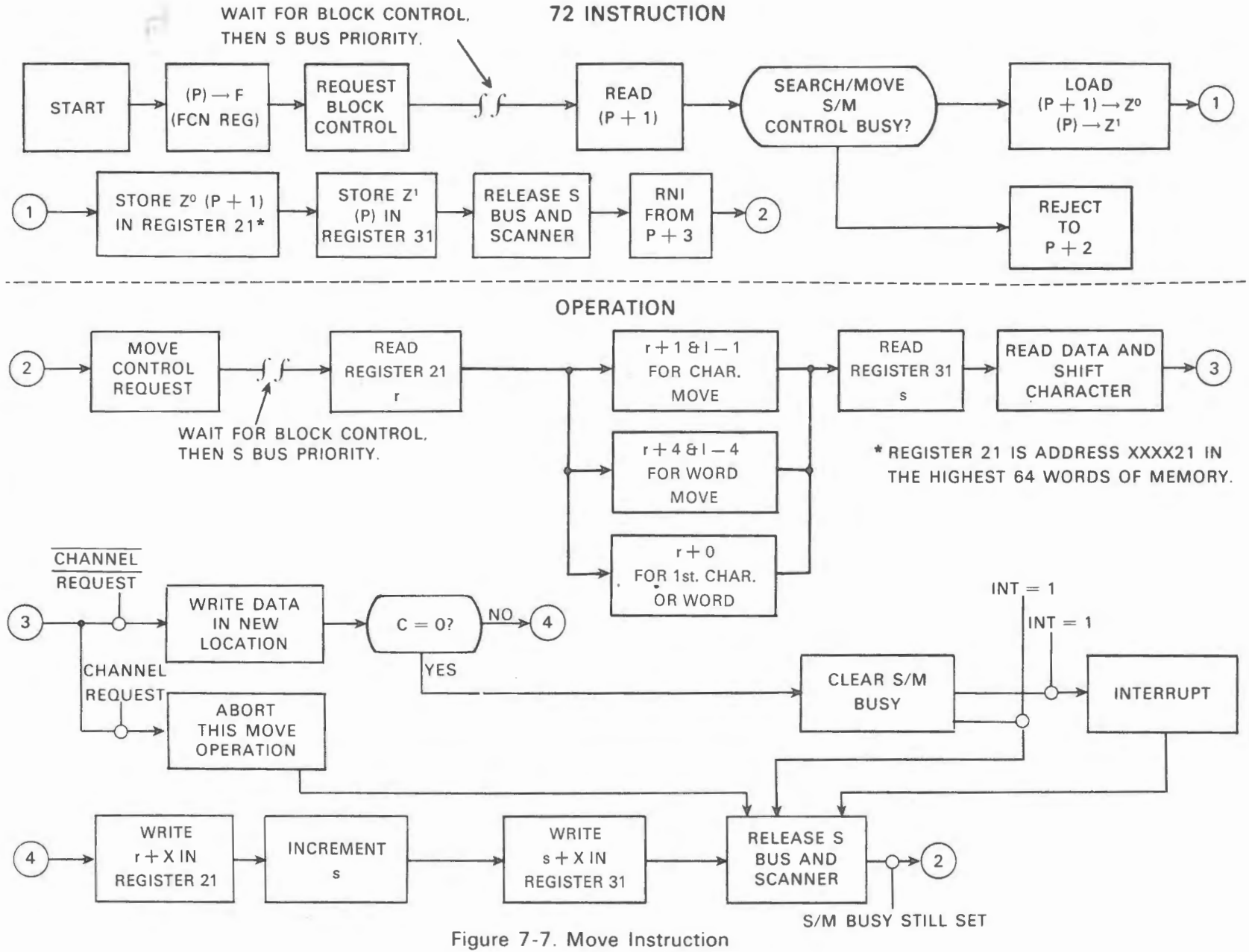
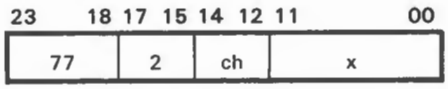


Figure 7-7. Move Instruction

SENSING

Operation		Address	Interpretation
77.2	EXS	x,ch;x≠0	Sense external status
	COPY	x,cn;x=0	Copy external status
77.4	INTS	x,ch	Sense interrupt
77.3	INS	x,ch;x≠0	Sense internal status
	CINS	x,ch;x=0	Copy internal status

EXS Sense External Status



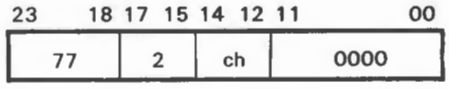
(Approximate execution time: 2.2 μsec.)

ch = I/O channel designator, 0-3
x = external status sensing mask code (see Comments below)

Instruction Description: When a peripheral equipment controller is connected to an I/O channel by the CON (77.0) instruction, the EXS instruction can sense conditions within that controller. Twelve status lines run between each controller and its I/O channel. Each line may monitor one condition within the controller, and each controller has a unique set of line definitions. To sense a specific condition, a "1" is placed in the bit position of the status sensing mask that corresponds to the line number. When this instruction is recognized in a program, RNI at address P + 1 if an external status line is active when its corresponding mask bits are "1". RNI at address P + 2 if no selected line is active.

Comments: Refer to the 3000 Series Computer Systems Peripheral Equipment Codes manual, publication no. 60113400 for a complete list of status response codes.

COPY Copy External Status



(Approximate execution time: 2.2 μsec.)

ch = I/O channel designator, 0-3

Instruction Description: This instruction performs the following functions:

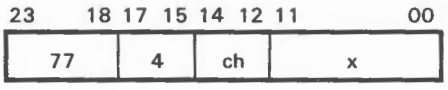
1. The external status code from I/O channel ch is loaded into the lower 12 bits of A. See EXS instruction.
2. The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. See Table 7-4.
3. RNI from address P + 1.

TABLE 7-4. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Mask Bits *	Mask Codes (x)	Interrupt Conditions Represented
00	0001	I/O Channel 0 (includes interrupts generated within the channel and external equipment interrupts)
01	0002	1
02	0004	2
03	0010	3
04	0020	(not used)
05	0040	(not used)
06	0100	(not used)
07	0200	(not used)
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow & divide faults
11	4000	Search/Move completion

*Mask bits 00-03 represent internal and external I/O interrupts for all instructions except INCL.

INTS Sense Interrupt



(Approximate execution time: 2.2 μsec.)

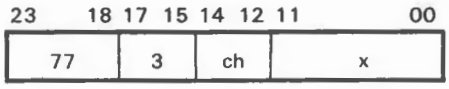
ch = I/O channel designator, 0-3
x = interrupt sensing mask code

Instruction Description: Sense for the interrupt conditions listed in Table 7-4a. RNI from P + 1 if an interrupt line is active and the corresponding mask bit is a "1". If none of the selected lines is active, RNI from P + 2. Internal faults are cleared as soon as they are sensed.

TABLE 7-4a. BIT ASSIGNMENTS FOR INTERRUPT SENSING CONDITIONS

Mask Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	External equipment interrupt line 0 active
01	0002	1
02	0004	2
03	0010	3
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow & divide faults
11	4000	Search/Move completion

INS Sense Internal Status



(Approximate execution time: 2.2 μsec.)

ch = I/O channel designator, 0-3
x = internal status sensing mask code.

Instruction Description: Table 7-5 lists the bit definitions of the internal status sensing mask. Bits 00-04 and 06-07 represent conditions within I/O channel ch. Bits 05 and 08-11, which represent internal faults, may be sensed without regard to channel designation.

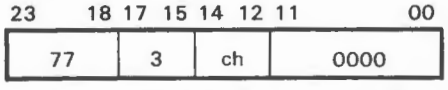
To sense a specific condition, load a "1" into the bit position of the mask that corresponds to the condition. When this instruction is executed, RNI from address P + 1 if an internal status line is active and the corresponding mask bit is a "1". RNI from address P + 2 if none of the selected lines is active. Logic associated with the faults marked by an asterisk in Table 7-5 is cleared as soon as these conditions are sensed.

TABLE 7-5. INTERNAL STATUS SENSING MASK

Mask Bit Positions	Mask Codes (x)	Condition Represented
00	0001	Parity error on channel ch
01	0002	Channel ch busy reading
02	0004	Channel ch busy writing
03	0010	External reject active on channel ch
04	0020	No-response reject active on channel ch
05	0040	*Illegal write
06	0100	Channel ch preset by CON or SEL, but no reading or writing in progress
07	0200	Internal I/O channel interrupt on channel ch, upon: 1) completion of read or write operation, or 2) end of record
08	0400	*Exponent overflow/underflow fault (floating point)
09	1000	*Arithmetic overflow fault (adder)
10	2000	*Divide fault
11	4000	*BCD fault

*Refer to INS instruction description.

CINS Copy Internal Status



(Approximate execution time: 2.2 μsec.)

ch = I/O channel designator, 0-3

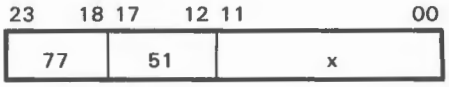
Instruction Description: The CINS instruction performs the following functions:

1. The internal status code is loaded into the lower 12 bits of A. See INS instruction.
2. The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. See Table 7-4.
3. RNI from address P + 1.

CONTROL

Operation Field	Address Field	Interpretation
77.51 IOCL	x	Clear I/O, typewriter, and Search/Move
77.6 PAUS	x	Pause

IOCL Clear I/O, Typewriter, and Search/Move



(Approximate execution time: 1.8 μsec.)

x = block control clearing mask

Instruction Description: This instruction may be used to clear the I/O channels. It also clears all associated peripheral equipment, the typewriter or the Search/Move control according to bits set in the block control clearing mask. (Table 7-6).

TABLE 7-6. BLOCK CONTROL CLEARING MASK

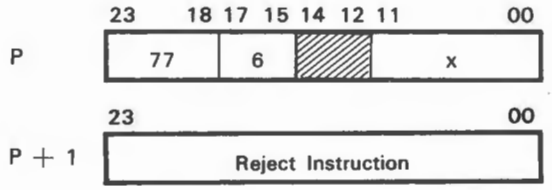
Mask Bits	Mask Codes (x)	Controls Cleared
00	0001	I/O channel 0
01	0002	1
02	0004	2
03	0010	3
04	0020	(not used)
05	0040	(not used)
06	0100	(not used)
07	0200	(not used)
08	0400	Typewriter
09	1000	(see note)
10	2000	(see note)
11	4000	Search/Move

NOTE

If bits 09 and 10 are both set or both clear, the channel(s) specified by bits 00 through 03 of the mask are cleared i.e. Read or Write, Status, and Channel Interrupt are cleared. A 5.5 μsec. Clear signal is also sent to the peripheral equipment and controllers connected to the selected channel(s).

If bit 09 is clear and bit 10 is set, the instruction will clear the channel(s) only and the 5.5 μsec. Clear signal is not transmitted. Bit 08 clears the typewriter as well as the Type Load or Type Dump logic in block control.

PAUS Pause



(Approximate execution time: μsec. to 40 ms.)

x = pause sensing mask code

Instruction Description: This instruction allows the program to halt for a maximum of 40 ms if a condition (excluding typewriter—see note) defined by the pause sensing mask exists. See Table 7-7. If a "1" appears on a line that corresponds to a mask bit that is set, the count in P will not advance. If the advancement of P is delayed for more than 40 ms, a reject instruction is read from address P + 1. If none of the lines being sensed is active, or if they become inactive during the pause, the program immediately skips to address P + 2. If an interrupt occurs and is enabled during a PAUS, the pause condition is terminated, the interrupt sequence is initiated and the address of the PAUS instruction is stored as the interrupted address.

Comments: Bits 12 through 14 of the instruction at P should be loaded with zeros.

NOTE

If either bit 08, 09 or 10 (or any combination of these bits) is set and the sensed condition exists, a pause will not occur and the instruction at P + 1 is read up immediately. If these bit(s) are set but the condition(s) does not exist, the program immediately skips to P + 2. For all other bits, the normal PAUS routine is followed. TYPE FINISH and/or TYPE REPEAT are cleared if bit 09 and/or bit 10 are set and the condition(s) does not exist.

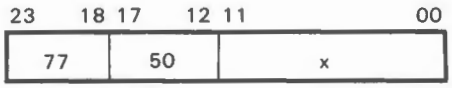
TABLE 7-7. PAUSE SENSING MASK

Mask Bits	Mask Codes	Condition	Notes
00	0001	I/O channel 0 busy	Channel read or write operation in progress, or the External MC logic within the channel is set
01	0002	1	
02	0004	2	
03	0010	3	
04	0020	(Not used)	
05	0040	(Not used)	
06	0100	(Not used)	
07	0200	(Not used)	
[redacted]	0400	Typewriter busy	Typewriter input or output in progress
09	1000	Typewriter NOT finish	Finish logic not set
10	2000	Typewriter NOT repeat	Repeat logic not set
11	4000	Search/Move control busy	Search or Move operation in progress

INTERRUPT

Operation Field	Address Field	Interpretation
77.50 INCL	x	Clear interrupt
77.52 SSIM	x	Selectively set interrupt mask
77.53 SCIM	x	Selectively clear interrupt mask
77.57 IAPR		Interrupt associated processor
77.71 SFPF		Set floating point fault
77.72 SBCD		Set BCD fault
77.73 DINT		Disable interrupt control
77.74 EINT		Enable interrupt control

INCL Clear Interrupt



x = interrupt mask register codes

(Approximate execution time: 1.8 μsec.)

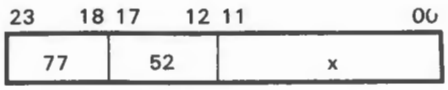
Instruction Description: This instruction clears the interrupt faults defined by the mask codes in Table 7-8. Internal I/O channel interrupts are cleared by this instruction and although the Interrupt Clear is sent to peripheral equipment, not all equipments drop their interrupt lines. Refer to the Peripheral Equipment Reference Manual, Pub. No. 60108800, for information of specific equipment.

TABLE 7-8. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Mask Bits *	Mask Codes (x)	Interrupt Conditions Represented
00	0001	I/O Channel 0 (includes interrupts gener-
01	0002	1 ated within the channel
02	0004	2 and external equipment
03	0010	3 interrupts)
04	0020	(Not used)
05	0040	(Not used)
06	0100	(Not used)
07	0200	(Not used)
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow & divide faults
11	4000	Search/Move completion

*Mask bits 00-03 represent internal and external I/O interrupts for all instructions except INCL.

SetM Selectively Set Interrupt Mask Register



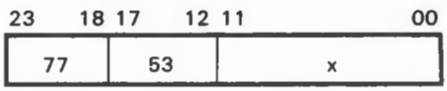
x = interrupt mask register codes

(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction selectively sets the Interrupt Mask register according to the interrupt mask code x. For each bit set to "1" in x, the corresponding bit position in the Interrupt Mask register is set to "1" (see Table 7-8). Bit positions representing missing or nonavailable I/O channels cannot be set.

Comments: This instruction should not be executed when the interrupt system is enabled.

SCIM Selectively Clear Interrupt Mask Register



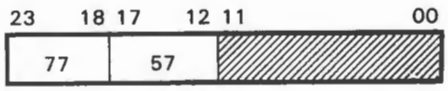
x = interrupt mask register codes.

(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction selectively clears the Interrupt Mask register according to the interrupt mask code x. For each bit set to "1" in x, the corresponding bit position in the Interrupt Mask register is set to "0" (see Table 7-8).

Comments: This instruction should not be executed when the interrupt system is enabled.

IAPR Interrupt Associated Processor

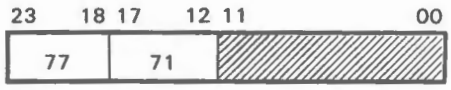


(Approximate execution time: interrupting processor: 1.8 μsec.)

Instruction Description: The processor (computer) executing this instruction sends an interrupt to an associated processor on its left, via storage modules 0 and 1. The interrupt remains active in the receiving computer until it is recognized.

Comments: Bits 00 through 11 should be loaded with zeros.

SFPF Set Floating Point Fault

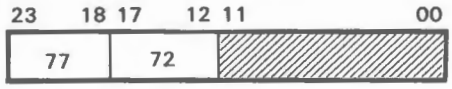


(Approximate execution time: 1.8 μsec.)

Instruction Description: The floating-point fault logic sets when a floating point fault occurs. This instruction is used when the optional floating point arithmetic logic is not present in a system. An interpretive software routine should recognize any conditions which would have caused a fault if the operation had been executed by the optional hardware.

Comments: Bits 00 through 11 should be loaded with zeros.

SBCD Set BCD Fault

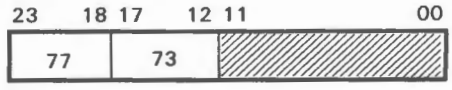


(Approximate execution time: 1.8 μsec.)

Instruction Description: The BCD fault logic sets when a BCD fault occurs. This instruction is used when the optional BCD arithmetic is not present in a system. An interpretive software routine should recognize any condition which would have caused a fault if the operation had been executed by the optional hardware.

Comments: Bits 00 through 11 should be loaded with zeros.

DINT Disable Interrupt Control

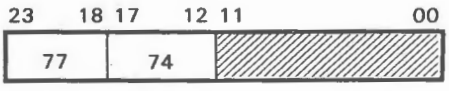


(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction disables the interrupt control system. The system remains disabled until an EINT instruction is executed. Selected interrupts may still be sensed. POWER FAILURE + TRAPPED IST ARE NOT DISABLED

Comments: Bits 00 through 11 should be loaded with zeros.

EINT Enable Interrupt Control



(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction enables the interrupt control system. After executing this instruction, one more instruction will be executed before any interrupt can be recognized.

Comments: Bits 00 through 11 should be loaded with zeros.

Operation Field		Address Field	Interpretation
77.512		cm	Clear channel activity
77.511		cm	External interrupt lockout
77.0	CON	x,ch	Connect to external equipment
77.1	SEL	x,ch	Select function
77.75	CTI		Set console typewriter input
77.76	CTO		Set console typewriter output
73	INPC,INT,B,H	ch,r,s	Character-Addressed Input to storage
	INAC,INT	ch	Character-Addressed Input to A
74	INPW,INT,B,N	ch,m,n	Word-Addressed Input to storage
	INAW,INT	ch	Word-Addressed Input to A
75	OUTC,INT,B,H	ch,r,s	Character-Addressed Output from storage
	OTAC,INT	ch	Character-Addressed Output from A
76	OUTW,INT,B,N	ch,m,n	Word-Addressed Output from storage
	OTAW,INT	ch	Word-Addressed Output from A

I/O operations with storage, unlike operations with A, are buffered. Main computer control relinquishes control of the I/O operations and returns to the main program as soon as Read or Write signals have been activated.

During the execution of word-addressed I/O instructions, the addresses m and n are shifted left two places to the upper 15 bits of the 17-bit address positions. From this time on, they are treated as character addresses.

Registers $00\text{-}17_8$ of the Register File are reserved for I/O operations. The lowest order octal digit (X) of the register designator corresponds to the I/O channel ch being used. Registers $00\text{-}03_8$ are used to hold the instruction word which contains the current character address; $10\text{-}13_8$ hold the instruction word which contains the last character address ±1, depending on the operation. The Register File controls modify bits 21-23 of the first and second I/O instruction words. The modified values, listed in Table 7-9, are predictable. Bits 18 through 23 of register file locations 00, 01, 02 and 03 are used by block control during each I/O transfer—thus, alteration of these bits by a programmer is not recommended. In cases where the addresses require modification to obtain dynamic I/O operations, care should be taken to provide proper read-out and restoration of the control bits. If the instruction cannot be executed, program control jumps to the reject instruction.

If the bit reserved for Interrupt Upon Completion (INT) is a "1" and the mask bit for the affected I/O channel is a "1" and the interrupt system is enabled, the control logic receives a channel-generated interrupt when the output operation is completed. I/O efficiency can be increased by utilizing this bit when applicable.

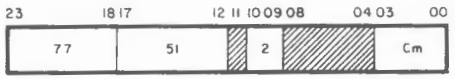
FOR INPC(73) and OUTC(75) instructions with H = 1, an even character count must be specified. If the count is odd, the last character will be lost.

TABLE 7-9. MODIFIED I/O INSTRUCTION WORDS

	Instruction		Instruction Word	Relative Location	Modified Code	Register* Designator
Operations with Storage	73	INPC	1	P	3 - - - - - - -	1X
			2	P + 1	3 - - - - - - -	0X
	74	INPW	1	P	0 - - - - - - -	1X
			2	P + 1	0 - - - - - - -	0X
	75	OUTC	1	P	1 - - - - - - -	1X
			2	P + 1	1 - - - - - - -	0X
	76	OUTW	1	P	2 - - - - - - -	1X
			2	P + 1	2 - - - - - - -	0X
Operations with A	73	INAC	1	P	7 - - - - - - -	1X
			2	P + 1	7 - - - - - - -	0X
	74	INAW	1	P	4 - - - - - - -	1X
			2	P + 1	4 - - - - - - -	0X
	75	OTAC	1	P	5 - - - - - - -	1X
			2	P + 1	5 - - - - - - -	0X
	76	OTAW	1	P	6 - - - - - - -	1X
			2	P + 1	6 - - - - - - -	0X

*X represents an I/O channel designator ch. 0 through 3.

CLCA Clear Channel Activity



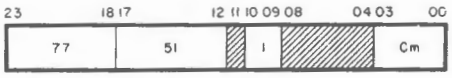
cm = channel mask
Bits 04-08 and 11 should be loaded with zeros.

(Approximate execution time: 1.8 μsec.)

Instruction Description: Clear only the selected I/O channel(s).

Comments: The peripheral equipment associated with the selected channel(s) are not cleared by executing this instruction. Bit 00 corresponds to channel 0, bit 01 corresponds to channel 1. More than one channel may be set to "1" for multiple channel clearing.

CILO Channel* Interrupt Lockout



cm = channel mask
Bits 04-08 and 11 should be loaded with zeros.

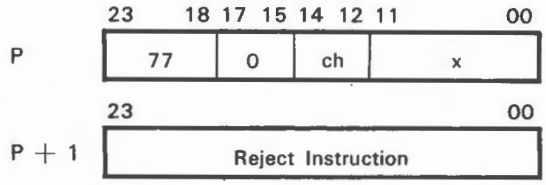
(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction disables all external interrupts on channel(s) 'cm' while the channel(s) are busy (see footnote).

Comments: Bit 00 corresponds to channel 0, bit 01 corresponds to channel 1. More than one channel may be set to "1" for multiple channel interrupt lockout. The mask is cleared by termination of the I/O operation, by clearing the channel(s), or by a Negate Channel Interrupt Lockout signal from certain peripherals.

*The CILO instruction is not available on the 3104-A/B or 3114-A/B Computer.

CON Connect



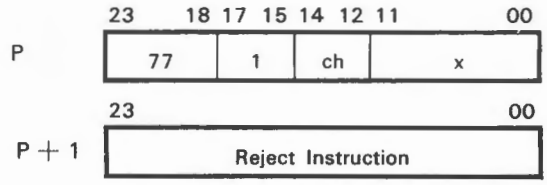
(Approximate execution time: 2.3 μsec. or until Reply or Reject)

ch = I/O channel designator, 0-3

x = 12 bit connect code. Bits 09-11 select one of eight controllers which may be attached to channel ch. Bits 00-08 select the peripheral units connected to the controller.

Instruction Description: **This instruction sends a 12-bit connect code along with a connect enable to an external equipment controller on I/O channel ch. If a Reply is received from the controller within 100 μsec, the next instruction is read from address P + 2. If a Reject is received or there is no response within 100μsec, a reject instruction is read from address P + 1. If the I/O channel is busy, a reject instruction is read from address P + 1.**

Select Function



(Approximate execution time: 2.3 μsec. or until Reply or Reject)

ch = I/O channel designator, 0-3

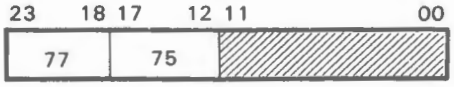
x = 12-bit function code. Each piece of external equipment has a unique set of function codes to specify operations within that device. Refer to the 3000 Series Computer Systems Peripheral Equipment Codes publication No. 60113400 for a complete list of function codes.

Instruction Description: **This instruction sends a 12-bit function code along with a function enable to the unit connected to I/O channel ch. If a Reply is received from the unit within 100 μsec, the next instruction is read from P + 2. If a Reject is received or there is no response within 100 μsec, a reject instruction is read from address P + 1. If the I/O channel is busy, a reject instruction is read from address P + 1.**

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 μsec, the Reject signal is generated automatically by the I/O channel.**
2) *Undefined Code:* **When the Function code x is not defined for the specific device, a Reject may be generated by the device. However, in some cases an undefined code will cause the device to generate a Reply although no operation is performed. (Refer to the reference manual for the specific device.)**
3) *Equipment or Unit Busy or Not Ready:* **The device cannot perform the operation specified by the function code x without damaging the equipment or losing data. For example, a Write End of File code is rejected by a tape unit if the tape unit is rewinding.**
4) *Channel Busy:* **The selected data channel is currently performing a Read or Write operation.**

CTI Set Console Typewriter Input

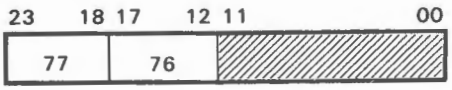


(Approximate execution time: 1.8 μsec.)

Instruction Description: This instruction, like the TYPE LOAD switch, permits a block of data to be entered into storage as soon as the Type Load indicator lights. If a block of data smaller than the one defined by registers 23 and 33 is to be typed, the FINISH switch should be depressed when the typing is completed. If more data is entered than the defined block can hold, the excess data is lost. If a typing error occurs, the REPEAT button should be depressed. When either the FINISH or REPEAT switches are depressed, the typewriter input operation is terminated and the appropriate status bits (09 and 10) may be sensed with the PAUS instruction. Refer to page 7-64 for additional information on the PAUS instruction.

Comments: **Bits 00 through 11 should be loaded with zeros.**

CTO Set Console Typewriter Output



(Approximate execution time: 1.8 μsec.)

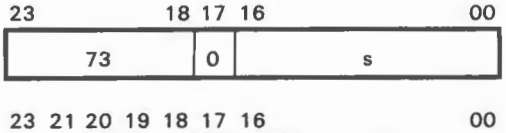
Instruction Description: This instruction, like the TYPE DUMP switch, causes the typewriter to print out the block of data defined by the character addresses in registers 23 and 33.

Comments: Bits 00 through 11 should be loaded with zeros.

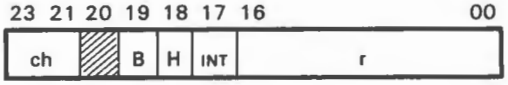
NOTE

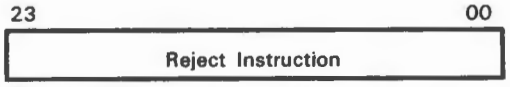
The CTI and CTO instructions are mutually exclusive. Any attempt to execute one while the other is being executed will be ignored by the computer. Typewriter busy should be checked before these instructions are used and before registers 23 and 33 are altered.

INPC Character-Addressed Input to Storage



(Approximate execution time: $7.0 + 7.0n^{*}$ μsec.)

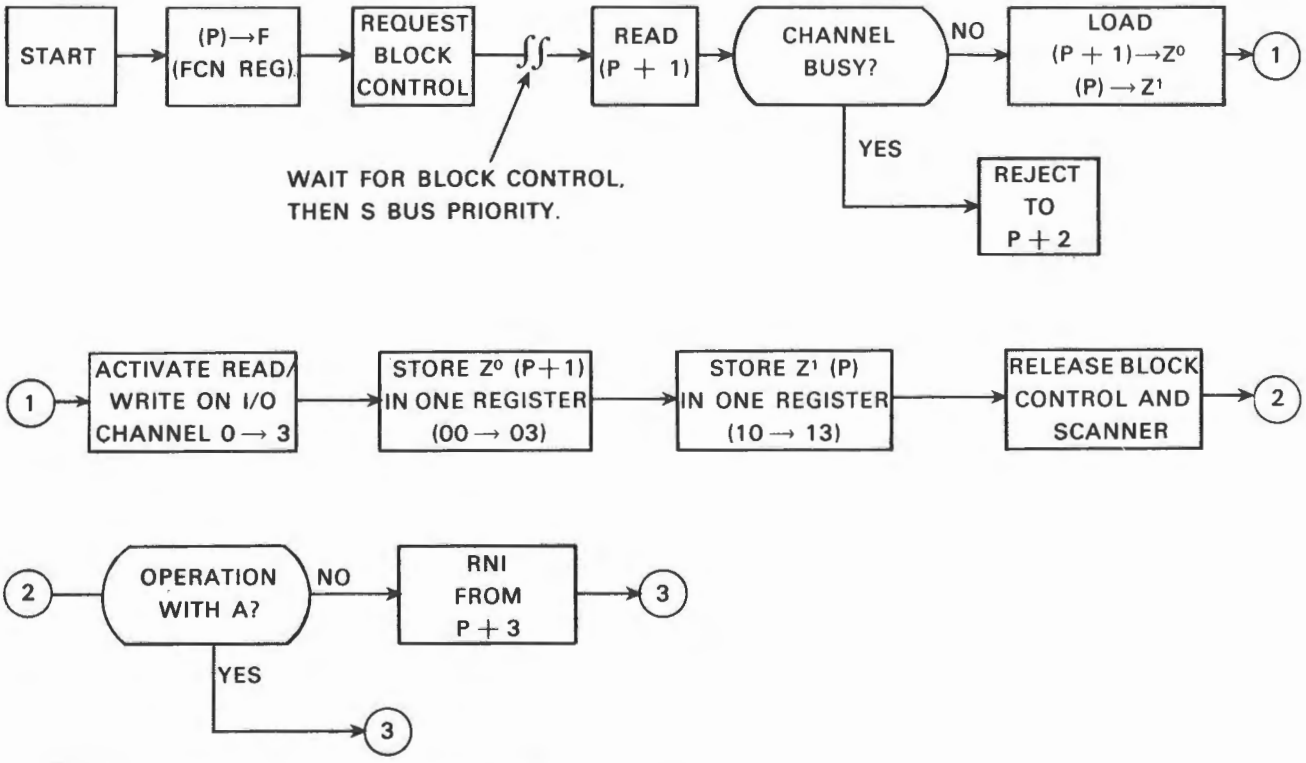




B = "1" for backward storage
ch = I/O channel designator, 0-3
H = "0" for 6- to 24-bit assembly
H = "1" for 12- to 24-bit assembly
INT = "1" for interrupt upon completion
r = first character address of I/O data block; becomes current address as I/O operation progresses
s = last character address of input data block, plus one (minus one, for backward storage)

Instruction Description: This instruction transfers a character-address block of data, consisting of 6-bit characters or 12-bit bytes, from an external equipment to storage. During 12- to 24-bit assembly, the lowest bit of each character address is forced to remain a "0" in register OX. This ensures that assembled bytes are in either the upper or the lower half of the word being stored. If H = 1, an even character count must be used. If the count is odd, the last character will be lost.

INSTRUCTION SEQUENCE



*n = number of characters or 12-bit bytes transferred.

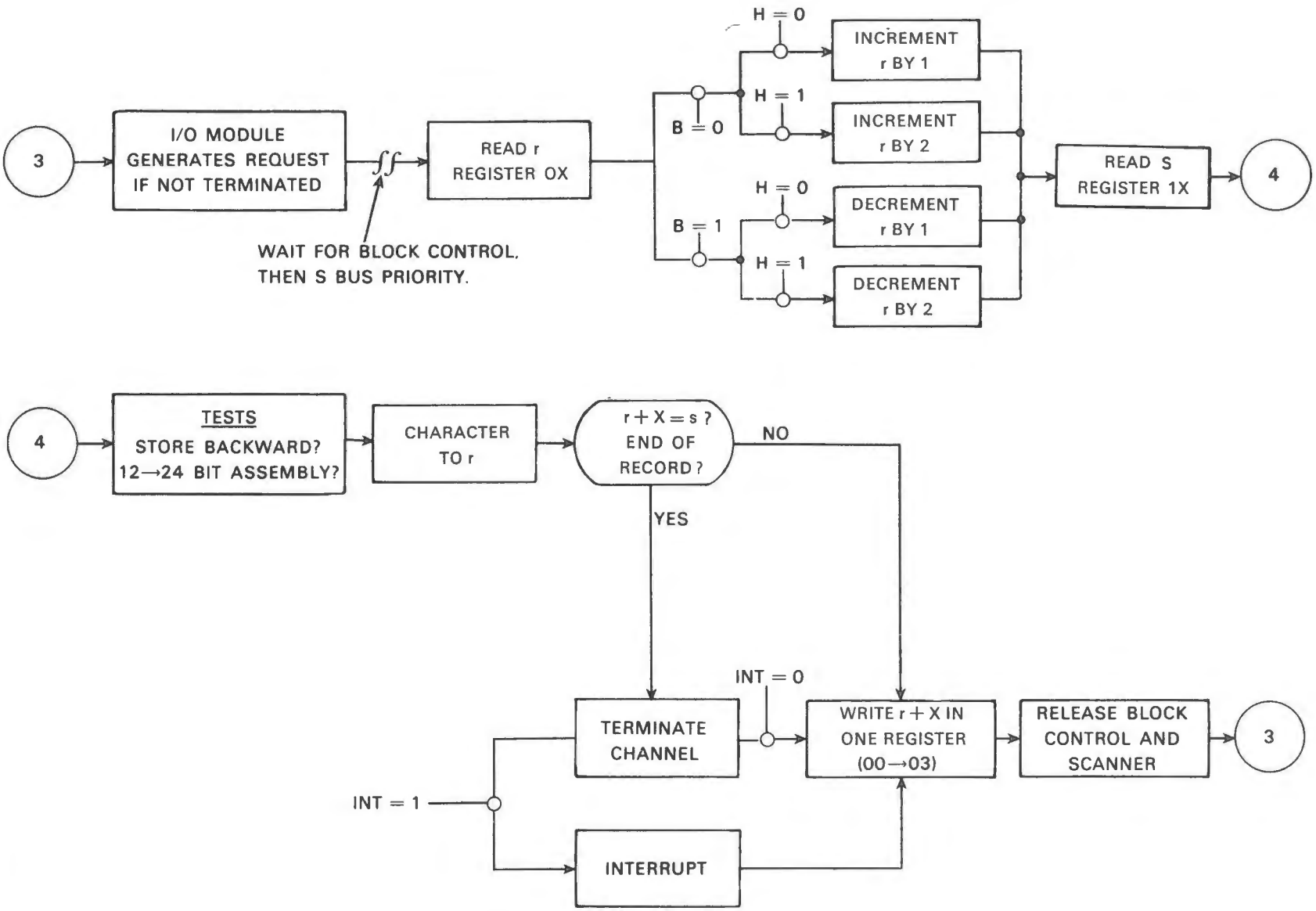
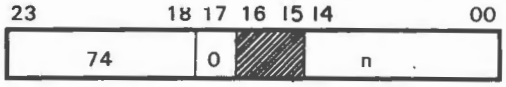
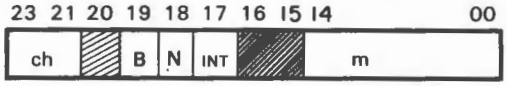


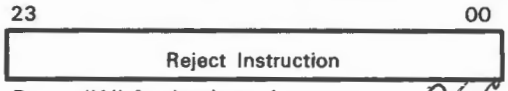
Figure 7-8. 73 I/O Operation with Storage

INPW Word Addressed Input to Storage



(Approximate execution time: 7.0+7.0n* μsec.)





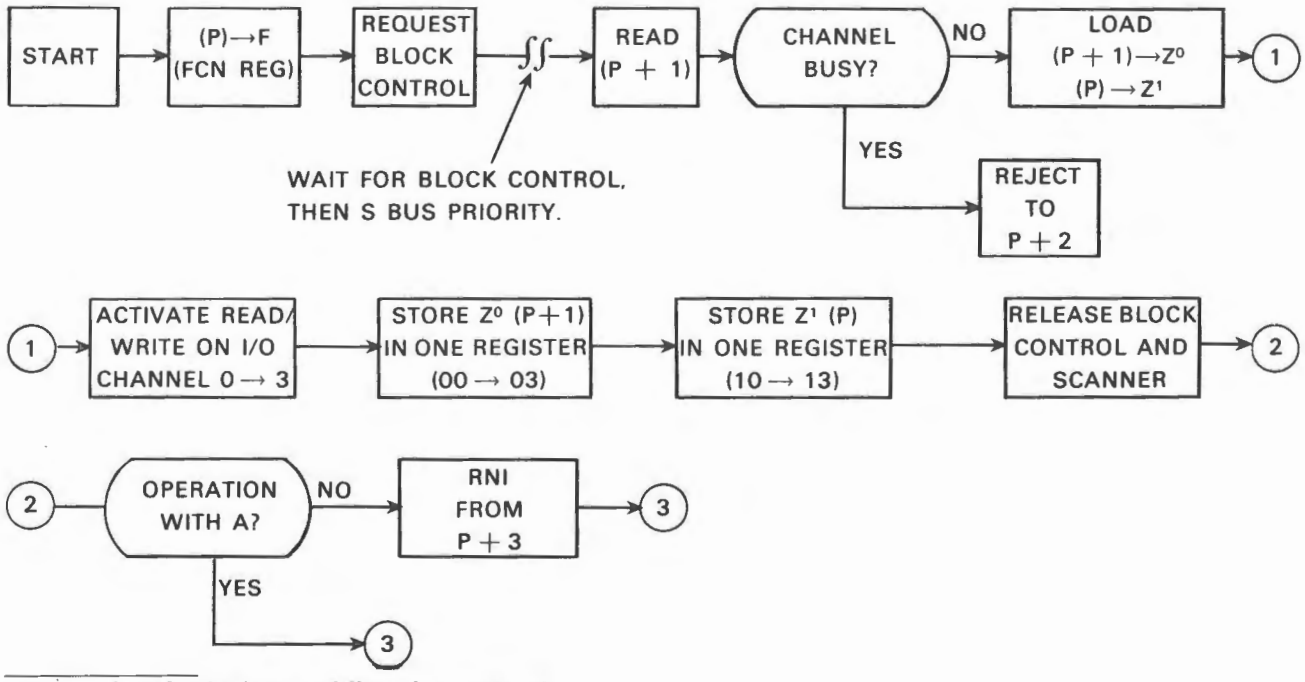
B = "1" for backward storage
ch = I/O channel designator, 0-3
INT = "1" for interrupt upon completion
N = "0" for 12- to 24-bit assembly
N = "1" for no assembly
m = first word address of I/O data block; becomes current address as I/O operation progresses
n = last word address of input data block, plus one (minus one, for backward storage)

Instruction Description: This instruction transfers a word-addressed data block from an external equipment to storage. Transferring 12-bit bytes or 24-bit words depends upon the type of I/O channel used. The 3106 utilizes 12-bit bytes and the 3107 uses 24-bit words.

During forward storage and 12- to 24-bit assembly, the first byte of a block of data is stored in the upper half of the memory location specified by the storage address. Conversely, during backward storage, the first byte is stored in the lower half of the memory location.

I/O OPERATION WITH STORAGE

INSTRUCTION SEQUENCE



*n = number of 12-bit bytes or full words transferred.

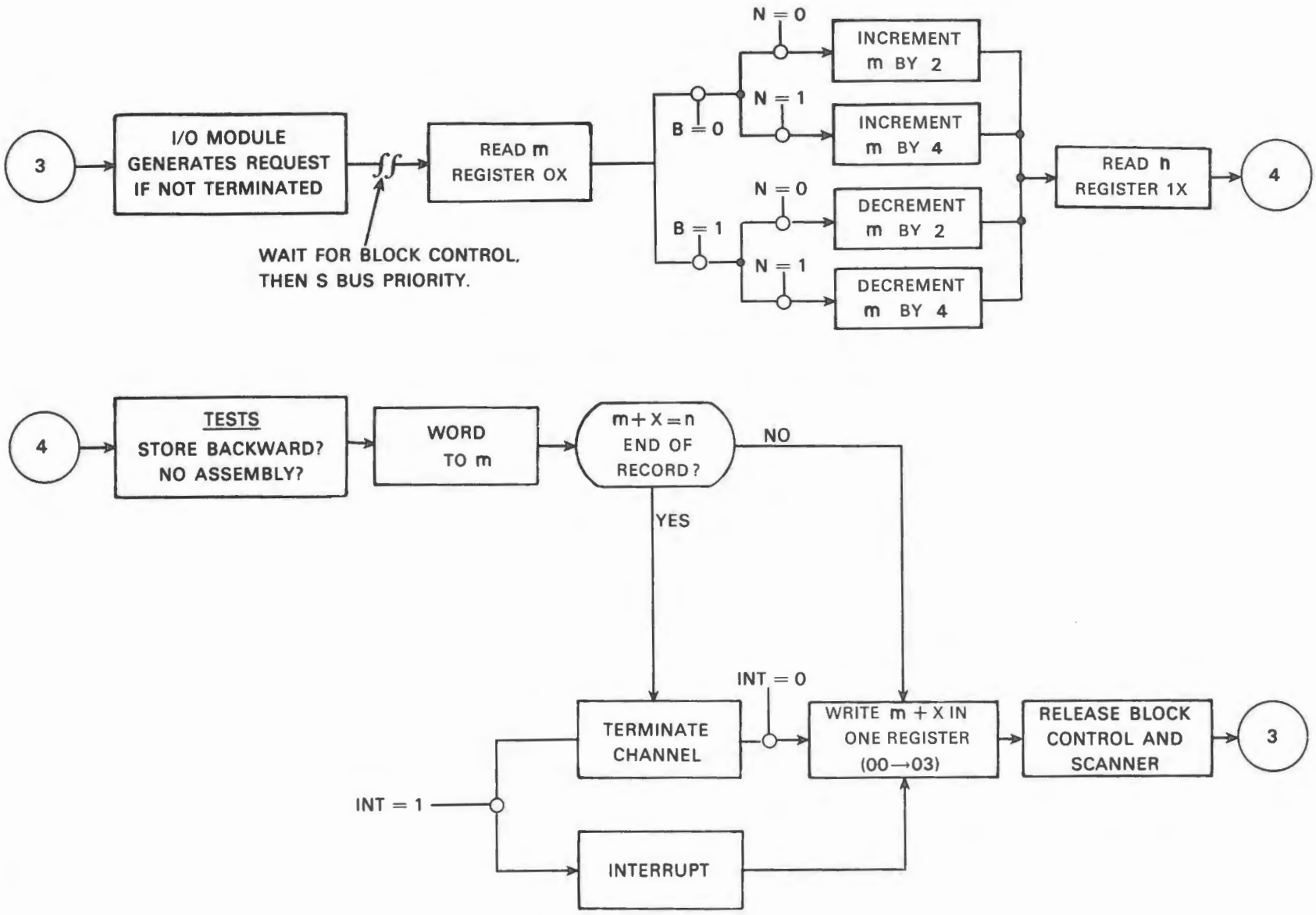
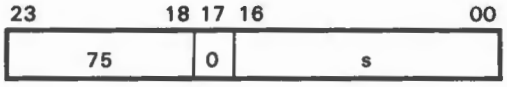
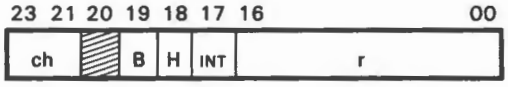


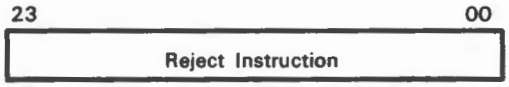
Figure 7-9. 74 I/O Operation with Storage

OUTC Character-Addressed Output from Storage



(Approximate execution time: 7.0 + 7.0n μsec.)



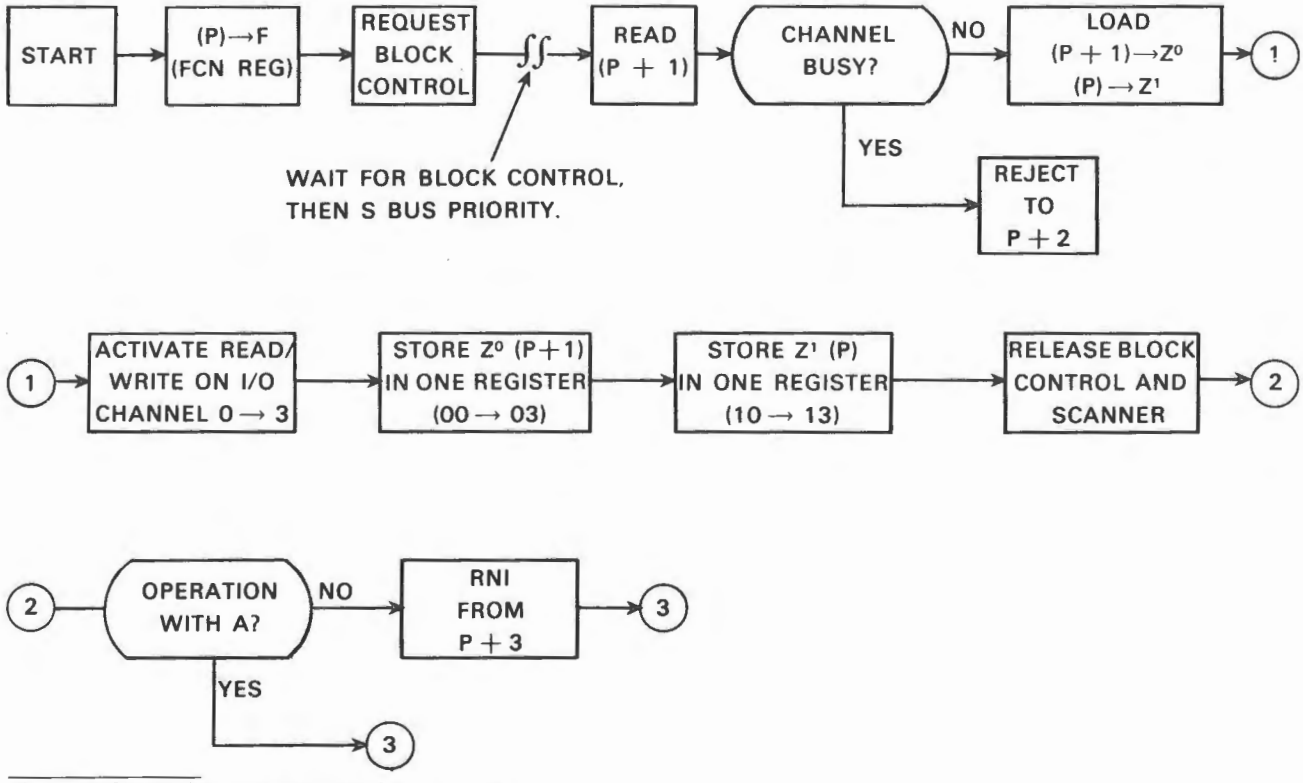


B = "1" for backward storage
ch = I/O channel designator, 0-3
H = "0" for 24- to 6-bit disassembly
H = "1" for 24- to 12-bit disassembly
INT = "1" for interrupt upon completion
r = first character address of I/O data block; becomes current address as I/O operation progresses
s = last character address of output data block, plus one (minus one, for backward output)

Instruction Description: **This instruction transfers a character-addressed block of data, consisting of 6-bit characters or 12-bit bytes, from storage to an external equipment. If H = 1, an even character count must be used. If the count is odd, the last character will be lost.**

I/O OPERATION WITH STORAGE

INSTRUCTION SEQUENCE



*n = number of characters of 12-bit bytes transferred.

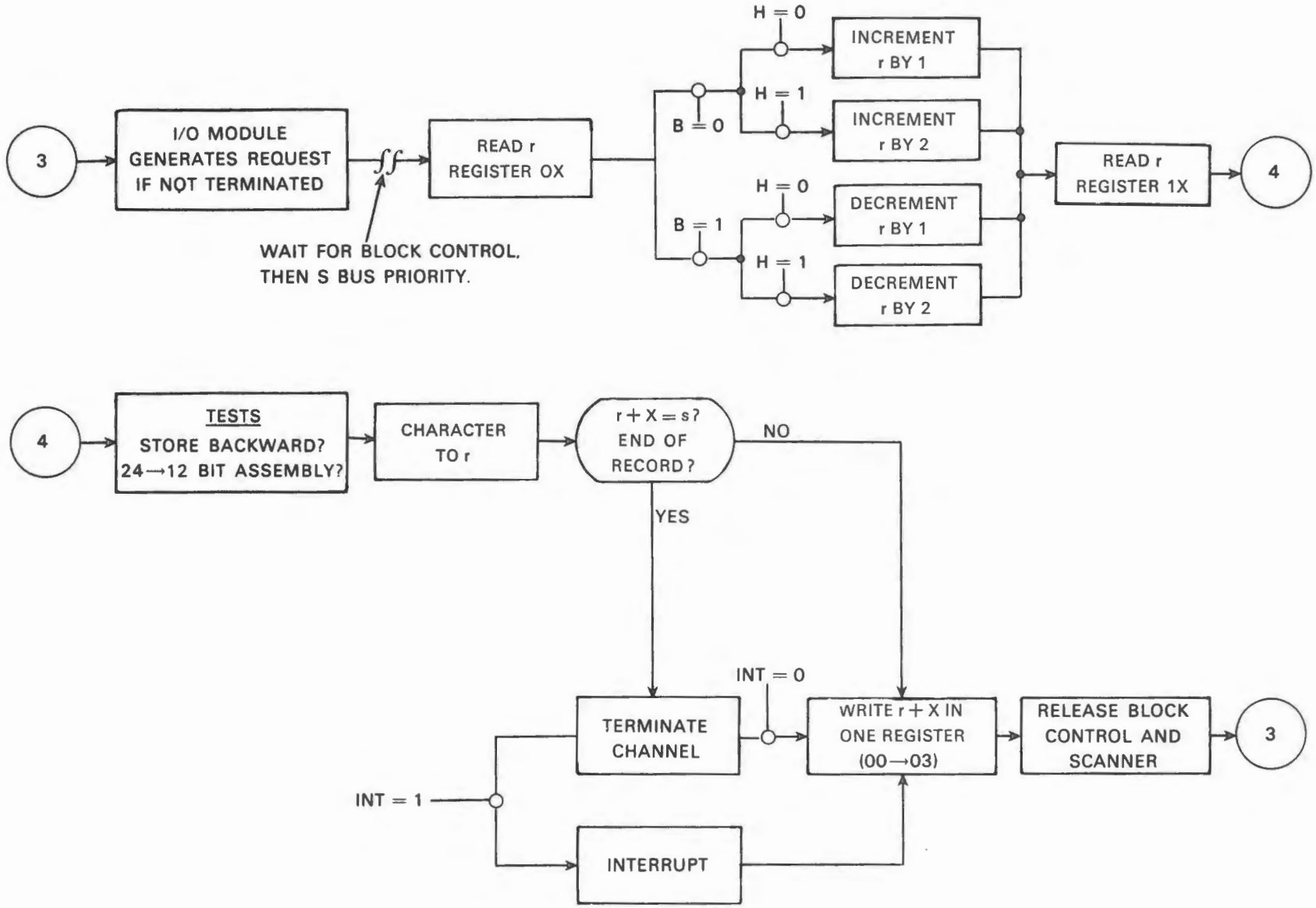
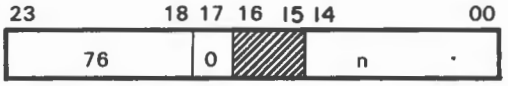
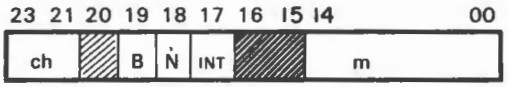


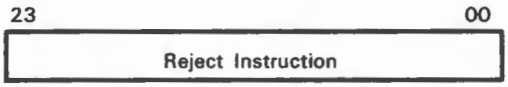
Figure 7-10. 75 I/O Operation with Storage

OUTW Word-Addressed Output from Storage



(Approximate execution time: 7.0 + 7.0n* μsec.)





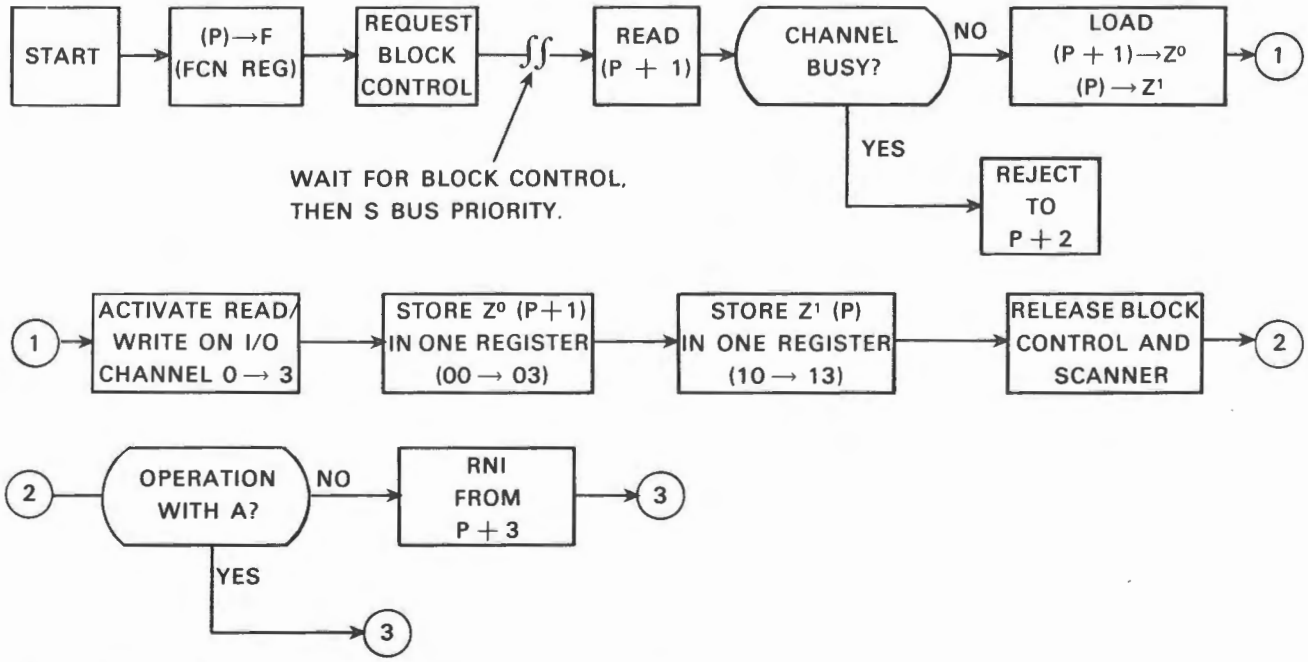
B = "1" for backward storage
ch = I/O channel designator
INT = "1" for interrupt upon completion
m = first word address of I/O data block; becomes current address as I/O operation progresses
N = "0" for 24- to 12-bit disassembly
N = "1" for straight 12- or 24-bit data transfer
n = last word address of output data block, plus one (minus one, for backward output)

Instruction Description: This instruction transfers a word-addressed block of data consisting of 12-bit bytes or 24-bit words, from storage to an external equipment.

With no disassembly, 12 or 24-bit transfer capability depends upon whether a 3106 or 3107 I/O channel is used. If an attempt is made to send a 24-bit word over a 3106 I/O channel, the upper byte will be lost.

I/O OPERATION WITH STORAGE

INSTRUCTION SEQUENCE



*n = number of 12-bit bytes or full words transferred.

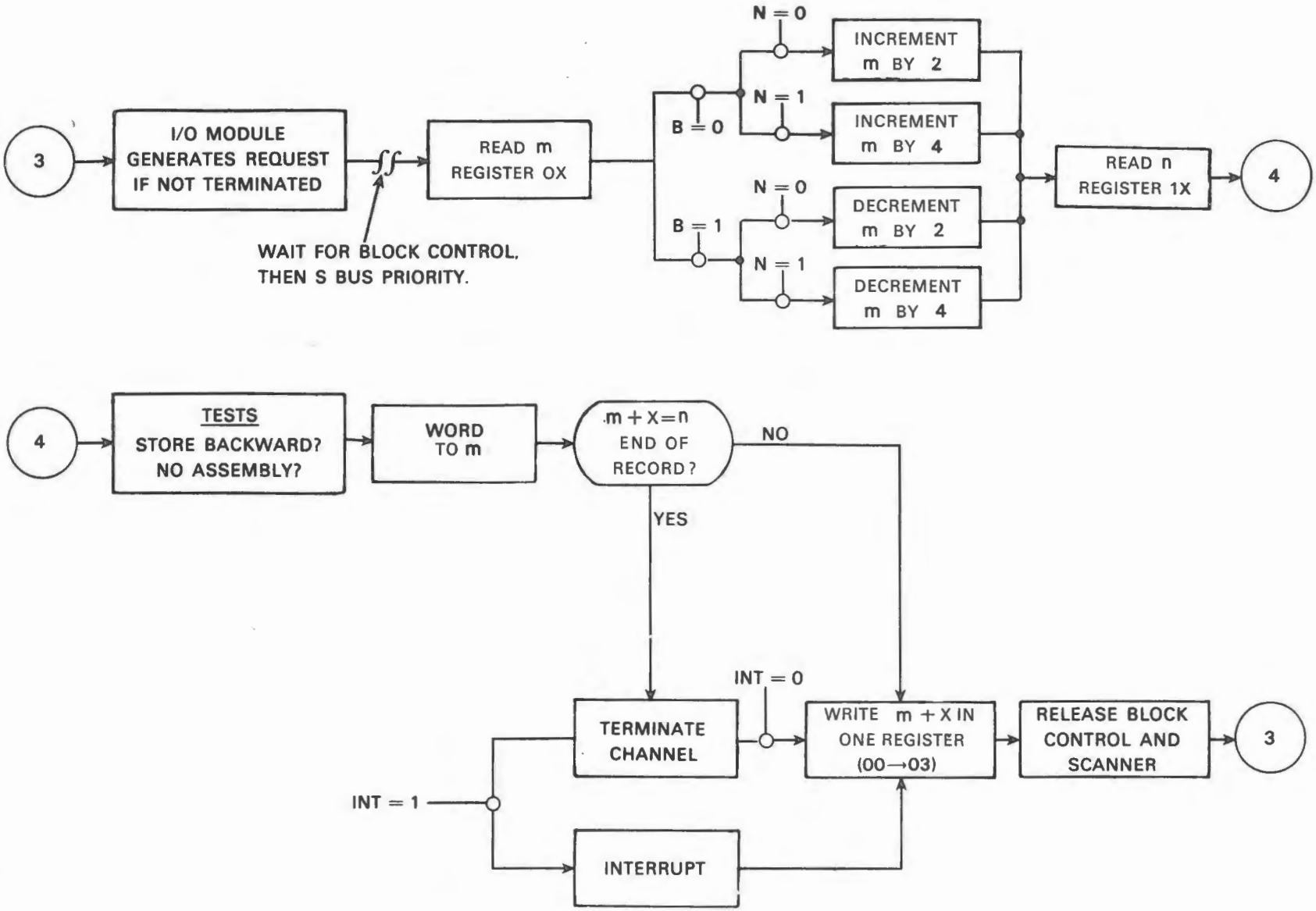
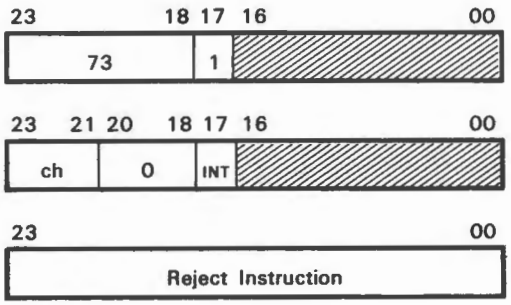


Figure 7-11. 76 I/O Operation with Storage

INAC Input, Character to A



(Approximate execution time: 7.0 μsec. + response time.)

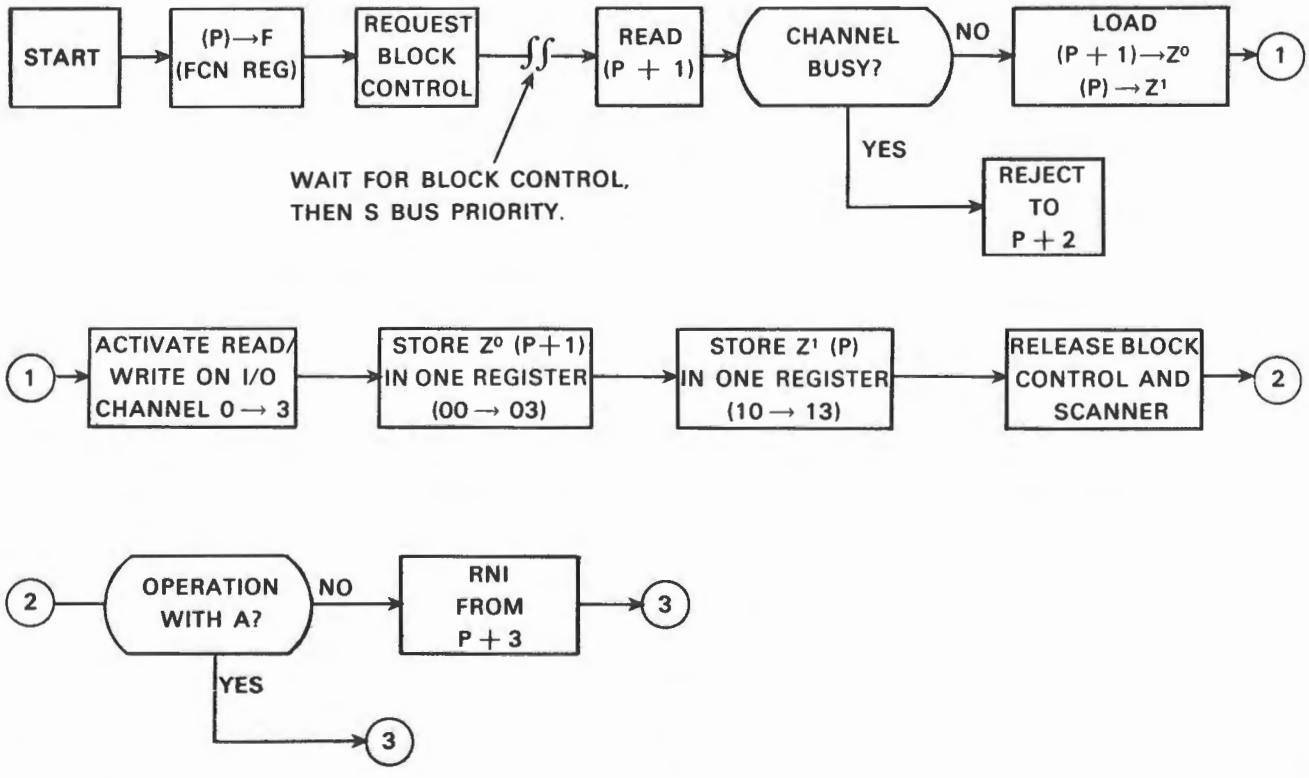
ch = I/O channel designator, 0-3
INT = "1" for interrupt upon completion

Instruction Description: **This instruction transfers a 6-bit character from an external equipment into the lower six bits of the A register. A is cleared prior to loading, and the upper 18 bits remain cleared.**

Comments: **Bits 00-16 at P and P + 1 should be loaded with zeros.**

I/O OPERATION WITH A

INSTRUCTION SEQUENCE



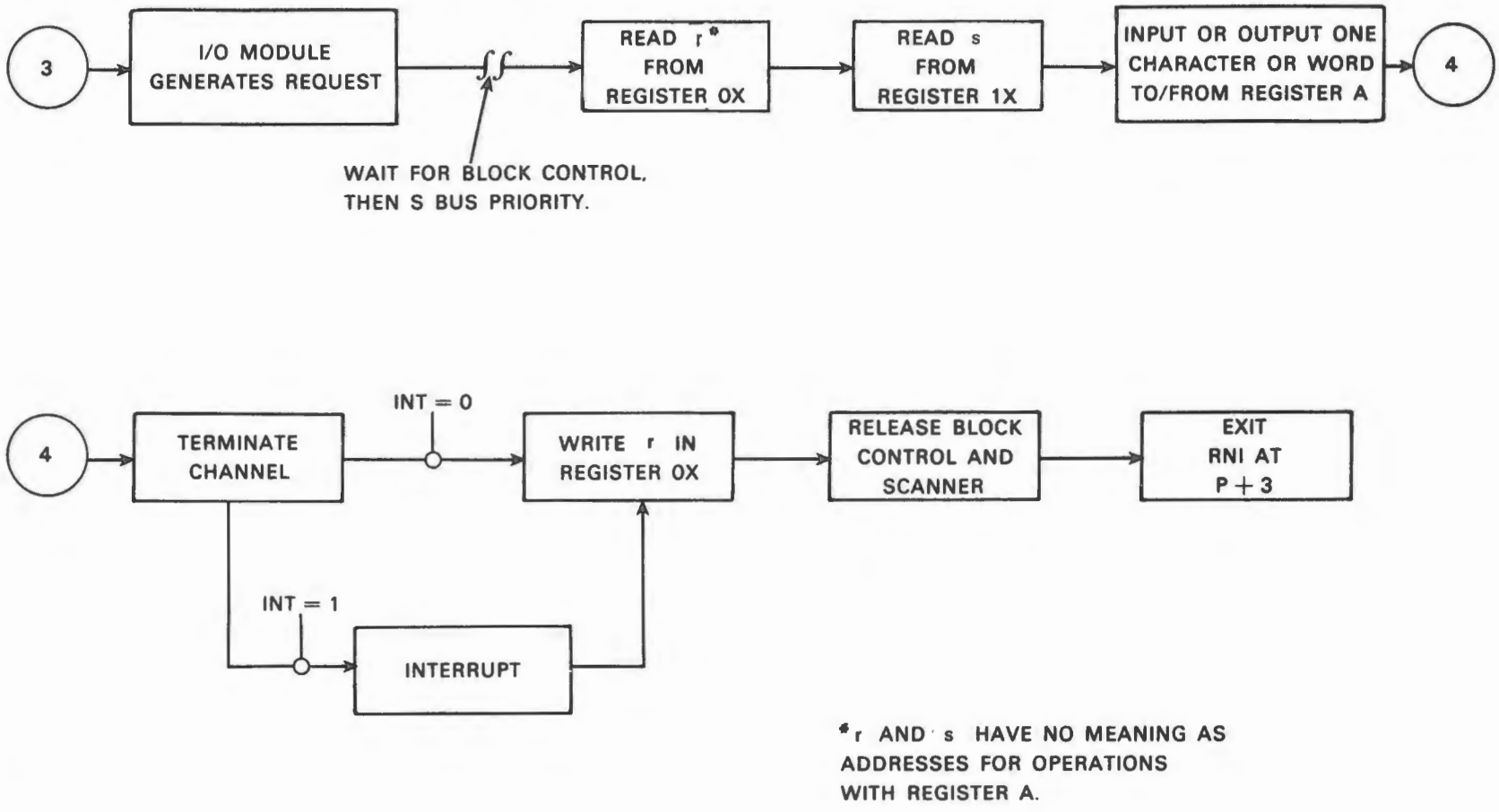
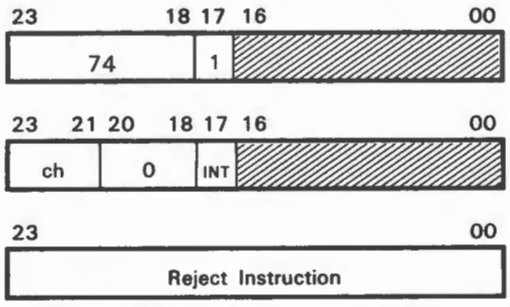


Figure 7-12. 73 I/O Operation with A

INAW Input, Word to A



(Approximate execution time: 7.0 μsec. + response time.)

ch = I/O channel designator, 0-3
INT = "1" for interrupt upon completion

Instruction Description: This instruction transfers a 12-bit byte into the lower 12 bits of A or a 24-bit word into all of A from an external equipment. Transferring 12 or 24 bits depends upon whether a 3106 or 3107 I/O channel is used. (A) is cleared prior to loading and, in the case of a 12-bit input, the upper 12 bits remain cleared.

Comments: Bits 00-16 at P and P + 1 should be loaded with zeros.

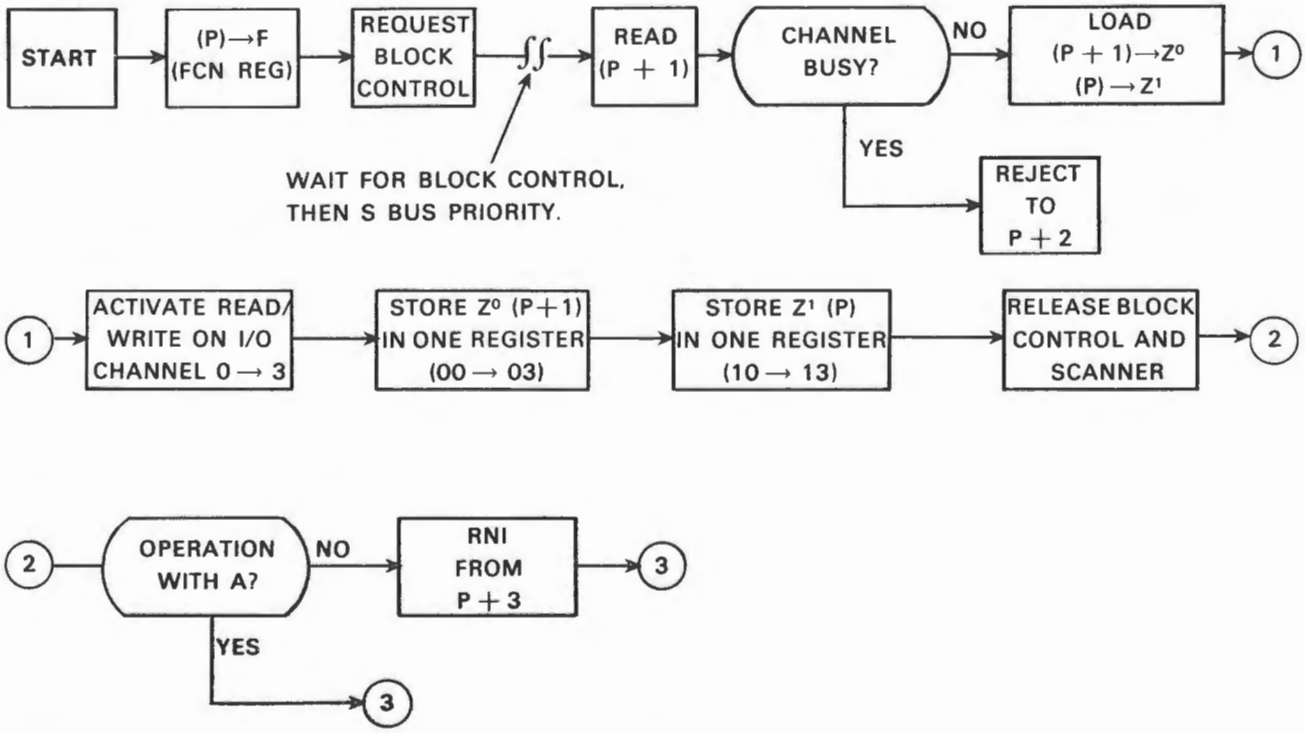
NOTE

Bits 18, 19, and 20 are all zeros when a 3106 data channel is used. If the operation with A involves the use of a 3107, these bits take on the following significance:

Bit 20 = always a "0".
Bit 19 = If bit 18 = "1", the state of bit 19 is of no consequence.
If bit 18 = "0", a "1" in bit 19 signifies backward operation.
A "0" in bit 19 signifies a forward operation.

I/O OPERATION WITH A

INSTRUCTION SEQUENCE



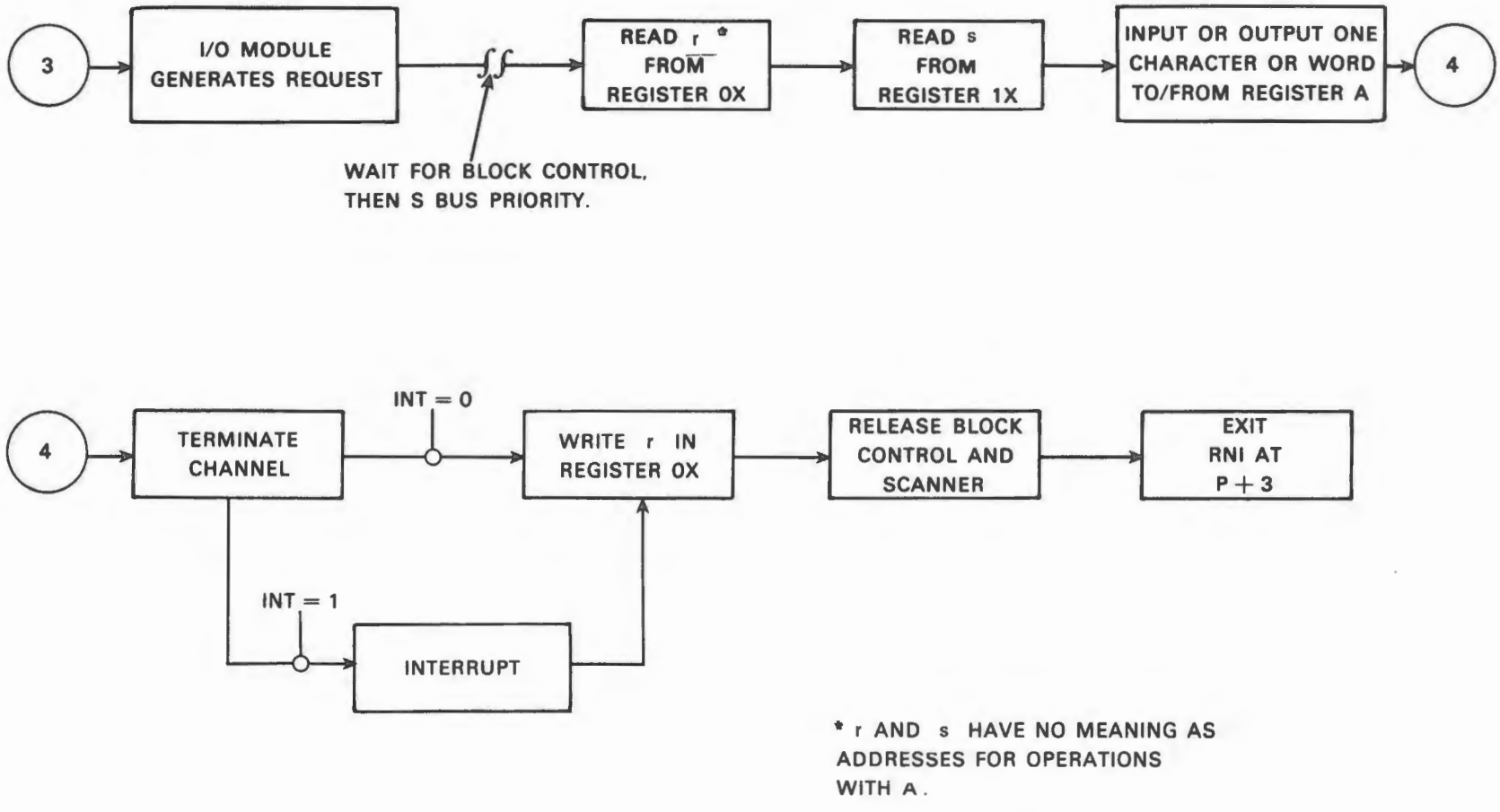
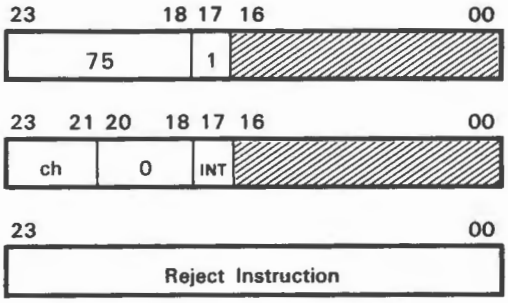


Figure 7-13. 74 I/O Operation with A

OTAC Output, Character from A



(Approximate execution time: 7.0 μsec.)

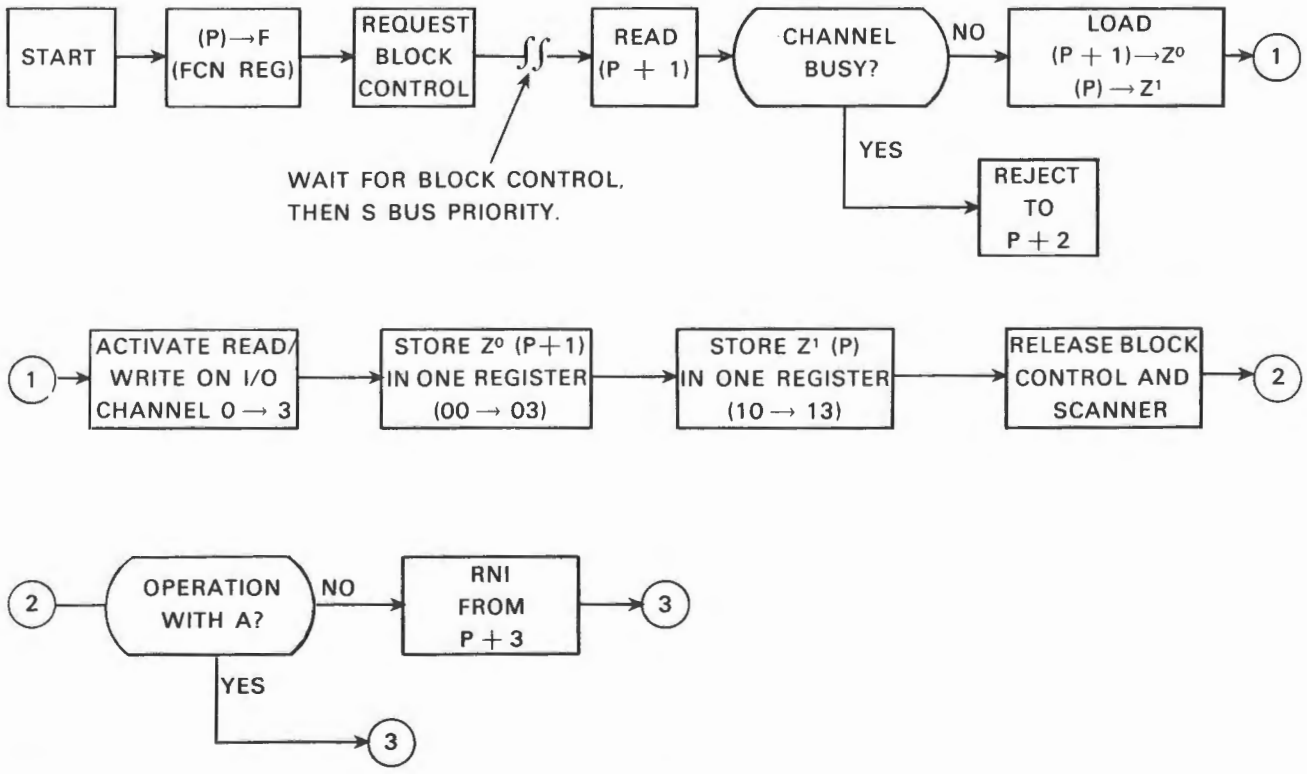
ch = I/O channel designator, 0-3
INT = "1" for interrupt upon completion

Instruction Description: **This instruction transfers a character from the lower 6 bits of A to an external equipment. The original contents of A are retained.**

Comments: **Bits 00-16 at P and P + 1 should be loaded with zeros.**

I/O OPERATION WITH A

INSTRUCTION SEQUENCE



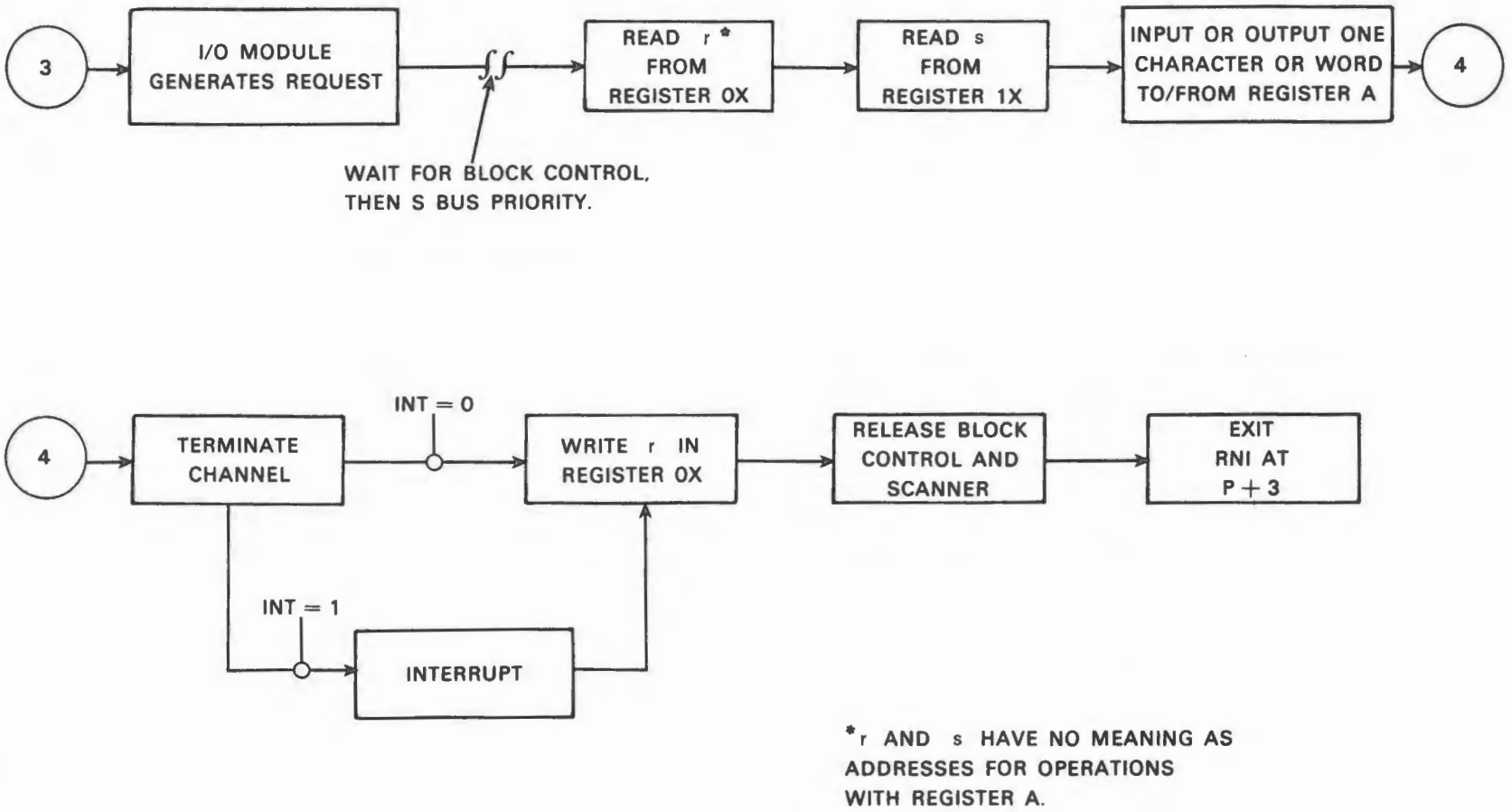
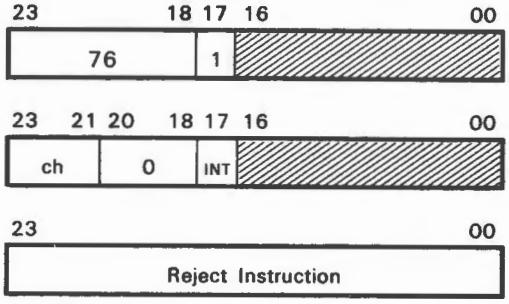


Figure 7-14. 75 I/O Operation with A

OTAW Output, Word from A



(Approximate execution time: 7.0 μsec.)

ch = I/O channel designator, 0-3
INT = "1" for interrupt upon completion.

Instruction Description: This instruction transfers the lower 12 bits of A, or (A) to an external equipment, depending upon the type of I/O channel (3106 or 3107) that is used. The original (A) remains unchanged.

Comments: Bits 00-16 at P and P + 1 should be loaded with zeros.

NOTE

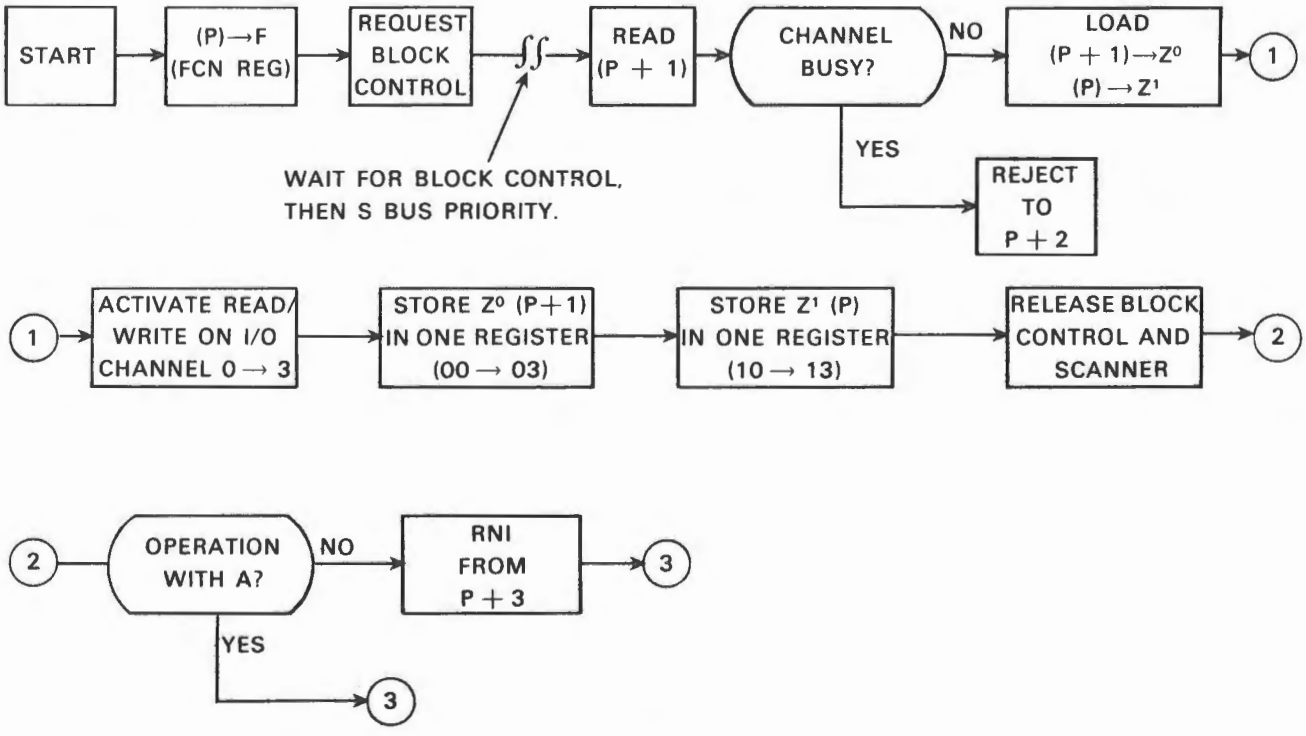
Bits 18, 19, and 20 are all zeros when a 3106 data channel is used. If the operation with A involves the use of a 3107, these bits take on the following significance:

Bit 20 = always a "0".

Bit 19 = If bit 18 = "1", the state of bit 19 is of no consequence.
If bit 18 = "0", a "1" in bit 19 signifies backward operation.
A "0" in bit 19 signifies a forward operation.

I/O OPERATION WITH A

INSTRUCTION SEQUENCE



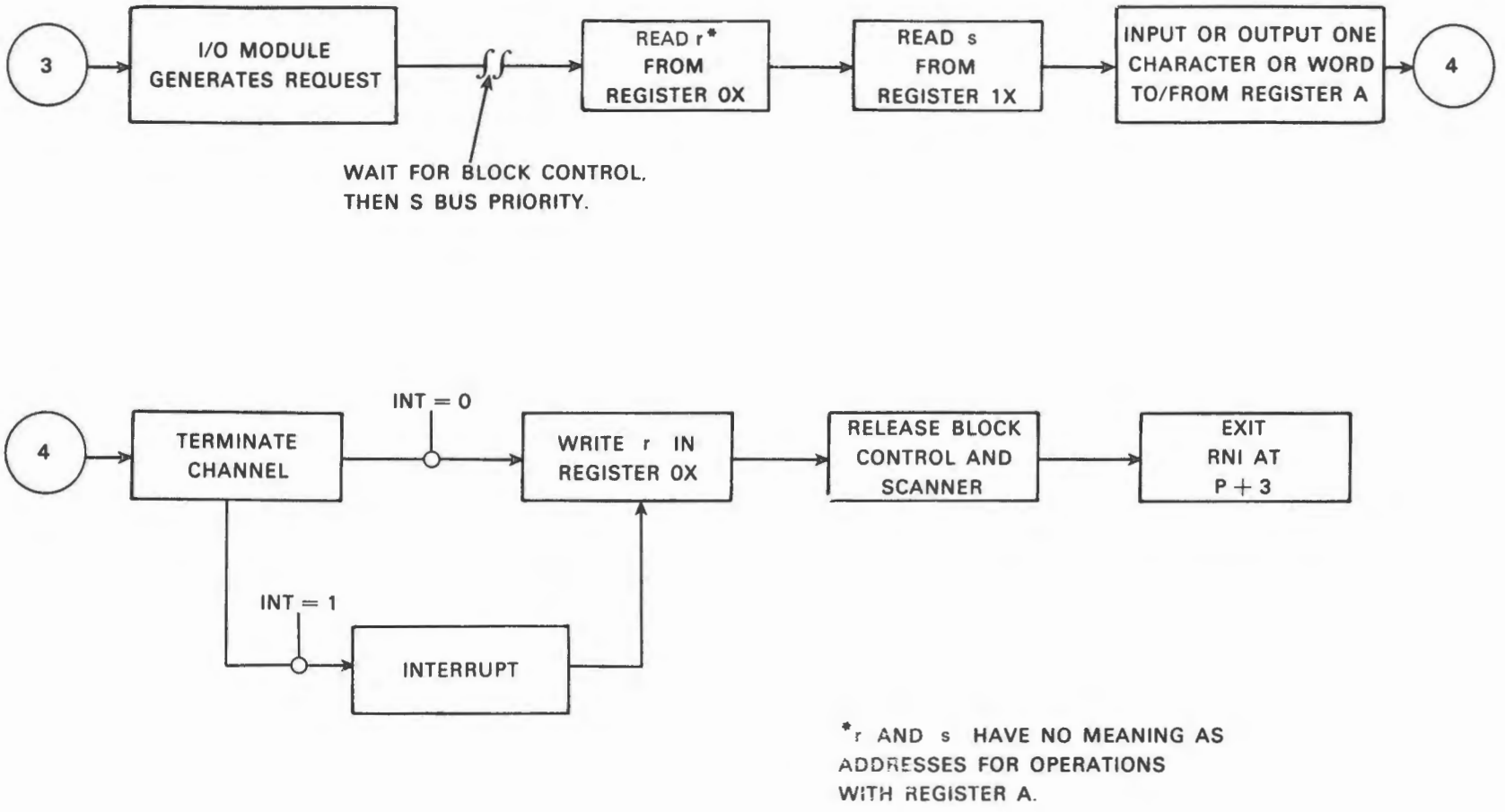


Figure 7-15. 76 I/O Operation with A

Section 8
SOFTWARE SYSTEMS

GENERAL DESCRIPTION

This chapter presents a synopsis of the major software systems applicable to a 3100 Computer System. The software information contained in this chapter is also valid for 3200 and 3300 Computer Systems.

Reference manuals are available for each of the systems described in this chapter and should be consulted for detailed information. Copies of these manuals and others as they become available may be obtained by corresponding with the nearest Control Data sales office listed on the back cover of this manual.

3100, 3200, 3300 SCOPE

SCOPE is the operation system for the CONTROL DATA 3100, 3200, 3300 Computers. Modular in structure, the system provides efficient job processing while minimizing its own memory and time requirements. Programming with the operating system is simplified by the use of control cards which are included with program decks. Among the functions performed by SCOPE are the following:

Job Processing

- Processes stacked or single jobs
- Controls I/O and interrupt requests
- Monitors compilations and assemblies
- Loads and links object subprograms
- Stores accounting information
- Initiates recovery dumps
- Prepares overlay tapes

Equipment Assignments

- Logical unit references
- Physical unit assignment at run time
- Drivers for all standard peripheral equipment
- System units which facilitate job processing and minimize monitor programming

Debugging Aids

- Extensive diagnostics
- Octal corrections
- Snapshot dumps
- Recovery dumps

Library Preparation and Editing

- Prepare a new library
- Edit an existing library
- List the contents of a library

3100, 3200, 3300 COMPASS

COMPASS is the comprehensive assembly system for the 3100, 3200, 3300 Computers. Operating under 3100, 3200, 3300 SCOPE, it assembles relocatable machine language programs. The program may consist of subprograms, each of which may be independently assembled. COMPASS source language includes the following features:

Operation codes	Machine operations are written as one or more mnemonic or octal subfields.
Addressing	Expressions, used as addresses, may represent either word or character locations. Expressions consist of symbols, constants, and special characters connected by + and −.
Data storage	A data area, shared by subprograms, may be specified and loaded with data in the source program.
Common storage	A common area may be designated to facilitate communication among subprograms.
Data definitions	Constants may be defined as octal, decimal, double-precision, integer or floating-point numbers; BCD words, BCD characters; or as strings of bits.
Library access	Library routines may be called by reference to their entry points or by inclusion of macros in the source program (data processing macros, input/output macros).
Listing control	The format of the assembly listing may be controlled by pseudo instructions.
Diagnostics	Diagnostics for source program errors are included with the output listing.
Macro instructions	Macros may be defined in the source program or entered into the library; the sequence of instructions will be inserted whenever the macro name appears in the operation field.

The Assembler

The COMPASS assembly program converts programs written in COMPASS source language into a form suitable for execution under the 3100, 3200, 3300 operating systems. Source program input may be on punched cards or in the form of card images on magnetic or paper tape. The output from the assembler includes an assembly listing and a relocatable binary object program on punched cards or magnetic tape.

Equipment Configuration

The assembly system, which is stored on the SCOPE library tape, is designed to operate on a computer with a minimum of 8,192 words of storage. In addition to the SCOPE library unit, the following input/output equipment is required:

Input unit: card reader, magnetic tape, or paper tape
Scratch unit: magnetic tape (may also be used for output)
Listable output unit: magnetic tape or printer
Object program output unit: magnetic tape or card punch

Program Structure

Source programs may be divided into subprograms which are assembled independently. All location symbols except COMMON and DATA symbols are local to the subprogram in which they appear, unless they are declared as external symbols. Locations which will be referenced by other subprograms are declared as entry points. For example, if subprogram IGOR references locations KIEV and MINSK in subprogram DEMETRI, KIEV and MINSK must be declared external symbols in subprogram IGOR and entry points in subprogram DEMETRI.

The links among subprograms are associated by the SCOPE loader. As each subprogram is loaded, all external symbols and entry points are entered into a symbol table. When an external symbol is found which matches an entry point already entered in the table, or an entry point is found which matches an external symbol, linkage between the two points is established.

If any external symbols are not matched with entry points after the last subprogram is loaded, the library tape is searched for routines with the names of unmatched symbols. If these routines are found, they are loaded and linked to the other subprograms. If unmatched external symbols remain, the job is terminated and an error message written by the system.

3100, 3200, 3300 DATA PROCESSING PACKAGE

The Data Processing Package is composed of Data Processing Routines and a General Purpose Input/Output System.

Data Processing Routines

The Data Processing Routines, called macros, are used in COMPASS assembly language programs to do particular data handling jobs; included are the following:

TRANSMIT	Transmits any string of up to 4,095 characters from one place in memory to another.
COMPARE	Compares fields located at A-address and B-address according to the data processing collating sequence. Fields may contain 1 to 4,095 characters. The fields are treated as equal, regardless of their specified lengths, by assuming blank fill to the right of the shorter field.
EDIT	Moves a numeric field to a receiving field with report editing.
MULTIPLY	Multiplies two BCD numbers and stores the result in a third.
DIVIDE	Divides one BCD number by another and stores the result in a third.

General Purpose Input/Output System

The General Purpose Input/Output System is a series of library routines which provide complete input/output control for data processing. These routines are used in COMPASS assembly programs, and they simplify programming while offering versatile data handling and optimum usage of internal storage space and processing time. Complete, partial or no buffering may be designated, depending upon the amount of storage the programmer has available; multi-file reels or multi-reel files may be read or written; fixed or variable length logical or physical records may be processed; and magnetic tape, paper tape, cards or printer may be used for input/output units. Both labeled and unlabeled tapes may be handled. The input/output macros perform the following functions:

OPEN	Opens an input or output file.
READ	Reads one logical record into the record area or a specified area in memory.
WRITE	Writes one logical record from the record area or a specified area in memory.
CLOSE	Closes a reel or file.

In addition to the input/output operations, the programmer also describes the files to be processed through use of macros.

FIELDESC	Defines logical records, buffers, logical units, recording density and rerun requirements.
LABELING	Describes file label and tape retention time (prevents accidental destruction of tapes).
VARIABLE	Indicates whether the size of a variable length record is determined by a record mark or a key field.
STOP OPEN	Allows user to let files share the same areas in storage. Defines multi-file reels.

The I/O System interprets each set of instructions, refers to the file description, and then initiates the requested operation; it controls buffering, transmission errors, and logical-physical record divisions.

3100, 3200, 3300 UTILITY

The Utility Package consists of a small control routine and a group of closed subroutines which, operating under control of the SCOPE operating system, will perform such functions as tape handling, copying of records from unit to unit, and record comparison of two files. The package is open-ended; subroutines may be added as desired.

3100, 3200, 3300 COBOL

COBOL is a programming system designed to facilitate the solution of business data processing problems. To use COBOL, the programmer describes the problem in a language resembling English; the COBOL processor translates this source language input into relocatable machine language for program execution.

THE COBOL language contains the elements of required COBOL as set forth by the official government manual describing COBOL-61, plus many of the features defined as elective COBOL.

A COBOL source program is specified in four divisions: IDENTIFICATION, ENVIRONMENT, DATA and PROCEDURE. The IDENTIFICATION division identifies the name, author, date, and so forth of the program. The ENVIRONMENT division defines the computer configuration required for both compilation and execution. The DATA division describes the format of the data files which the program is to process. The PROCEDURE division contains a sequence of statements which describe the processing to be performed.

The COBOL compiler is a three pass system. No object code is produced until the entire source program has been thoroughly analyzed. Whenever possible, in-line coding is produced. Depending on the needs of the program, the compiler provides an input/output system which allows variable length records, up to two buffer areas per file, multi-file reels, multi-reel files, and so forth. In general, the features of the COBOL input/output system correspond to those described for the Data Processing Package.

3100, 3200, 3300 FORTRAN

The 3100, 3200, 3300 FORTRAN system incorporates a problem-oriented language that facilitates simple algebraic solution of mathematical or scientific problems.

3100, 3200, 3300 FORTRAN programs are written as a sequence of statements, using familiar arithmetic operations and English expressions. Large programs may be written independently in sections, the sections tested, then executed together.

Statements are available to reserve areas of memory for variables and arrays. Strings of values may be loaded with the program for reference during the program execution. Equivalence statements allow the same areas of memory to be identified with different variables and arrays during the execution of a program.

Type statements specify the mode in which values are to be stored. The possible types include: REAL, INTEGER, and CHARACTER. The programmer may also declare a special mode, type OTHER, to handle information which does not conveniently conform to the standard modes.

Arithmetic expressions are indicated by arithmetic sign and algebraic names. For example, A+B−C means add A to B and subtract C. Logical and relational operators are available for use in expressions which may be true or false.

Statements are usually executed in sequence. However, control statements may be used to transfer to another part of the program. (The transfer may be specified as dependent on a test indicated by an expression in the transfer statement.)

Sets of statements which are to be executed several times with minor changes or increments may be written once with a statement to indicate how many times they are to be repeated, and if they are to be changed each time.

Input/output operations provide a means to read information into the machine from various sources and to record results on a selected output device. If buffered input/output operation is specified, other operations may continue while information is read in or out.

Facilities are also available to transfer a number of characters from one area of memory to another, and to test machine conditions through calls to 3100, 3200, 3300 FORTRAN library functions.

The 3100, 3200, 3300 FORTRAN compiler produces machine language programs which may be executed immediately or stored for execution at a later date.

GENERALIZED SORT/MERGE PROGRAM

The GENERALIZED SORT/MERGE PROGRAM organizes data on magnetic tape into one continuous predetermined order. SORT/MERGE operates under the SCOPE operating system. Control cards read from the standard input unit contain file descriptions and SORT/MERGE specifications.

SORT/MERGE orders fixed or variable length tape records, blocked or unblocked, written in either BCD or binary mode, according to a specified collating sequence. BCD and binary collating sequences are provided within SORT/MERGE, or the user may specify his own. The resultant output file may be merged with other presorted files in a final merge pass, or, if a number of presorted files exist, the merge phase only can be performed.

The SORT/MERGE program can transfer instruction execution to the user's prepared subroutines which in turn perform the following typical functions. Other subroutines not shown on this list may also be used:

- Edit acceptable records
- Reject records
- Check nonstandard labels
- Modify nonstandard labels
- Generate messages for the operator
- Write secondary output file (edit sorted records)
- Prepare summary file (summarize sorted records)
- Terminate the sort process

The SORT/MERGE checks standard header and trailer labels and provides rerun dumps. The SORT/MERGE contains an internal sort phase and a merge phase. The sort uses the tournament replacement technique which makes maximum use of available core storage and takes advantage of existing bias in the data. The method of merging, which is selected by the user, can be normal balanced or polyphase with either forward or backward reading.

3100, 3200, 3300 BASIC SYSTEM

Included in the 3100, 3200, 3300 BASIC system are:

- BASIC Assembler
- BASIC FORTRAN II
- BASIC Utility

BASIC Assembler

The BASIC Assembler language forms a subset of the 3100, 3200, 3300 COMPASS language. Although designed primarily for use on a 4K configuration, it can readily be used on larger systems. Object programs produced by the BASIC Assembler are loaded by the self-contained loader or can be loaded by SCOPE. Source language programs must be prepared as complete entities if they are to be loaded by the internal loader. As a result, facilities for referencing external storage areas (COMMON, DATA) and external program elements (ENTRY, EXT. Macros) are not used in BASIC Assembler language, nor are a few of the more complex pseudo instructions (VFD, IF). All other features of the language are similar: operating codes, addressing, data definitions, listing control, etc.

To assemble a BASIC Assembler program, the following configuration is required:

- Minimum of 4K words of storage
- Input unit: card reader, magnetic tape or paper tape (used for source language input, library, and BASIC Assembler)
- Listable output unit: printer, magnetic tape, paper tape, typewriter
- Object program output unit: card punch, magnetic tape, paper tape, typewriter (All output may be written on one tape unit if desired.)

BASIC FORTRAN II

BASIC FORTRAN II is a problem-oriented language that performs familiar mathematical operations in arithmetic expressions and replacement statements. The source language provides substantial power and flexibility through a variety of statements. BASIC FORTRAN II is compatible with other FORTRAN II systems.

BASIC Utility

This package is similar to the 3100, 3200, 3300 Utility but incorporates its own loader and input/output control routine.

CODING PROCEDURES

COMPASS subprograms are written on standard coding sheets. A subprogram consists of symbolic or octal machine instructions and pseudo instructions. Symbolic machine instructions are alphabetic mnemonics for each of the machine instructions. Pseudo instructions are COMPASS instructions used for the following operations:

- Subprogram identification and linkage
- Data definition (constants conversion)
- Data storage
- System calls
- Assembler control
- Output listing control
- Macro definition

INSTRUCTION FORMAT

A COMPASS instruction may contain location, operation code, address, comment, and identification fields.

Location Field

A symbol in the location field (LOCN) is placed in columns 1-8. A symbol identifies the address of an instruction or data item.

Location field symbols may be blank or consist of one to eight alphabetic or numeric characters; the first character must be alphabetic. Embedded blanks are illegal in location symbols. The following are examples of location symbols:

A
H3
ABCDEFGH
P1234567

A single * in column 1 of the location field signifies a line of comments.

Operation Code Field

The operation code field (OP) consists of any of the mnemonic or octal instruction codes with modifiers, or any macro or pseudo instructions. The field begins in column 10 and ends at the first blank column. If a modifier is used, a comma must separate the operation code from the modifier; no blank columns may intervene. A blank operation field or a blank in column 10 results in a machine word with zeros in the operation field.

Address Field

The address field begins before column 41 and after the blank which terminates the operation field, and ends at the first blank column. It is composed of one or more subfields, depending upon the instruction. Subfields, which are separated by commas on the coding form, specify the following quantities:

m or n	word address
r or s	character address
y	operand (15-bit)
z	operand (17-bit)
b or i	index register or interval quantity
c	character
v	register file location
ch	channel
x	function code or comparison mask
l	number of characters in a block

The interpretations of the address subfields for each set of instructions are described in Table 8-1.

An m,n,r,s,y or z subfield may contain:

- A location symbol
- The symbol ** which causes each bit in the subfield to be set to one
- The symbol * which causes the assembler to insert the relocatable address of that instruction in the address field
- An integer constant
- An arithmetic expression
- A literal

TABLE 8-1. INSTRUCTION INTERPRETATIONS

FIELD	Subfields	INSTRUCTION OPERATION CODES			
		00-70	71 Search	72 Move	73-77 I/O
	m, n	word address	—	—	first word address, last word address +1
	b	index register	—	—	—
	y or z	operand	—	—	—
	c	—	character	—	—
	r	character address	address of first character	first character address of source field	first character address
	s	—	address of last character ± 1	first character address of receiving field	last character address ± 1
	ch	—	—	—	channel
	x	—	—	—	I/O or interrupt code
	i	interval quantity	—	—	—
	l	—	—	field length	—

b SUBFIELD — The index field (b) specifies an index register 1-3, or a symbol or expression which results in one of these registers. Some instructions require a particular index register. If the b subfield is used with the octal operation codes, 0-7 may be used.

c SUBFIELD — The character field may contain any octal or decimal number, expression, or a symbol which is equivalent to a 6-bit binary number. Octal numbers must be suffixed with the letter B.

ch SUBFIELD — The channel field may contain one digit to designate an input/output channel, or a symbol equated to one of these digits, or an expression resulting in one of the digits.

x SUBFIELD — The code field may contain any of the interrupt or input/output codes or comparison mask. Decimal numbers, octal numbers suffixed with the letter B, symbols or expressions resulting in constants may be used.

v SUBFIELD — The register file subfield specifies a location which may be 00_8-77_8. Any legal coding which results in a value 00_8-77_8 may be used.

i SUBFIELD — In the MEQ and MTH instructions, this subfield specifies a decrement interval quantity of 1-8.

***l* SUBFIELD** — In the MOVE instruction, this subfield specifies the number of characters (1 to 128) to be moved.

Comments Field

Comments may be included with any instructions. A blank column must separate them from the last character in the address field, and they may extend to column 72. Comments have no effect upon compilation but are included on the assembly listing.

Identification Field

Columns 73-80 may be used for sequence numbers or for program identification. This field has no effect upon assembly.

PSEUDO-INSTRUCTIONS

Monitor Control

The following pseudo instructions provide communication between COMPASS subprograms and the monitor. Some are required in every subprogram; others are optional. Unless otherwise noted, each instruction may have a location field and an address field.

IDENT m — appears at the beginning of every COMPASS subprogram. The address field contains the name of the subprogram, which may be a maximum of eight alphanumeric characters, the first being alphabetic. A symbol in the location field is ignored and results in an error flag (L) on the listing.

END m — marks the end of every subprogram. When a program (consisting of one or more subprograms) is assembled for execution, one of the subprogram END cards must contain a location symbol in the address field to indicate the first instruction to be executed in the program. Only one END card can contain an address field symbol. A term in the location field is ignored.

FINIS — terminates an assembly operation. It is a signal to the assembler that no more programs are to be assembled. The FINIS card is placed after the last END card of the last subprogram in the source program.

Symbol Assignments

The pseudo instructions, EQU; EQU, C; ENTRY; and EXT define symbols as equal to other symbols or values, or identify symbols used to communicate with subprograms. Linkage between symbols in separate subprograms is provided by the monitor system. These pseudo instructions may appear anywhere between an IDENT and an END pseudo instruction.

EQU m — assigns the result of the expression in the address field to the symbol in the location field. The result is a 15-bit address.

The following forms are allowed:

```
symbol   EQU   symbol
symbol   EQU   constant (octal or decimal)
symbol   EQU   expression (address arithmetic)
```

Example:

```
OUT  EQU  JUMP+2
```

If JUMP is assembled to address 00100, OUT will be assigned the value 00102.

Numerical constants must follow the rules for symbolic instructions. Address arithmetic is permitted. A location field symbol may be equated to a decimal or octal constant.

EQU, C m — is similar to EQU, except that the result is a 17-bit address.

ENTRY m — defines location symbols which are referenced in other subprograms. These symbols, called entry points, must be placed in the address field of an ENTRY pseudo instruction. Any number of locations may be declared as entry points in the same ENTRY instruction. If two or more names appear in the address field, they must be separated by commas. No spaces (blanks) can appear within a string of symbols. The address field of the ENTRY pseudo instruction may be extended to column 72 and the location field must be blank. Only word-location symbols (15-bits) may be used.

Example:

```
ENTRY   SYM1,SYM2,SYM3
```

SYM1, SYM2, SYM3 can now be referenced by other subprograms.

EXT m — Symbols used by a subprogram which are defined in another subprogram are declared as external symbols by placing them in the address field of an EXT pseudo instruction. Only word-location symbols (15-bit) may be used. For example, to use the external symbols SYM1, SYM2, SYM3 in subprogram A, the following pseudo instruction would be written in subprogram A:

```
EXT  SYM1,SYM2,SYM3
```

These symbols must be declared as ENTRY points in some other subprogram or subprograms which are loaded for execution with subprogram A. The address field may be extended to column 72; symbols are separated by commas. No spaces (blanks) can appear in a string of symbols. The location field of an EXT must be blank.

Address arithmetic cannot be performed on external symbols.

Example:

```
        IDENT   CAIRO
        ENTRY   DEED, FFI
        EXT     ABE, DAVID
FFI     SJ1     **
          •
BEN     EQU     HAKIM
          •
DEED    LDA     ABE
          •
        RTJ     DAVID
          •
          •
        END
        END     FFI
        FINIS
```

Listing Control

The pseudo instructions which provide listing control for assembly listings are shown below. These instructions do not appear on the assembly listing and may be placed anywhere in a program.

SPACE — controls line spacing on an assembly listing. A decimal constant in the address field designates the number of spaces to be skipped before printing the next line. If the number of spaces to be skipped is greater than the number of lines remaining to be printed on a page, the line printer skips to the top of the next page. A symbol in the location field is ignored.

EJECT — causes the line printer to skip to the top of the next page when the assembled program is listed. A symbol in the location field is ignored.

REM — is used to insert program comments in an assembly listing. The address field can be extended to column 72. Any standard key punch character can be used in the comments. If the comments are to be written on more than one line, successive REM pseudo instructions must be used. A symbol in the location field is ignored.

NOLIST — causes the assembler to discontinue writing a listing of the program, starting with this instruction.

LIST — causes the assembler to resume listing the program. This instruction is used after a NOLIST instruction; it is not necessary to use it to obtain a complete listing of a program.

Macro Instructions

MACRO — defines the beginning of a sequence of instructions that are inserted by the assembler in the source program whenever the location symbol of the MACRO instruction appears in an operation field. The end of the sequence of instruction is marked by an ENDM pseudo instruction. For example, if the sequence

```
HOPE    MACRO   (PA, MA)
        LDA     PA
        INA     24B
        STA     MA
        ENDM
```

were defined and the following instructions appeared in the same program

```
        STA     GARAGE
        HOPE    (DW21, D6)
        LDA     FARM
```

the assembled output would be

```
        STA     GARAGE
        LDA     DW21
        INA     24B
        STA     D6
        LDA     FARM
```

ENDM — defines the end of a MACRO sequence.

LIBM — names library macros.

NAME ($p_1, \ldots, p_n$) — is used to reference macros.

The parameters $p_1, \ldots, p_n$ are used by the routine, and NAME is a macro name.

Data Storage Assignments

The following pseudo instructions reserve storage areas for blocks of data. BSS may be used to reserve storage blocks within the subprogram in which it appears. If these storage areas are to be referenced by other subprograms, the name assigned to the block is declared as an entry point in the program containing the block, and as an external symbol in the program referencing the block. Only word location symbols may be used. COMMON identifies storage areas to be referenced by more than one subprogram. DATA specifies special areas which may be preloaded with data; EXT and ENTRY are not needed to reference COMMON or DATA areas. Address arithmetic may be used, but all symbols must have been defined before the instruction is encountered.

BSS m — reserves a storage area of length m in a subprogram on a common or data storage area. The address field may contain any expression which results in a constant. The resultant constant specifies the number of words to be used. The address field of the first word of the reserved area is assigned the location field term of the BSS instruction. Other words or characters in the area may be referenced by addressing arithmetic or by indexing.

BSS, C m — reserves a character storage area of length m in a subprogram. The address field is similar to the address field of BSS pseudo instruction. However, the resultant constant specifies the number of character positions to be reserved.

COMMON — assigns location terms following it to a common storage block until a DATA or PRG pseudo instruction is encountered. EQU, EXT, ENTRY, IFT, IFN, IFF, IFZ, END, ORGR, BSS and BSS,C are the only pseudo instructions which may follow a COMMON pseudo instruction.*

*Occurrence of any other machine or data definition command causes the command and its successors to be assembled into the subprogram area.

```
        IDENT     BURKE
        COMMON
A       BSS       20
B       BSS       10
C       BSS        6
          •
          •
        END
        IDENT     SPINOZA
        COMMON
MARKET  BSS        5
STREET  BSS       13
SINGER  BSS        4
        END
```

Location and address fields of a COMMON pseudo instruction should be blank.

COMMON may not be preset with data.

During execution, one area in storage is assigned as COMMON. All COMMON may be filled repeatedly during execution. A storage location assigned to the nth word in COMMON in subprogram 1 is the same location assigned to the nth word in COMMON in subprogram 2.

If the two subprograms in the above example were loaded together, the memory assignments would be:

Example:

Locations in memory relative to the beginning of common	Name in subprogram BURKE	Name in subprogram SPINOZA
0-4	A ⟶ A+4	MARKET ⟶ MARKET+4
5-17	A+5 ⟶ A+17	STREET ⟶ STREET+12
18-19	A+18 ⟶ A+19	SINGER ⟶ SINGER+1
20-21	B ⟶ B+1	SINGER+2 ⟶ SINGER+3
22-29	B+2 ⟶ B+9	_____
30-35	C ⟶ C+5	_____

PRG — terminates the definition of a COMMON or DATA area.

DATA — assigns all location symbols following it to a data block until a COMMON or PRG pseudo instruction is encountered. Data described by OCT; BCD; BCD,C; DEC; DECD and VFD pseudo instructions may be assembled into a DATA block. Areas may be reserved within a DATA block by the BSS and BSS,C pseudo instructions. The following is an example of a DATA pseudo instruction coded within a subprogram:

Example:

```
          •
          •
          •
          LDA       APRESMOI
          DATA
CONS      OCT       10, 11, 12, 13
          PRG       *
          STA       LEDELUGE
```

A data area named CONS is reserved and the octal constants 10, 11, 12, and 13 are loaded into the four words in this area. In the source program, STA LEDELUGE would appear in the next location after LDA APRESMOI.

Constants

Octal, decimal, and BCD constants may be inserted in a COMPASS program by using the pseudo instructions listed below. Location terms may be used and the address field may extend to column 72, if necessary.

OCT $m_1,m_2,\ldots,m_n$ — inserts octal constants into consecutive machine words. A location term is optional; if present, it will be assigned to the first word. The address field consists of one or more consecutive subterms, separated by commas. Each subterm may consist of a sign (+ or − or none), followed by up to eight octal digits. Each constant is assigned to a separate word. If a location term is present, it is assigned to the first word. If less than eight digits are specified, the constant is right-justified in the word and leading zeros are inserted.

DEC $m_1,m_2,\ldots,m_n$ — inserts 24-bit decimal integer constants in consecutive machine words. The D and B scaling is identical to the DECD scaling, but only positive integer values less than 2^{33} may be used. If a location term is present, it is assigned to the first constant.

DECD $M_1,m_2,\ldots,m_n$ — converts decimal constants to equivalent 48-bit binary values and stores them in consecutive groups of two machine words. Each constant may be written in either fixed or floating point format.

The decimal numbers to be converted are written in the address field of the DECD instruction as follows:

Floating Point Constant format consists of a signed or unsigned decimal integer of 14 digits. It is identified as a floating point constant by a decimal point which may appear anywhere within the digital string. A decimal scale factor indicated by D $\pm$ d is permitted. The result after scaling must not exceed the capacity of the hardware (approximately $10^{\pm 308}$).

Fixed Point Constant format is similar to that of the DEC single precision constants. Up to 14 decimal digits may be specified, expressing a value the magnitude of which is less than 2^{47}. Decimal and binary (B$\pm$b) scale factors may be used. Low order bits are not lost; the signed 48-bit binary result is stored in two consecutive computer words.

No spaces may occur within a number, including its associated scale factors, since a space indicates the end of the constant. Plus signs may be omitted. Any number of constants may appear in a DECD instruction. Successive constants are separated by commas.

Examples:

LOCN	Op	Address Field	Comments
CONST A	DECD	−12345.	FLOATING PT CONST
CONST B	DECD	+12345	FIXED PT CONST
CONST C	DECD	−12345.D+5	FLOATING PT CONST, DECSCALE
CONST D	DECD	12345D−3	FIXED PT CONST, DECSCALE
CONST E	DECD	+12345B+8	FIXED PT CONST, BINSCALE

BCD $n,c_1c_2,\ldots,c_{4n}$ — inserts binary-coded decimal characters into consecutive words. If a location term is present, it is assigned to the first word. The address field consists of a single digit n, which specifies the number of four-character words needed to store the BCD constant, followed by a comma and the BCD characters. The next 4n character positions after the comma are stored. Any character string which terminates before column 73 may be used; n is restricted accordingly.

BCD,C $n,c_1c_2,\ldots,c_n$ — places n characters in the next available n character positions in memory. If the previous instruction were also a BCD,C instruction, the next character position is defined as the one which follows the last position used by the previous instruction. If a location symbol is used, it is assigned to the first character position in this field. If the previous instruction were not a BCD,C instruction, the next character position would be the first character position (0) of the next available word. Any character string which terminates before column 73 may be used; n is restricted accordingly.

VFD $m_1n_1/v_1,m_2n_2/v_2\ldots,m_pn_p/v_p$ — assigns data in continuous strings of bits rather than in word units. Octal numbers, character codes, program locations and arithmetic values may be assigned consecutively in memory, regardless of word breaks. The address field consists of one or more data fields.

In each data field m specifies the mode of the data, n the number of bits allotted, and v the value. Four modes are allowed:

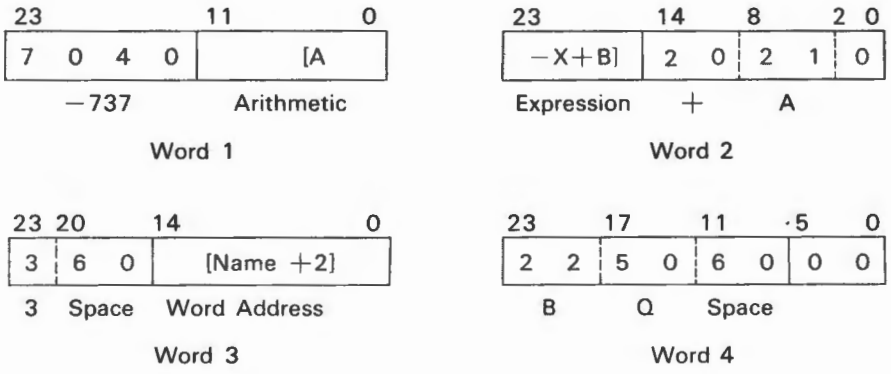
- **O** Octal number. If it is preceded by a minus sign, the one's complement form is stored.
- **H** Hollerith character code. The field length must be a multiple of six. Any printable character may appear in the v field except blanks or commas. Either a space or comma immediately succeeds the last character.
- **A** Arithmetic expression or decimal constant. The v field consists of an expression formed according to the rules for address field arithmetic, with the following restrictions:
  1. n must be ≤ 24 and $|v| \leq 2^{n-1}-1$ unless a relocatable expression is used, in which case, $n = 15$.
  2. When a relocatable expression is used, it must be placed in the correct position in the address portion of a word to insure that it will be relocated by the loader.
- **C** Character expression. The rules governing the A-field apply, except that $n = 17$ for a relocatable expression.

The VFD address field is terminated by the first blank column.

Example:

```
VFD 012/-737,A21/A-X+B,H24/+A3 ,A15/NAME
+2,H18/BQ.
```

A, X, and B are nonrelocatable symbols. Four words are generated, with the data placed as follows:



Additional Pseudo Instructions

Additional lines of coding may be generated by the following pseudo instructions:

IFZ m,n — n succeeding lines of coding are assembled if m is zero. The expression n must result in a positive numerical integer, and m may be a symbol, an address arithmetic symbol, or a literal. If m is non-zero, n succeeding lines of coding will be bypassed by the assembler.

IFN m,n — n succeeding lines of coding will be assembled if m is non-zero; the expression n must result in a positive numerical integer, and m may be a symbol, address arithmetic symbol or literal. If m is zero, n succeeding lines of coding will be bypassed by the assembler.

The pseudo instructions, IFT and IFN may be used within the range of a MACRO definition only.

IFT m,p,n, — n succeeding lines of coding will be generated if character string m equals character string p. The expression n must result in a positive numerical integer, and m and p may be a formal parameter or a literal. If m $\neq$ p, n succeeding lines of coding will be bypassed.

IFF m,p,n — n succeeding lines of coding will be generated if m $\neq$ p. The expression n must result in a positive integer, and m and p may be a formal parameter or a literal. If m = p, n succeeding lines of coding will be bypassed.

ORGR m — the value in the address field will be assembled as the beginning location for subsequent instructions. The value may be in program, data area or common area mode. The occurrence of a mode change pseudo operation, COMMON, DATA or PRG, terminates ORGR and subsequent instructions are assembled in the new mode.

NOP — No operation. An ENI y, O instruction is inserted.

TITLE — the information beginning in the address field is printed at the head of each page of the output listing which follows. The first page of listing may be titled by presenting the TITLE card immediately following the IDENT card.

ASSEMBLY LISTING FORMAT

An assembly listing contains the source program instructions and the corresponding octal machine instructions. The addresses assigned to each subprogram are relative addresses only. Absolute addresses are assigned when the program is loaded by the monitor loader. All common blocks are assigned consecutively, starting at relative location 00000. Preceding the body of the subprogram are summaries of undefined symbols, doubly defined symbols, external names, entry point names, subprogram length, common length and data length. References to external symbols are strung together by the assembler. The monitor loader assigns the proper absolute addresses.

The address of each instruction word is the left-most field for each instruction in the assembled listing. (Error codes appear to the left of this field.) External address field symbols are indicated by an X immediately to the left of the octal address field of each instruction. P indicates Program Relocatable, and C indicates Common. Subsequent fields from left to right on the listing are an 8-digit location contents field, a 2-digit operation code, a 1-digit b-subfield, a 5-digit address, and a 1-digit character position. The remaining fields correspond to those in the symbolic source program.

Listing format:

location	location contents	op	b	addr	char pos	source line	
5 or 6 digits	8	2	1	5	1	80	
05264	55300000	55	0	00000	3	EAQ	
05265	40003361	40	0	P03361	0	STA	HOLDAB
05266	27000173	27	0	P00173	0	LDL	CONTABLE+4
05267	40003362	40	0	P03362	0	STA	HOLDAB+1
05270	14600000	14	1	00000	2	ENA	0
05271	40003350	40	0	P03350	0	STA	SIMA
05272	25003300	25	0	P03300	0	LDAQ	TEMP6
05273	45003357	45	0	P03357	0	STAQ	HOLDADST
05274	77300400	77	0	00400	3	INS	400B
05275	01005301	01	0	P05301	0	UJP	*+4
05276	20003325	20	0	P03325	0	LDA	EXPFLTFG
05277	03105306	03	0	P05306	1	AZJ,NE	ADSBR5A-3
05300	01005311	01	0	P05311	0	UJP	ADSBR5A
05301	20003325	20	0	P03325	0	LDA	EXPFLTFG
05302	03105314	03	0	P05314	1	AZJ,NE	ADSBR5B
05303	14600001	14	1	00001	2	ENA	01
05304	40003341	40	0	P03341	0	STA	HOLDSH

ERROR CODES

The following error codes may appear as the left-most field on an assembled listing. If multiple errors are detected, multiple error codes are produced.

Code

- **A** Illegal character or expression in the address field.
- **D** Same symbol used in more than one location field term. Only the first symbol is recognized; the remainder are ignored. A list of doubly defined symbols appears on the assembled listing.
- **F** Symbol table is full. No more location field symbols will be recognized. Also designates overflow of MACRO parameter table.
- **O** Illegal operation code. Zeros are substituted for the operation code.
- **U** Undefined symbol. A list of undefined symbols will appear on the output listing.
- **C** An attempt was made to preset COMMON. The instructions are processed as if PRG was encountered.
- **L** A symbol appears in the location field when not permitted, a symbol is missing in the location field when one is required, or an illegal location symbol appears.
- **M** A modifier appears in the location field when not permitted, a modifier is missing in the operation field when one is required, or an illegal modifier appears in the operation field.
- **T** A character address symbol was used in an address subfield requiring a word symbol; significant bits are lost.

TABLE 8-2. COMPASS CODING FORM DESCRIPTION

FIELD	COLUMNS
Location	Use columns 1-8. Column 9 is always blank.
Operation	Begins in column 10 and continues until the first blank column.
Address	Address may begin after the column terminating the operation field; however, it must begin before column 41. The address field terminates when the first blank column or column 73 is reached.
Comments or Remarks	Comments or remarks are written between the end of the address field and column 73.
Identification or Sequence Number	Columns 73-80 are treated as comment by COMPASS.

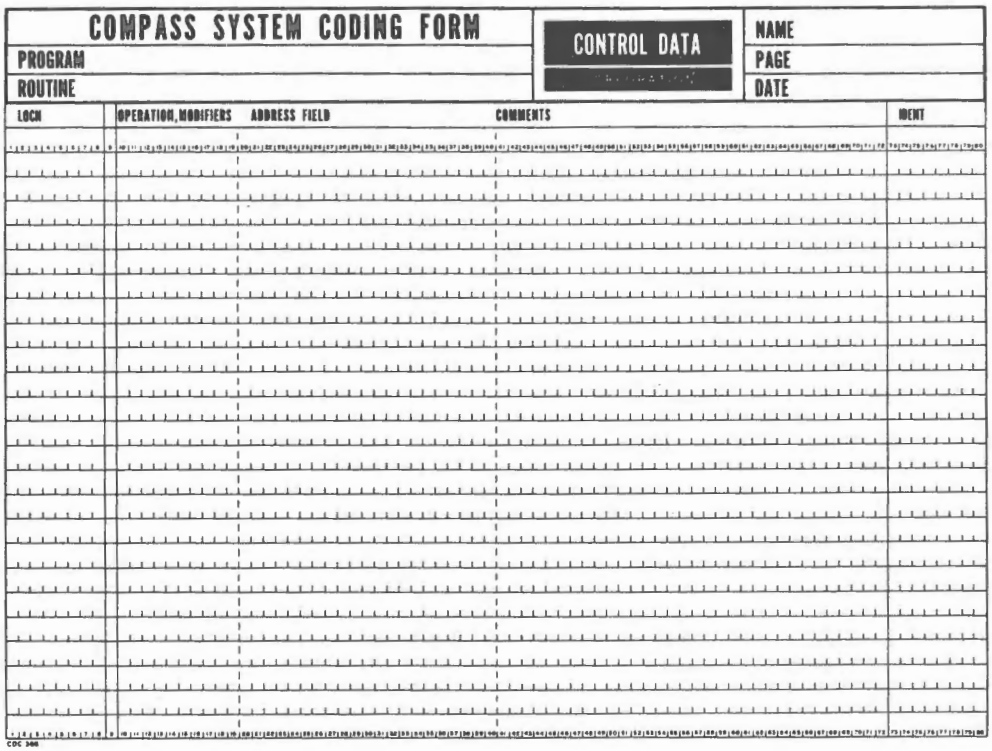


Figure 8-1. COMPASS Coding Form

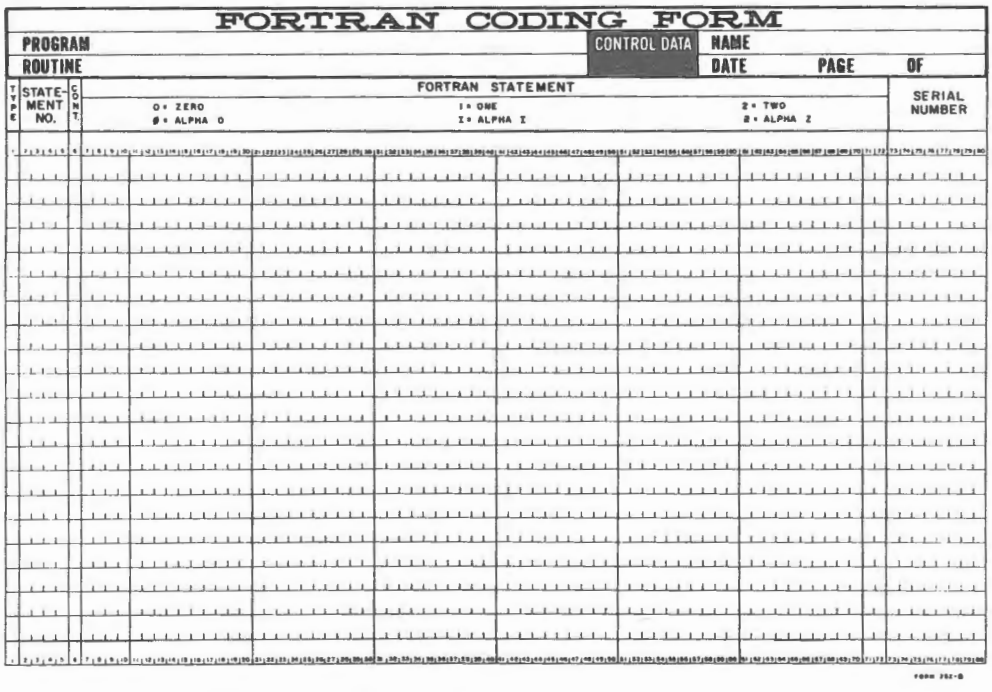


Figure 8-2. FORTRAN Coding Form

Appendix A

CONTROL DATA 3100, 3200, 3300 Computer Systems Character Set

Internal BCD Codes	External BCD Codes	Console Typewriter Characters (Uses Internal BCD Only)	Magnetic Tape Unit Characters	Punched Card Codes
00	12	0 (zero)	0 (zero)	0
01	01	1	1	1
02	02	2	2	2
03	03	3	3	3
04	04	4	4	4
05	05	5	5	5
06	06	6	6	6
07	07	7	7	7
10	10	8	8	8
11	11	9	9	9
12	(illegal)	±	---	2,8
13	13	=	#	3,8
14	14	"	@	4,8
15	15	:	---	5,8
16	16	;	---	6,8
17	17	?	(file mark)	7,8
20	60	+	&	12
21	61	A	A	12,1
22	62	B	B	12,2
23	63	C	C	12,3
24	64	D	D	12,4
25	65	E	E	12,5
26	66	F	F	12,6
27	67	G	G	12,7
30	70	H	H	12,8
31	71	I	I	12,9
32	72	(Shift to lower case)	+0	12,0
33	73	.	.	12,3,8
34	74	)	¤	12,4,8
35	75	[	---	12,5,8
36	76	@	---	12,6,8
37	77	!	---	12,7,8

(Cont.)

Internal BCD Codes	External BCD Codes	Console Typewriter Characters (Uses Internal BCD Only)	Magnetic Tape Unit Characters	Punched Card Codes
40	40	— (minus)	— (minus)	11
41	41	J	J	11,1
42	42	K	K	11,2
43	43	L	L	11,3
44	44	M	M	11,4
45	45	N	N	11,5
46	46	O	O	11,6
47	47	P	P	11,7
50	50	Q	Q	11,8
51	51	R	R	11,9
52	52	° (degree)	-0	11,0
53	53	$	$	11,3,8
54	54	*	*	11,4,8
55	55	#	---	11,5,8
56	56	%	---	11,6,8
57	57	(Shift to upper case)	---	11,7,8
60	20	(space)	(blank)	(blank)
61	21	/	/	0,1
62	22	S	S	0,2
63	23	T	T	0,3
64	24	U	U	0,4
65	25	V	V	0,5
66	26	W	W	0,6
67	27	X	X	0,7
70	30	Y	Y	0,8
71	31	Z	Z	0,9
72	32	&	---	0,2,8
73	33	(comma)	(comma)	0,3,8
74	34	(	%	0,4,8
75	35	(tab)	---	0,5,8
76	36	(backspace)	---	0,6,8
77	37	(carriage return)	---	0,7,8

Appendix B

Supplementary Arithmetic Information

Appendix B
SUPPLEMENTARY ARITHMETIC INFORMATION

NUMBER SYSTEMS

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

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

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

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:

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

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:

...	2^5	2^4	2^3	2^2	2^1	2^0	
	32	16	8	4	2	1	Binary point

The binary number 0 1 1 0 1 0 represents:

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

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

	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5} ...
Binary Point	½	¼	⅛	1/16	1/32

The binary number 0.10 110 may be represented as:

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

OCTAL NUMBER SYSTEM

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

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

The octal number 513_8 represents:

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

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

8^{-1}	8^{-2}	8^{-3}	8^{-4}	...
1/8	1/64	1/512	1/4096	

The octal number 0.4520 represents:

$$\begin{aligned} 4x8^{-1} &= 4x1/8 = 4/8 = 256/512 \\ +5x8^{-2} &= 5x1/64 = 5/64 = 40/512 \\ +2x8^{-3} &= 2x1/512 = 2/512 = \underline{2/512} \\ & \qquad\qquad 298/512 = 149/256_{10} \end{aligned}$$

ARITHMETIC

ADDITION AND SUBTRACTION

Binary numbers are added according to the following rules:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ with a carry of 1

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

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

Subtraction may be performed as an addition:

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

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

One's Complement

The computer performs all arithmetic and counting 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:

1111	
-1001	9
0110	(one's complement of 9)

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

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

$0000 \longrightarrow 00_2$	Positive (+) Zero
$1111 \longrightarrow 11_2$	Negative (−) Zero

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

- Both positive and negative zero are acceptable as arithmetic operands.
- If the result of an arithmetic operation is zero, it will be expressed as positive zero.

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

MULTIPLICATION

Binary multiplication proceeds according to the following rules:

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

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

Decimal example:

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

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

Binary example:

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

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

DIVISION

The following examples shows the familiar method of decimal division:

		14	quotient
divisor	13	185	dividend
		13	
		55	partial dividend
		52	
		3	remainder

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

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

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

CONVERSIONS

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

TABLE B-1. RECOMMENDED CONVERSION PROCEDURES (INTEGER AND FRACTIONAL)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Radix Arithmetic
Decimal to Octal	Radix Arithmetic
Binary to Octal	Substitution
Octal to Binary	Substitution

GENERAL RULES

$r_i > r_f$: use Radix Arithmetic, 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)

$$\begin{aligned} .44_8 &= 4(8^{-1}) + 4(8^{-2}) \\ &= 4/8 + 4/64 \\ &= 36/64_{10} \\ &= 9/16_{10} \end{aligned}$$

RADIX ARITHMETIC

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

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

To convert a fractional number from r_i to r_f:

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

EXAMPLE 1 Decimal to Binary (Integer)

```
45 ÷ 2 = 22 remainder 1; record          1
22 ÷ 2 = 11 remainder 0; record         0
11 ÷ 2 = 5  remainder 1; record        1
 5 ÷ 2 = 2  remainder 1; record       1
 2 ÷ 2 = 1  remainder 0; record      0
 1 ÷ 2 = 0  remainder 1; record     1
                                   ------
Thus: 45₁₀ = 101101₂               101101
```

EXAMPLE 2 Decimal to Binary (Fractional)

```
.25 x 2 = 0.5; record     0
.5  x 2 = 1.0; record      1
.0  x 2 = 0.0; record     0
                         ----
Thus: .25₁₀ = .010₂      .010
```

EXAMPLE 3 Decimal to Octal (Integer)

```
273 ÷ 8 = 34 remainder 1; record      1
 34 ÷ 8 =  4 remainder 2; record     2
  4 ÷ 8 =  0 remainder 4; record    4
                                   ----
                                    421
Thus: 273₁₀ = 421₈
```

EXAMPLE 4 Decimal to Octal (Fractional)

```
.55 x 8 = 4.4; record     4
.4  x 8 = 3.2; record      3
.2  x 8 = 1.6; record       1
--   --                     -
--   --                    -
                          ------
                          .431...
Thus: .55₁₀ = .431...₈
```

SUBSTITUTION

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

EXAMPLE 1 Binary to Octal

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

EXAMPLE 2 Octal to Binary

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

SUPPLEMENTARY INSTRUCTION INFORMATION

FIXED POINT ARITHMETIC

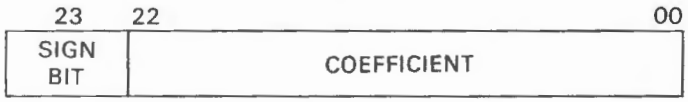
24-Bit Precision

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 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any fixed point number designates the sign of the coefficient (23 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 radix (binary) point is assumed to be immediately to the right of the lowest order bit (00).

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

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

48-Bit Precision (Double Precision)

The 48-bit Add, Subtract, Multiply and Divide instructions enable operands to be processed. The Multiply and Divide instructions utilize the E register and therefore are executed as trapped instructions if the applicable arithmetic option is not present in a system. Figure 7-4 in the Instruction Section illustrates the operand formats in 48-bit precision Multiply and Divide instructions.

FLOATING POINT ARITHMETIC

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.

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

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

NOTE

Refer to Figure 7-5 in the Instruction Section for the operand format and bit functions for specific floating point instructions.

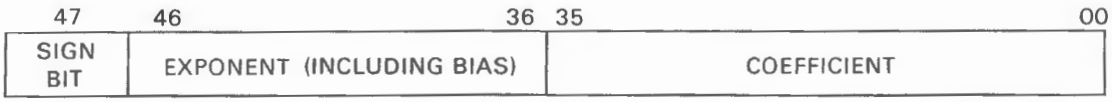
Coefficient

The coefficient consists of a 36-bit fraction in the 36 lower order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than ½ but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent

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

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



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

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

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

EXAMPLE 1

Number = +52

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

Number = +0.02

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

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

EXAMPLE 2

Number = $+52_8$

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

Number = $+0.02_8$

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

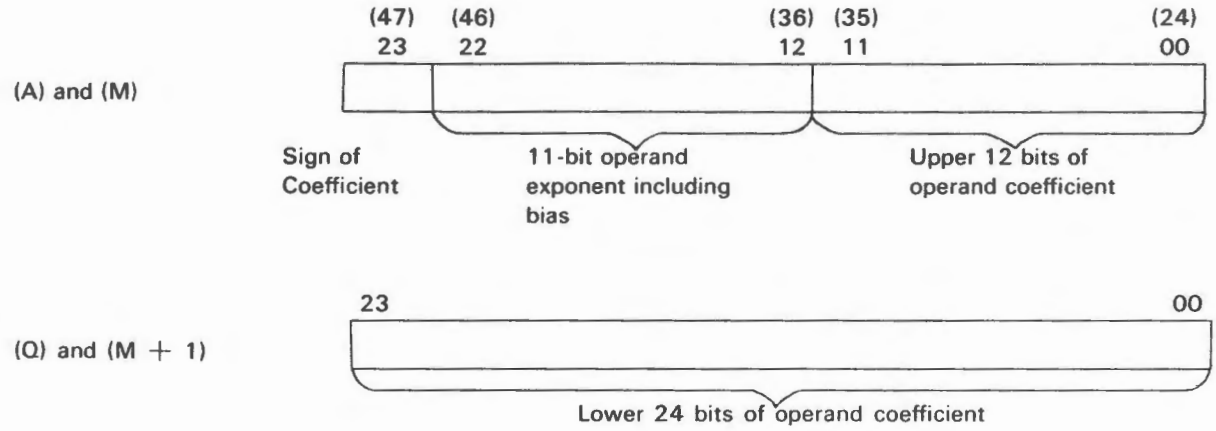
When bias is used with the exponent, floating point operation is more versatile since floating point operands can be compared with each other in the normal fixed point mode.

All floating point operations involve the A, Q, and E registers, plus two consecutive storage locations M and M + 1. The A and Q registers are treated as one 48-bit register. Indirect addressing and address modification are applicable to this whole group of instructions.

Operand Formats

The AQ register and the storage address contents have identical formats.

In both cases the maximum possible shift is 64 (77_8) bit positions. Since the coefficient consists of only 36 bits at the start, any shift greater than 36 positions will, of course, always result in an answer equal to the larger of the two original operands.



Exponents

The 3100, 3200, 3300 Computers use an 11-bit exponent that is biased by 2000_8 for floating point operations. The effective modulus of the exponent is $\pm$ 1777_8 or $\pm$ 1023_{10}.

Exponent Equalization

During floating point addition and subtraction, the exponents involved are equalized prior to the operation.

1. Addition – The coefficient of the algebraically smaller exponent is automatically shifted right in AQE until the exponents are equal. A maximum of 77_8 shifts may occur.
2. Subtraction – If AQ contains the algebraically smaller exponent, the coefficient in AQ is shifted right in AQE until the exponents are equal. If (M) and (M + 1) have the smaller exponent, the complement of the coefficient of (M) and (M+1) is shifted right in AQE until the exponents are equal or until a maximum of 77_8 shifts are performed.

Rounding

Rounding is an automatic floating point operation and is particularly necessary when floating point arithmetic operations yield coefficient answers in excess of 36 bits.

Although standard floating point format requires only a 36-bit coefficient, portions of the E register are used for extended coefficients. Refer to individual instruction descriptions for E register applications.

Rounding modifies the coefficient result of a floating point operation by adding or subtracting a "1" from the lowest bit position in Q without regard to the biased exponent. The coefficient of the answer in AQ passes through the adder with the rounding quantity before normalization. The conditions for rounding are classified according to arithmetic operation in Table B-2.

TABLE B-2. CONDITIONS PRIOR TO ROUNDING

Arithmetic OPERATION	Bit 23 of the A Register	Bit 47 of the E Register or (Ratio of Residue/Divisor for Divide Only)	Applicable Rounding
ADD or SUBTRACT	0*	0	No
	0*	1	Add "1"
	1*	0	Subtract "1"
	1*	1	No
	Comments: Rounding occurs as a result of inequality between the sign bits of AQ and E.		
MULTIPLY	0	0	No
	0	1	Add "1"
	1	0	Subtract "1"
	1	1	No
	Comments: A floating point multiplication yields a 76 bit coefficient. Comparison between the sign bits of AQ and E indicates that the lower 36 bits are equal to or greater than ½ of the lowest order bit in AQ.		
DIVIDE	0	≥ ½ (absolute)	Add "1"
	0	≤ ½ (absolute)	No
	1	≥ ½ (absolute)	Subtract "1"
	1	≤ ½ (absolute)	No
	Comments: Rounding occurs if the answer resulting from the final residue division is equal to or greater than ½.		

*Condition of bit 23 of the A register immediately after equalization. (Refer to Exponent Equalization on preceeding page).

Normalizing

Normalizing brings the above answer back to a fraction with a value between one-half and one with the binary point to the left of the 36th bit of the coefficient. In other words, the final normalized coefficient in AQ will range in value from 2^{36} to 2^{37}-1 including sign. Arithmetic control normalizes the answer by right or left shifting the coefficient the necessary number of places and adjusting the exponent. It does not shift the residue that is in E.

Faults

Three conditions are considered faults during the execution of floating point instructions:

1. Exponent overflow ($> + 1777_8$)
2. Exponent underflow ($< - 1777_8$)
3. Division by zero, by too small a number, or by a number that is not in floating point format.

These faults have several things in common:

1. They can be sensed by the INS (77.3) instruction
2. Sensing automatically clears them
3. The program should sense for these faults only after the floating point instructions have had sufficient time to go to completion
4. They may be used to cause an interrupt.

FIXED POINT/FLOATING POINT CONVERSIONS

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 $\frac{1}{2} \leq k < 1$.
3. Inspect the sign of the true exponent. If the sign is positive add 2000_8 (bias) to the true exponent of the normalized number. If the sign is negative, add the bias 1777_8 to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.
4. Assemble the number in floating point.
5. Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive, the assembled floating point number is the true representation.

EXAMPLE 1 Convert +4.0 to floating point

1. The number is expressed in octal.
2. Normalize. $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$
3. Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.
4. Assemble number in floating point format.
   Coefficient = 400 000 000 000$_8$
   Biased Exponent = 2003$_8$
   Assembled word = 2003 400 000 000 000$_8$
5. Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

EXAMPLE 2 Convert -4.0 to floating point

1. The number is expressed in octal.
2. Normalize. $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$
3. Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3
4. Assemble number in floating point format.
   Coefficient = 400 000 000 000$_8$
   Biased Exponent = 2003$_8$
   Assembled word = 2003 400 000 000 000$_8$
5. Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for
   -4.0 = 5774 377 777 777 777$_8$.

EXAMPLE 3 Convert 0.5_{10} to floating point

1. Convert to octal. $0.5_{10} = 0.4_8$
2. Normalize. $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$
3. Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 0.
4. Assemble number in floating point format.
   Coefficient = $400\ 000\ 000\ 000_8$
   Biased Exponent = 2000_8
   Assembled word = $2000\ 400\ 000\ 000\ 000_8$
5. Since the sign of the coefficient is positive, the floating point representation of $+0.5_{10}$ is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777_8 instead of 2000_8 because of the negative exponent. The 3100 and 3200, however, recognize -0 as + 0 and bias the exponent by 2000_8.

EXAMPLE 4 Convert 0.04_8 to floating point

1. The number is expressed in octal.
2. Normalize. $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$
3. Since the sign of the true exponent is negative, add 1777_8 (bias) to the true exponent. Biased exponent = $1777_8 + (-3) = 1774_8$
4. Assemble number in floating point format.
   Coefficient = $400\ 000\ 000\ 000_8$
   Biased Exponent = 1774_8
   Assembled word = $1774\ 400\ 000\ 000\ 000_8$
5. Since the sign of the coefficient is positive, the floating point representation of 0.04_8 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point Format

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

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

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

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

1. The sign of the coefficient is negative; therefore, complement the floating point number. Complement $= 2003\ 400\ 000\ 000\ 000_8$
2. The biased exponent (in complemented form) $> 2000_8$; therefore, subtract 2000_8 from the biased exponent to obtain the true exponent of the number. $2003 - 2000 = +3$
3. Coefficient $= 4000\ 000\ 000\ 000_8 = 0.100_2$. Move binary point to the right 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 400 000 000 000$_8$ to fixed octal

1. The floating point number is positive and remains uncomplemented.
2. The biased exponent $< 2000_8$; therefore, subtract 1777_8 from the biased exponent to obtain the true exponent of the number. $1774_8 - 1777_8 = -3$
3. Coefficient = $400\,000\,000\,000_8 = .100_2$. Move binary point to the left three places. Coefficient = $.000100_2$
4. The sign of the coefficient is positive because the floating point number was not complemented in step one.
5. Represent in fixed octal notation. $.000100_2 = .04_8$

BINARY CODED DECIMAL (BCD) ARITHMETIC

General

The Binary Coded Decimal (BCD) option expands the arithmetic capabilities of a 3100, 3200, or 3300 Computer by providing the necessary logic for loading, storing, shifting, adding and subtracting binary coded decimal characters. A standard 24-bit data word is comprised of four 6-bit BCD characters. The general format for a BCD word and the bit function within a typical character are illustrated in Figure B-1. Tables B-3 and B-4 define the significance of binary data within a character.
Figure B-2 depicts the E_D register and the other digits displayed on the 3200 Console.

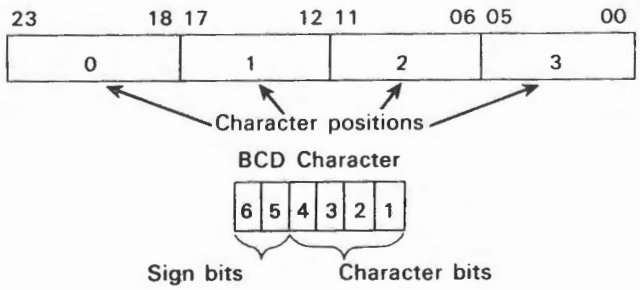


Figure B-1. BCD Word and Character Format

TABLE B-3. BCD SIGN BIT POSITIONS

Sign of BCD Character*	Relative Bit Positions 6	5
+	0	0
+	0	1
−	1	0
+	1	1

TABLE B-4. DECIMAL/BCD CHARACTER FORMAT

Decimal Number**	BCD Character Relative Bit Positions 4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

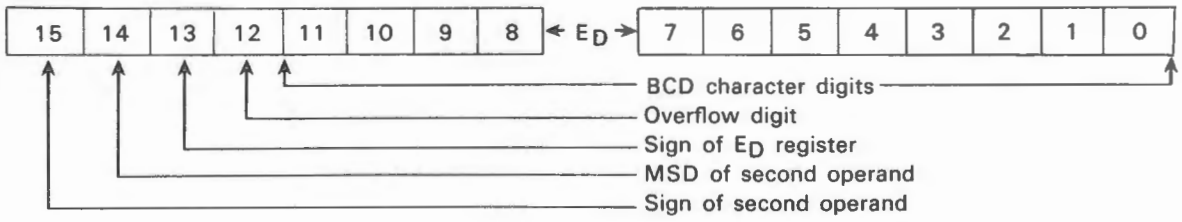


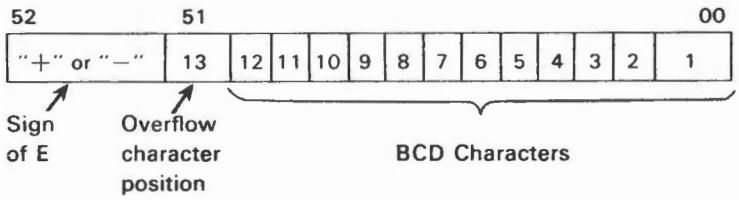
Figure B-2. E_D Register and Supplemental Digits

*The Lowest Significant Digit of a given BCD field contains the sign of the operand in relative bit positions 5 and 6. A fault is indicated if relative bits 5 and 6 in the remaining characters contain anything other than zeroes; however, the current instruction will continue to be executed.

**A fault is also indicated if an illegal character is sensed in bits 1 through 4 (1010, 1011, 1100, 1101, 1110 or 1111).

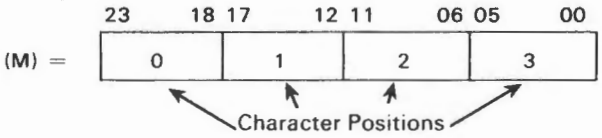
Formats: These instructions handle 4-bit BCD characters rather than whole 24-bit words. These characters are placed into the E_D register and storage in the following ways:

1) E_D Register



The 53-bit E_D register can hold 12 regular BCD characters plus one overflow character.

2) Storage



Each 24-bit storage word may be divided into four character positions of 6 bits each. The lower 4 bits of each position may hold any BCD character, 0-9; the upper 2 bits are reserved for the sign designator, one per field. For each field the sign accompanies the least significant character. 10xxxx specifies negative; any other combination, positive (refer to Table B-3). The upper 2 bits of all other characters in the field must equal zero. The most significant character precedes the least significant character of a field in storage.

Field Length: The field length is specified by the contents of the 4-bit D register. Any number 1-12 (0001-1100) is legal.*

Illegal Characters: By definition, any BCD characters other than 0-9 are illegal. Characters are tested for legality during:

1. Loading into E (LDE), and
2. Addition (ADE) and subtraction (SBE). If the translation of the lower four bits of a character exceeds 9, the value zero will be used for that character.

BCD Fault: The BCD fault will occur if:

1. A sign is present in any character position other than the least significant, or
2. An illegal character other than the lowest MB is sensed during the execution of LDE, ADE, SBE
3. The contents of D exceed 12 (will set only during a SET instruction).

*Although a fault will occur, D may equal 13 for storing 13 characters. The following sequence should be followed in storing 13 characters:

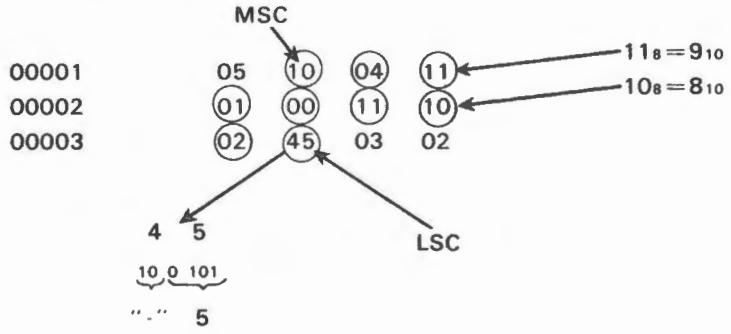
1) Set D (BCD fault will occur)
2) Sense for BCD fault (this clears the BCD Fault indicator)
3) Execute STE instruction.

If the BCD fault is disregarded and there is an attempt to load, add, or subtract 13 characters, only the lower 12 characters will be used. No additional fault will occur.

BCD Instruction Example

EXECUTED INSTRUCTIONS: 70 7 00011
64 0 00005

ADDRESSES: CONTENTS OF ADDRESS



NOTE

Only the LSC is analyzed for the sign of the field. A BCD fault occurs if anything other than zeros are in the upper two bits of the remaining characters.

ANALYSIS: 70 7 00011 instruction sets the field length register (D) with 11_8

64 0 00005 instruction specifies an LDE with successive BCD characters starting with the least significant character (LSC) at address R+(D-1) of 00005=address 0001, character position 1. 11_8 characters are loaded into E_D. The final contents of E_D are shown below.

E_D = -0000849109825

(A BCD character cannot be loaded into the 13th digit. A zero will always be entered here during a 64 instruction.)

Appendix C

Programming Reference Tables and Conversion Information

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625

DECIMAL/BINARY POSITION TABLE

Largest Decimal Integer	Decimal Digits Req'd*	Number of Binary Digits	Largest Decimal Fraction
1		1	.5
3		2	.75
7		3	.875
15	1	4	.937 5
31		5	.968 75
63		6	.984 375
127	2	7	.992 187 5
255		8	.996 093 75
511		9	.998 046 875
1 023	3	10	.999 023 437 5
2 047		11	.999 511 718 75
4 095		12	.999 755 859 375
8 191		13	.999 877 929 687 5
16 383	4	14	.999 938 964 843 75
32 767		15	.999 969 482 421 875
65 535		16	.999 984 741 210 937 5
131 071	5	17	.999 992 370 605 468 75
262 143		18	.999 996 185 302 734 375
524 287		19	.999 998 092 651 367 187 5
1 048 575	6	20	.999 999 046 325 683 593 75
2 097 151		21	.999 999 523 162 841 796 875
4 194 303		22	.999 999 761 581 420 898 437 5
8 388 607		23	.999 999 880 790 710 449 218 75
16 777 215	7	24	.999 999 940 395 355 244 609 375
33 554 431		25	.999 999 970 197 677 612 304 687 5
67 108 863		26	.999 999 985 098 838 806 152 343 75
134 217 727	8	27	.999 999 992 549 419 403 076 171 875
268 435 455		28	.999 999 996 274 709 701 538 085 937 5
536 870 911		29	.999 999 998 137 354 850 769 042 968 75
1 073 741 823	9	30	.999 999 999 068 677 425 384 521 484 375
2 147 483 647		31	.999 999 999 534 338 712 692 260 742 187 5
4 294 967 295		32	.999 999 999 767 169 356 346 130 371 093 75
8 589 934 591		33	.999 999 999 883 584 678 173 065 185 546 875
17 179 869 183	10	34	.999 999 999 941 792 339 086 532 592 773 437 5
34 359 738 367		35	.999 999 999 970 896 169 543 266 296 386 718 75
68 719 476 735		36	.999 999 999 985 448 034 771 633 148 193 359 375
137 438 953 471	11	37	.999 999 999 992 724 042 385 816 574 096 679 687 5
274 877 906 943		38	.999 999 999 996 362 021 192 908 287 048 339 843 75
549 755 813 887		39	.999 999 999 998 181 010 596 454 143 524 169 921 875
1 099 511 627 775	12	40	.999 999 999 999 090 505 298 227 071 762 084 960 937 5
2 199 023 255 551		41	.999 999 999 999 545 252 649 113 535 881 042 480 468 75
4 398 046 511 103		42	.999 999 999 999 772 626 324 556 767 940 521 240 234 375
8 796 093 022 207		43	.999 999 999 999 886 313 162 278 383 970 260 620 117 187 5
17 592 186 044 415	13	44	.999 999 999 999 943 156 581 139 191 985 130 310 058 593 75
35 184 372 088 831		45	.999 999 999 999 971 578 290 569 595 992 565 155 029 296 875
70 368 744 177 663		46	.999 999 999 999 985 789 145 284 797 996 282 577 514 648 437 5
140 737 488 355 327	14	47	.999 999 999 999 992 894 572 642 398 998 141 288 757 324 218 75

*Larger numbers within a digit group should be checked for exact number of decimal digits required.

Examples of use:

1. ***Q.*** **What is the largest decimal value that can be expressed by 36 binary digits?**
   A. **68,719,476,735.**
2. ***Q.*** **How many decimal digits will be required to express a 22-bit number?**
   A. **7 decimal digits.**

OCTAL ARITHMETIC MATRICES

ADDITION-SUBTRACTION

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

MULTIPLICATION-DIVISION

0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	10	12	14	16
3	3	6	11	14	17	22	25
4	4	10	14	20	24	30	34
5	5	12	17	24	31	36	43
6	6	14	22	30	36	44	52
7	7	16	25	34	43	52	61

CONSTANTS

π	=	3.14159 26535 89793 23846 26433 83279 50
$\sqrt{3}$	=	1.732 050 807 569
$\sqrt{10}$	=	3.162 277 660 1683
e	=	2.71828 18284 59045 23536
ln 2	=	0.69314 71805 599453
ln 10	=	2.30258 50929 94045 68402
$\log_{10} 2$	=	0.30102 99956 63981
$\log_{10} e$	=	0.43429 44819 03251 82765
$\log_{10} \log_{10} e$	=	9.63778 43113 00537−10
$\log_{10} \pi$	=	0.49714 98726 94133 85435
1 degree	=	0.01745 32925 19943 radians
1 radian	=	57.29577 95131 degrees
$\log_{10}(5)$	=	0.69897 00043 36019
7!	=	5040
8!	=	40320
9!	=	362,880
10!	=	3,628,800
11!	=	39,916,800
12!	=	479,001,600
13!	=	6,227,020,800
14!	=	87,178,291,200
15!	=	1,307,674,368,000
16!	=	20,922,789,888,000
$\frac{\pi}{180}$	=	0.01745 32925 19943 29576 92369 07684 9
$\left(\frac{\pi}{2}\right)^2$	=	2.4674 01100 27233 96
$\left(\frac{\pi}{2}\right)^3$	=	3.8757 84585 03747 74
$\left(\frac{\pi}{2}\right)^4$	=	6.0880 68189 62515 20
$\left(\frac{\pi}{2}\right)^5$	=	9.5631 15149 54004 49
$\left(\frac{\pi}{2}\right)^6$	=	15.0217 06149 61413 07
$\left(\frac{\pi}{2}\right)^7$	=	23.5960 40842 00618 62
$\left(\frac{\pi}{2}\right)^8$	=	37.0645 72481 52567 57
$\left(\frac{\pi}{2}\right)^9$	=	58.2208 97135 63712 59
$\left(\frac{\pi}{2}\right)^{10}$	=	91.4531 71363 36231 53
$\left(\frac{\pi}{2}\right)^{11}$	=	143.6543 05651 31374 95
$\left(\frac{\pi}{2}\right)^{12}$	=	225.6516 55645 350
$\left(\frac{\pi}{2}\right)^{13}$	=	354.4527 91822 91051 47
$\left(\frac{\pi}{2}\right)^{14}$	=	556.7731 43417 624

CONSTANTS (Continued)

π^2	=	9.86960 44010 89358 61883 43909 9988
$2\pi^2$	=	19.73920 88021 78717 23766 87819 9976
$3\pi^2$	=	29.60881 32032 68075 85680 31729 9964
$4\pi^2$	=	39.47841 76043 57434 47533 75639 9952
$5\pi^2$	=	49.34802 20054 46793 09417 19549 9940
$6\pi^2$	=	59.21762 64065 36151 71300 63459 9928
$7\pi^2$	=	69.08723 08076 25510 33184 07369 9916
$8\pi^2$	=	78.95683 52087 14868 95067 51279 9904
$9\pi^2$	=	88.82643 96098 04227 56950 95189 9892

$\sqrt{2}$	=	1.414 213 562 373 095 048 801 688
$1 + \sqrt{2}$	=	2.414 213 562 373 095 048 801 688
$(1 + \sqrt{2})^2$	=	5.828 427 124 746 18
$(1 + \sqrt{2})^4$	=	33.970 562 748 477 08
$(1 + \sqrt{2})^6$	=	197.994 949 366 116 30
$(1 + \sqrt{2})^8$	=	1153.999 133 448 220 72
$(1 + \sqrt{2})^{10}$	=	6725.999 851 323 208 02
$(1 + \sqrt{2})^{12}$	=	39201.999 974 491 027 40
$(1 + \sqrt{2})^{14}$	=	228485.999 995 622 956 38
$(1 + \sqrt{2})^{16}$	=	1331713.999 999 246 711
$(1 + \sqrt{2})^{18}$	=	7761797.999 999 884 751

Sin .5	=	0.47942 55386 04203
Cos .5	=	0.87758 25618 90373
Tan .5	=	0.54630 24898 43790
Sin 1	=	0.84147 09848 07896
Cos 1	=	0.54030 23058 68140
Tan 1	=	1.55740 77246 5490
Sin 1.5	=	0.99749 49866 04054
Cos 1.5	=	0.07073 72016 67708
Tan 1.5	=	14.10141 99471 707

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	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

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
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

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

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

2000 (Octal)	1024 (Decimal)
to	to
2777 (Octal)	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
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	1519	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

3000 (Octal)	1536 (Decimal)
to	to
3777 (Octal)	2047 (Decimal)

	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	1315	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	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

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

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

	0	1	2	3	4	5	6	7
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

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

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

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

	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

Octal — Decimal

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

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

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3496	3497	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

OCTAL-DECIMAL FRACTION CONVERSION TABLE

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

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.400	.500000	.500	.625000	.600	.750000	.700	.875000
.401	.501953	.501	.626953	.601	.751953	.701	.876953
.402	.503906	.502	.628906	.602	.753906	.702	.878906
.403	.505859	.503	.630859	.603	.755859	.703	.880859
.404	.507812	.504	.632812	.604	.757812	.704	.882812
.405	.509765	.505	.634765	.605	.759765	.705	.884765
.406	.511718	.506	.636718	.606	.761718	.706	.886718
.407	.513671	.507	.638671	.607	.763671	.707	.888671
.410	.515625	.510	.640625	.610	.765625	.710	.890625
.411	.517578	.511	.642578	.611	.767578	.711	.892578
.412	.519531	.512	.644531	.612	.769531	.712	.894531
.413	.521484	.513	.646484	.613	.771484	.713	.896484
.414	.523437	.514	.648437	.614	.773437	.714	.898437
.415	.525390	.515	.650390	.615	.775390	.715	.900390
.416	.527343	.516	.652343	.616	.777343	.716	.902343
.417	.529296	.517	.654296	.617	.779296	.717	.904296
.420	.531250	.520	.656250	.620	.781250	.720	.906250
.421	.533203	.521	.658203	.621	.783203	.721	.908203
.422	.535156	.522	.660156	.622	.785156	.722	.910156
.423	.537109	.523	.662109	.623	.787109	.723	.912109
.424	.539062	.524	.664062	.624	.789062	.724	.914062
.425	.541015	.525	.666015	.625	.791015	.725	.916015
.426	.542968	.526	.667968	.626	.792968	.726	.917968
.427	.544921	.527	.669921	.627	.794921	.727	.919921
.430	.546875	.530	.671875	.630	.796875	.730	.921875
.431	.548828	.531	.673828	.631	.798828	.731	.923828
.432	.550781	.532	.675781	.632	.800781	.732	.925781
.433	.552734	.533	.677734	.633	.802734	.733	.927734
.434	.554687	.534	.679687	.634	.804687	.734	.929687
.435	.556640	.535	.681640	.635	.806640	.735	.931640
.436	.558593	.536	.683593	.636	.808593	.736	.933593
.437	.560546	.537	.685546	.637	.810546	.737	.935546
.440	.562500	.540	.687500	.640	.812500	.740	.937500
.441	.564453	.541	.689453	.641	.814453	.741	.939453
.442	.566406	.542	.691406	.642	.816406	.742	.941406
.443	.568359	.543	.693359	.643	.818359	.743	.943359
.444	.570312	.544	.695312	.644	.820312	.744	.945312
.445	.572265	.545	.697265	.645	.822265	.745	.947265
.446	.574218	.546	.699218	.646	.824218	.746	.949218
.447	.576171	.547	.701171	.647	.826171	.747	.951171
.450	.578125	.550	.703125	.650	.828125	.750	.953125
.451	.580078	.551	.705078	.651	.830078	.751	.955078
.452	.582031	.552	.707031	.652	.832031	.752	.957031
.453	.583984	.553	.708984	.653	.833984	.753	.958984
.454	.585937	.554	.710937	.654	.835937	.754	.960937
.455	.587890	.555	.712890	.655	.837890	.755	.962890
.456	.589843	.556	.714843	.656	.839843	.756	.964843
.457	.591796	.557	.716796	.657	.841796	.757	.966796
.460	.593750	.560	.718750	.660	.843750	.760	.968750
.461	.595703	.561	.720703	.661	.845703	.761	.970703
.462	.597656	.562	.722656	.662	.847656	.762	.972656
.463	.599609	.563	.724609	.663	.849609	.763	.974609
.464	.601562	.564	.726562	.664	.851562	.764	.976562
.465	.603515	.565	.728515	.665	.853515	.765	.978515
.466	.605468	.566	.730468	.666	.855468	.766	.980468
.467	.607421	.567	.732421	.667	.857421	.767	.982421
.470	.609375	.570	.734375	.670	.859375	.770	.984375
.471	.611328	.571	.736328	.671	.861328	.771	.986328
.472	.613281	.572	.738281	.672	.863281	.772	.988281
.473	.615234	.573	.740234	.673	.865234	.773	.990234
.474	.617187	.574	.642187	.674	.867187	.774	.992187
.475	.619140	.575	.744140	.675	.869140	.775	.994140
.476	.621093	.576	.746093	.676	.871093	.776	.996093
.477	.623046	.577	.748046	.677	.873046	.777	.998046

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, INSTRUCTION TABLES and INDEX

GLOSSARY 1

INSTRUCTION TABLES 7

(See Section 7 for detailed instruction and designator descriptions.)

Table 1. Octal Listing of Instructions 7

Table 2. Alphamnemonic Listing of Instructions 12

Table 3. Function Listing of Instructions 17

INDEX 23

GLOSSARY

A REGISTER — Principal arithmetic register; operates as a 24-bit additive accumulator (modulus 2^{24}-1).

ABSOLUTE ADDRESS — Synonymous with 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.

ADDER — A device capable of forming the sum of two or more quantities.

ADDRESS — A 15-bit operand which identifies a particular storage location; a 17-bit operand which identifies a particular character location in storage.

ADDRESS MODIFICATION — Normally the derivation of a storage address from the sum of the execution address and the contents of the specified index register.

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".

ARGUMENT — An operand or parameter used by a program or an instruction.

ASSEMBLER — A program which translates statements to machine language. Normally, one source language statement results in the generation of one line of object code.

BASE — A quantity which defines some system of representing numbers by positional notation; radix.

BINARY-CODED DECIMAL (BCD) — A form of decimal notation where decimal digits are represented by a binary code.

BIT — Binary digit, either "1" or "0".

BLOCK — A sequential group of storage words or characters in storage.

BOOTSTRAP — Any short program which facilitates loading of the appropriate system executive.

BRANCH — A conditional jump. Refer to Jump.

BREAKPOINT — A point in a routine at which the computer may be stopped by manual switches for a visual check of progress.

B^1, B^2, B^3 REGISTERS — Index registers used primarily for address modification and/or counting.

BUFFER — Any area that is used to hold data temporarily for input or output, normally storage.

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.

CHANNEL — An Input/Output (I/O) transmission path that connects the computer to an external equipment.

CHARACTER — A group of 6 bits which represents a digit, letter or symbol from the typewriter.

CLEAR – An operation 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 – Synonymous with Instruction.

COMPILER – A program with the compatability to generate more than one line of machine code (instruction or data word) from one source language statement.

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 or storage location, the contents of which may be incremented or decremented.

D REGISTER – A 4-bit field length register used for BCD operations.

DOUBLE PRECISION – Providing greater precision in the results of arithmetic operations by appending 24 additional bits of lesser significance to the initial operands.

ENTER – The operation where the current contents of a register or storage location are replaced by some defined operand.

EQUALIZE – Adjusting the operand of the algebraically smaller exponent to equal the larger, prior to adding or subtracting the floating point coefficients.

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 or 17 bits of a 24-bit instruction. Most often used to specify the storage address of an operand. Sometimes used as the operand.

EXIT – Initiation of a second control sequence by the first, occurring when the first is near completion; the circuit involved in exiting.

F REGISTER – Program Control register. Holds a program step while the single 24-bit instruction contained in it is executed.

FAULT – Operational difficulty which lights an indicator or for which interrupt may be selected.

FIXED POINT – A notation or system of arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the binary point implicitly located at some predetermined position; contrasted with floating point.

FLIP-FLOP (FF) – A 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.

FUNCTION CODE – See Operation Code.

INCREASE – The increase operation adds a quantity to the contents of the specified register.

INDEX DESIGNATOR – A 2-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 instruction.

INDIRECT ADDRESSING – A method of address modification whereby the lower 18 bits of the specified address become the new execution address and index designator.

INSTRUCTION – A 24- or 48-bit quantity consisting of an operation code and several other designators.

INTEGRATED REGISTER FILE – The upper 64_{10} locations of core storage. Reserved for special operations with block control.

INTERRUPT – A signal which results in transfer of control, following completion of the current instruction cycle, to a fixed storage location.

INTERRUPT REGISTER – A 24-bit register whose individual bits are set to "1" by the occurrence of specific interrupt conditions, either internal or external.

INTERRUPT MASK REGISTER – A 24-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.

LIBRARY – Any collection of programs (routines) and/or subprograms (subroutines).

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" 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.

MACRO CODE – A method of defining a subroutine which can be generated and/or inserted by the assembler.

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 unused; if the mask is "1", the other quantity is used.

MASTER CLEAR – A general command produced by pressing one of two switches: a) Internal Master Clear – Clears all operational registers and control FF's in the processor. b) External Master Clear – Clears all external equipments and the communication channels.

MNEMONIC CODE – A three- or 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 instructions to a second, and makes no preparation for returning to the first sequence. Also referred to as an Unconditional Jump.

NUMERIC CODING—A system of abbreviation in which all information is reduced to numerical quantities. Also called Absolute or Machine Language coding.

OBJECT PROGRAM—The machine language version of the source program.

ONE'S COMPLEMENT—With reference to a binary number, that number which results from subtracting each bit of a 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 or output data to or from the system is fed directly from or to the external equipment.

OPERAND—Usually refers to the quantity specified by the execution address.

OPERATION CODE (Function Code)—A 6-bit quantity in an instruction specifying the operation to be performed.

OPERATIONAL REGISTERS—Registers which are displayed on the operator's 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.

PARAMETER—An operand used by a program or subroutine.

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.

P REGISTER—The Program Address Counter (P register) is a one's complement additive register (modulus 2^{15}-1) which defines the storage addresses containing the individual program steps.

PROGRAM—A precise sequence of instructions that accomplishes the solution of a problem. Also called a routine.

PSEUDO CODE—A statement requesting a specific operation by the assembler or compiler.

Q REGISTER—Auxiliary 24-bit arithmetic register which assists the A register in the more complicated arithmetic operations.

RADIX— The number of different digits that can occur in a digit position for a specific number system. It may be referred to as the base of a number system.

RANDOM ACCESS—Access to storage under conditions in which the next position from which information is to be obtained can be independent of the previous one.

READ—To remove a quantity from a storage location.

REGISTER—The internal logic used for temporary storage or for holding a quantity during computation.

REJECT—A signal generated under certain circumstances by either the external equipment or the processor during the execution of Input/Output instructions.

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, also called 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.

S REGISTER – The 13-bit S register displays the address of the word.

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.

SOFTWARE – Programs and/or subroutines.

SOURCE LANGUAGE – The language used by the programmer to define his program.

STAGE – The FFs and inverters associated with a bit position of a register.

STATUS – The state or condition of circuits within the processor, I/O channels, or external equipment.

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.

SUBROUTINE – A set of instructions that is used at more than one point in program operation.

SYMBOLIC CODING – A system of abbreviation used in preparing information for input into a computer; e.g., Shift Q would be SHQ.

TOGGLE – To complement each specified bit of a quantity, i,e.: "1" to "0" or "0" to "1".

TRANSMIT (Transfer) – The term transfer 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 transmit is often used synonymously with transfer.

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.

UNDERFLOW – An illegal change of sign from – to +, e.g., subtracting from a quantity such that the result would be less than $-(2^n-1)$, where n is the modulus. In floating point notation, this occurs where the value of the exponent becomes less than $2^{-10} + 1$ (-1777_8).

WORD—The content of a storage location. It can be an instruction or 24 bits of data.

WRITE—To enter a quantity into a storage location.

X REGISTER—An arithmetic transfer register. Nonaddressable and nondisplayed.

Z REGISTER—A 28-bit storage data register. Receives the data and parity bits as they are read from storage or written into storage. Nonaddressable but displayed on the 'T' panel in the storage module.

TABLE 1. OCTAL LISTING OF INSTRUCTIONS

OCTAL OPERATION CODE	MNEMONIC CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
00.0	HLT	m	Unconditional stop, RNI @ m upon restarting	7-30
00.1	SJ1	m	If jump key 1 is set, jump to m	7-31
00.2	SJ2	m	If jump key 2 is set, jump to m	7-31
00.3	SJ3	m	If jump key 3 is set, jump to m	7-31
00.4	SJ4	m	If jump key 4 is set, jump to m	7-31
00.5	SJ5	m	If jump key 5 is set, jump to m	7-31
00.6	SJ6	m	If jump key 6 is set, jump to m	7-31
00.7	RTJ	m	$P + 1 \rightarrow m$ (address portion), RNI @ m +1, return to m for P + 1	7-32
01	UJP,I	m,b	Unconditional jump to m	7-32
02.0	No operation (see 14.0)			
02.1-3	IJI	m,b	If $(B^b) = 0$, RNI @ P + 1; if $(B^b) \neq 0$, $(B^b) - 1 \rightarrow B^b$, RNI @ m	7-33
02.4	No operation (see 14.0)			
02.5-7	IJD	m,b	If $(B^b) = 0$, RNI @ P + 1; if $(B^b) \neq 0$, $(B^b) - 1 \rightarrow B^b$, RNI @ m	7-34
03.0	AZJ,EQ	m	If (A) = 0, RNI @ m, otherwise RNI @ P + 1	7-35
03.1	AZJ,NE	m	If (A) ≠ 0, RNI @ m, otherwise RNI @ P + 1	7-35
03.2	AZJ,GE	m	If (A) ≥ 0, RNI @ m, otherwise RNI @ P + 1	7-35
03.3	AZJ,LT	m	If (A) < 0, RNI @ m, otherwise RNI @ P + 1	7-35
03.4	AQJ,EQ	m	If (A) = (Q), RNI @ m, otherwise RNI @ P + 1	7-36
03.5	AQJ,NE	m	If (A) ≠ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
03.6	AQJ,GE	m	If (A) ≥ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
03.7	AQJ,LT	m	If (A) < (Q), RNI @ m, otherwise RNI @ P + 1	7-36
04.0	ISE	y	If y = 0, RNI @ P + 2, otherwise RNI @ P +1	7-13
04.1-3	ISE	y,b	If $y = (B^b)$, RNI @ P + 2, otherwise RNI @ P + 1	7-13
04.4	ASE,S	y	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-13
04.5	QSE,S	y	If y = (Q), RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-13
04.6	ASE	y	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1. Lower 15 bits of A are used	7-13
04.7	QSE	y	If y = (Q), RNI @ P + 2, otherwise RNI @ P + 1. Lower 15 bits of Q are used	7-13
05.0	ISG	y	If y ≥ 0, RNI @ P + 2, otherwise RNI @ P + 1	7-14
05.1-3	ISG	y,b	$(B^b) \geq y$, RNI @ P + 2, otherwise RNI @ P + 1	7-14
05.4	ASG,S	y	If (A) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-14
05.5	QSG,S	y	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-14
05.6	ASG	y	If (A) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-14
05.7	QSG	y	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-14
06.0-7	MEQ	m,i	$(B^1) - i \rightarrow B^1$; if (B^1) negative, RNI @ P + 1; if (B^1) positive, test (A) = (Q) ∧ (M), if true RNI @ P + 2; if false, repeat sequence	7-54
07.0-7	MTH	m,i	$(B^2) - i \rightarrow B^2$; if (B^2) negative, RNI @ P + 1; if (B^2) positive, test (A) ≥ (Q) ∧ (M), if true, RNI @ P + 2; if false, repeat sequence	7-55
10.0	SSH	m	Test sign of (m), shift (m) left one place end around and replace in storage. If sign negative, RNI @ P + 2; otherwise RNI @ P + 1	7-50
10.1-3	ISI	y,b	If $(B^b) = y$, clear B^b and RNI @ P + 2; if $(B^b) \neq y$, $(B^b) + 1 \rightarrow B^b$, RNI @ P + 1	7-19

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (CONTINUED)

OCTAL OPERATION CODE	MNEMONIC CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
10.4-7	ISD	y,b	If $(B^b) = y$, clear B^b and RNI @ P + 2; if $(B^b) \neq y$, $(B^b) - 1 \rightarrow B^b$ and RNI @ P + 1	7-19
11.0	ECHA	z	$z \rightarrow A$, lower 17 bits of A are used	7-15
11.4	ECHA,S	z	$z \rightarrow A$, sign of z extended	7-15
12.0-3	SHA	y,b	Shift (A). Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left and lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-50
12.4-7	SHQ	y,b	Shift (Q). Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
13.0-3	SHAQ	y,b	Shift (AQ) as one register. Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
13.4-7	SCAQ	y,b	Shift (AQ) left end around until upper 2 bits of A are unequal. Residue K = k − shift count. If b = 1, 2, or 3, $K \rightarrow B^b$; if b = 0, K is discarded	7-52
14.0	No operation		No operation (COMPASS assembled NOP)	
14.1-3	ENI	y,b	Clear B^b, enter y	7-15
14.4	ENA,S	y	Clear A, enter y, sign extended	7-15
14.5	ENQ,S	y	Clear Q, enter y, sign extended	7-15
14.6	ENA	y	Clear A, enter y	7-15
14.7	ENQ	y	Clear Q, enter y	7-15
15.0	No operation			
15.1-3	INI	y,b	Increase (B^b) by y, signs of y and B^b are extended	7-16
15.4	INA,S	y	Increase (A) by y, sign extended	7-16
15.5	INQ,S	y	Increase (Q) by y, sign extended	7-16
15.6	INA	y	Increase (A) by y	7-16
15.7	INQ	y	Increase (Q) by y	7-16
16.0	No operation			
16.1-3	XOI	y,b	$y \vee (B^b) \rightarrow B^b$	7-17
16.4	XOA,S	y	$y \vee (A) \rightarrow A$. Sign of y extended	7-17
16.5	XOQ,S	y	$y \vee (Q) \rightarrow Q$. Sign of y extended	7-17
16.6	XOA	y	$y \vee (A) \rightarrow A$, no sign extension	7-17
16.7	XOQ	y	$y \vee (Q) \rightarrow Q$, no sign extension	7-17
17.0	No operation			
17.1-3	ANI	y,b	$y \wedge (B^b) \rightarrow B^b$	7-18
17.4	ANA,S	y	$y \wedge (A) \rightarrow A$, sign of y extended	7-18
17.5	ANQ,S	y	$y \wedge (Q) \rightarrow Q$, sign of y extended	7-18
17.6	ANA	y	$y \wedge (A) \rightarrow A$, no sign extension	7-18
17.7	ANQ	y	$y \wedge (Q) \rightarrow Q$, no sign extension	7-18
20	LDA,I	m,b	$(M) \rightarrow A$	7-20
21	LDQ,I	m,b	$(M) \rightarrow Q$	7-22
22	LACH	r,1	$(R) \rightarrow A$. Load lower 6 bits of A	7-20
23	LQCH	r,2	$(R) \rightarrow Q$. Load lower 6 bits of Q	7-22
24	LCA,I	m,b	$(\overline{M}) \rightarrow A$	7-21

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (CONTINUED)

OCTAL OPERATION CODE	MNEMONIC CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
25	LDAQ,I	m,b	(M) → A, (M + 1) → Q	7-21
26	LCAQ,I	m,b	($\overline{M}$) → A, ($\overline{M + 1}$) → Q	7-21
27	LDL,I	m,b	(M) ∧ (Q) → A	7-21
30	ADA,I	m,b	Add (M) to (A) → A	7-38
31	SBA,I	m,b	(A) minus (M) → A	7-39
32	ADAQ,I	m,b	Add (M,M + 1) to (AQ) → AQ	7-40
33	SBAQ,I	m,b	(AQ) minus (M,M + 1) → AQ	7-40
34	RAD,I	m,b	Add (M) to (A) → (M)	7-38
35	SSA,I	m,b	Where (M) contains a "1" bit, set the corresponding bit in A to "1"	7-37
36	SCA,I	m,b	Where (M) contains a "1" bit, complement the corresponding bit in A	7-37
37	LPA,I	m,b	(M) ∧ (A) → A	7-37
40	STA,I	m,b	(A) → (M)	7-23
41	STQ,I	m,b	(Q) → (M)	7-24
42	SACH	r,2	(A_{00-05}) → R	7-23
43	SQCH	r,1	(Q_{00-05}) → R	7-24
44	SWA,I	m,b	(A_{00-14}) → (M_{00-14})	7-25
45	STAQ,I	m,b	(AQ) → (M,M + 1)	7-24
46	SCHA,I	m,b	(A_{00-16}) → (M_{00-16})	7-25
47	STI,I	m,b	(B^b) → (M_{00-14})	7-25
50	MUA,I	m,b	Multiply (A) by (M) → QA. Lowest order bits of product in A	7-39
51	DVA,I	m,b	(A) ÷ (M) → A, remainder → Q	7-39
52	CPR,I	m,b	(M) > (A), RNI @ P + 1 (Q) > (M), RNI @ P + 2 (A) ≥ (M) ≥ (Q), RNI @ P + 3 } (A) and (Q) are unchanged	7-53
53.1-3	TIA	b	Clear (A),(B^b) → A_{00-14}	7-27
53.40-70	TAI	b	(A_{00-14}) → B^b	7-27
53.01	TMQ	v	(v) → Q	7-27
53.41	TQM	v	(Q) → v	7-27
53.02	TMA	v	(v) → A	7-28
53.42	TAM	v	(A) → v	7-28
53.(0+b)3	TMI	v,b	(v_{00-14}) → B^b	7-28
53.(4+b)3	TIM	v,b	(B^b) → v_{00-14}	7-28
53.04	AQA		Add (A) to (Q) → A	7-26
53.(0+b)4	AIA	b	Add (A) to (B^b) → A	7-26
53.(4+b)4	IAI	b	Add (A) to (B^b) → B^b. Sign of B^b extended prior to addition	7-26
			All other combinations of 53.00-77 are undefined and will be rejected by the assembler	
54	LDI,I	m,b	(M_{00-14}) → B^b	7-22
55.0	No operation			
55.1	ELQ		(E_L) → Q	7-29
55.2	EUA		(E_U) → A	7-29
55.3	EAQ		(E_UE_L) → AQ	7-29
55.4	No operation			
55.5	QEL		(Q) → E_L	7-29
55.6	AEU		(A) → E_U	7-29
55.7	AQE		(AQ) → E_UE_L	7-29

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (CONTINUED)

OCTAL OPERATION CODE	MNEMONIC CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
56	MUAQ,I	m,b	Multiply (AQ) by (M,M + 1) → AQE	7-42
57	DVAQ	m,b	(AQE) ÷ (M,M + 1) → AQ and remainder with sign extended to E. Divide fault halts operation and program advances to next instruction	7-42
60	FAD,I	m,b	Floating point addition of (M,M + 1) to (AQ) → AQ	7-43
61	FSB,I	m,b	Floating point subtraction of (M,M + 1) from (AQ) → AQ	7-44
62	FMU,I	m,b	Floating point multiplication of (AQ) and (M,M + 1) → AQ	7-44
63	FDV,I	m,b	Floating point division of (AQ) by (M,M + 1) → AQ, remainder with sign extended to E	7-44
64	LDE	r,1	Load E with up to 12 numeric BCD characters from storage. BCD field length is specified by (D). Characters are read consecutively from least significant character at address (R + (D) − 1) until the most significant character at address R is in E. (E) is shifted right as loading progresses. The sign of the field is acquired along with the least significant character	7-48
65	STE	r,2	Store up to 13 numeric BCD characters from E. Least significant character is stored at R + (D) − 1 continuing back to most significant character stored in R	7-48
66	ADE	r,3	Up to twelve 4-bit characters (most significant character at address R) are added to (E). Sum appears in E. (D) register specifies field length	7-47
67	SBE	r,3	Up to twelve 4-bit characters (most significant character at address R) are subtracted from (E). Difference appears in E. (D) specifies field length	7-47
70.0-3	SFE	y,b	Shift E in one character (4 bit) steps. Left shift: bit 23 = "0", magnitude of shift = lower 4 bits of K = k + (B^b). Right shift: bit 23 = "1", magnitude of shift = lower 4 bits of complement of K = k + (B^b)	7-49
70.4	EZJ,EQ	m	(E) = 0, jump to m; (E) ≠ 0, RNI @ P + 1	7-49
70.5	EZJ,LT	m	(E) < 0, jump to m; (E) ≥ 0, RNI @ P + 1	7-49
70.6	EOJ	m	Jump to m if E overflows, otherwise RNI @ P + 1	7-49
70.7	SET	y	Set (D) with lower 4 bits of y	7-46
71***	SRCE,INT	c,r,s	Search for equality of character c in a list beginning at location r until an equal character is found, or until character location s is reached; 0 ≤ c ≤ 63_{10}	7-56
71****	SRCN,INT	c,r,s	Same as SRCE except search condition is for inequality	7-56
72	MOVE,INT	c,r,s	Move c characters from r to s; 1 ≤ c ≤ 128_{10}	7-58
73.0**	INPC,INT,B,H	ch,r,s	A 6- or 12-bit character is read from peripheral device and stored in memory at a given location	7-72
73.1**	INAC,INT	ch	(A) is cleared and a 6-bit character is transferred from a peripheral device to the lower 6 bits of A	7-80
74.0**	INPW,INT,B,N	ch,m,n	Word address is placed in bits 00-14, 12- or 24-bit words are read from a peripheral device and stored in memory	7-74
74.1*	INAW,INT	ch	(A) is cleared and a 12- or 24-bit word is read from a peripheral device into the lower 12 bits or all of A (word size depends on I/O channel)	7-82
75.0**	OUTC,INT,B,H	ch,r,s	Storage words disassembled into 6- or 12-bit characters and sent to a peripheral device	7-76
75.1*	OTAC,INT	ch	Character from lower 6 bits of A is sent to a peripheral device, (A) retained	7-84
76.0**	OUTW,INT B,N	ch,m,n	Words read from storage to a peripheral device	7-78

*7-bit operation code, bit 17 = "1"
**7-bit operation code, bit 17 = "0"

***7-bit operation code, bit 17 in P + 1 = "0"
****7-bit operation code, bit 17 in P + 1 = "1"

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (CONTINUED)

OCTAL OPERATION CODE	MNEMONIC CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
76.1***	OTAW,INT	ch	Word from lower 12 bits or all of A (depending on type of I/O channel) sent to a peripheral device	7-86
77.0	CON	x,ch	If channel ch is busy, reject instruction, RNI @ P + 1 If channel ch is not busy, 12-bit connect code sent on channel ch with connect enable, RNI @ P + 2	7-70
77.1	SEL	x,ch	If channel ch is busy, read reject instruction from P + 1. If channel ch is not busy, a 12-bit function code is sent on channel ch with a function enable, RNI @ P + 2	7-70
77.2	EXS	x,ch	Sense external status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-64
77.2	COPY	ch	External status code from I/O channel ch → lower 12 bits of A, contents of interrupt mask register → upper 12 bits of A; RNI @ P + 1	7-60
77.3	INS	x,ch	Sense internal status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-62
77.3	CINS	ch	Interrupt mask and internal status to A	7-62
77.4	INTS	x,ch	Sense for interrupt condition; if "1" bits occur simultaneously in interrupt lines and in the interrupt mask, RNI @ P + 1; if not, RNI @ P + 2	7-61
77.50	INCL	x	Interrupt faults defined by x are cleared	7-65
77.51	IOCL	x	Clears I/O channel or search/move control as defined by bits 00-03,08 and 11 of x.	7-63
77.511*	CILO	cm	Lockout external interrupts while channel(s) are busy	7-69
77.512	CLCA	cm	Clear channel activity, not peripherals	7-69
77.52	SSIM	x	Selectively set interrupt mask register for each "1" bit in x. The corresponding bit in the mask register is set to "1"	7-66
77.53	SCIM	x	Selectively clear interrupt mask register for each "1" bit in x. The corresponding bit in the mask register is set to "0"	7-66
77.54-56	No operation			
77.57	IAPR		Interrupt associated processor	7-66
77.6	PAUS		Sense busy lines. If "1" appears on a line corresponding to "1" bits in x, do not advance P. If P is inhibited for longer than 40 ms, read reject instruction from P + 1. If no comparison, RNI @ P + 2	7-64
77.70	SLS		Program stops if Selective Stop switch is on; upon restarting, RNI @ P + 1	7-31
77.71	SFPF		Set floating point fault logic	7-67
77.72	SBCD		Set BCD fault logic	7-67
77.73	DINT		Disables interrupt control	7-67
77.74	EINT		Interrupt control is enabled, allows one more instruction to be executed before interrupt	7-67
77.75	CTI		Set Type In / Set Type Out: Beginning character address must be present in location 23 of register file and last character address + 1 must be preset in location 33 of the file	7-71
77.76	CTO			
77.77	UCS		Unconditional stop. Upon restarting, RNI @ P + 1	7-31

*The CILO instruction is not available on the 3104-A/B or 3114-A/B Computer.

TABLE 2. ALPHAMNEMONIC LISTING OF INSTRUCTIONS

MNEMONIC CODE	OCTAL OPERATION CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
ADA,I	30	m,b	Add (M) to (A) → A	7-38
ADAQ,I	32	m,b	Add (M,M + 1) to (AQ) → AQ	7-40
ADE	66	r,3	Up to twelve 4-bit characters (most significant character at address R) is added to (E). Sum appears in E. (D) specifies field length	7-47
AEU	55.6		(A) → E_U	7-29
AIA	53.(0+b)4	b	Add (A) to (B^b) → A	7-26
ANA	17.6	y	y ∧ (A) → A, no sign extension	7-18
ANA,S	17.4	y	y ∧ (A) → A, sign of y extended	7-18
ANI	17.1-3	y,b	y ∧ (B^b) → B^b	7-18
ANQ	17.7	y	y ∧ (Q) → Q, no sign extension	7-18
ANQ,S	17.5	y	y ∧ (Q) → Q, sign of y extended	7-18
AQA	53.04		Add (A) to (Q) → A	7-26
AQE	55.7		(AQ) → $E_U E_L$	7-29
AQJ,EQ	03.4	m	If (A) = (Q), RNI @ m, otherwise RNI @ P + 1	7-36
AQJ,GE	03.6	m	If (A) ≥ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
AQJ,LT	03.7	m	If (A) < (Q), RNI @ m, otherwise RNI @ P + 1	7-36
AQJ,NE	03.5	m	If (A) ≠ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
ASE	04.6	y	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1 lower 15 bits of A are used	7-13
ASE,S	04.4	y	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1 Sign of y is extended.	
ASG	05.6	y	If (A) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-13
ASG,S	05.4	y	If (A) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1 Sign of y is extended	7-14 7-14
AZJ,EQ	03.0	m	If (A) = 0, RNI @ m, otherwise RNI @ P + 1	7-35
AZJ,GE	03.2	m	If (A) ≥ 0, RNI @ m, otherwise RNI @ P + 1	7-35
AZJ,LT	03.3	m	If (A) < 0, RNI @ m, otherwise RNI @ P + 1	7-35
AZJ,NE	03.1	m	If (A) ≠ 0, RNI @ m, otherwise RNI @ P + 1	7-35
CILO*	77.511	cm	Lockout external interrupts while channel(s) are busy	7-69
CINS	77.3	ch	Interrupt mask and internal status to A	7-62
CLCA	77.512	cm	Clear channel activity, not peripherals	7-69
CON	77.0	x,ch	If channel ch is busy, reject instruction, RNI @ P + 1 If channel ch is not busy, 12-bit connect code sent on channel ch with connect enable, RNI @ P + 2	7-70
COPY	77.2	ch	External status code from I/O channel ch to lower 12-bits of A, contents of interrupt mask register to upper 12-bits of A. RNI @ P + 1	7-60
CPR,I	52	m,b	(M) > (A), RNI @ P + 1 (Q) > (M), RNI @ P + 2 (A) ≥ (M) ≥ (Q), RNI @ P + 3 } (A) and (Q) are unchanged	7-53
CTI	77.75		Set Type In } Beginning character address must be preset in location 23 of register file and last character address + 1 must be preset in location 33 of the file	7-71
CTO	77.76		Set Type Out	
DINT	77.73		Disables interrupt control	7-67
DVA,I	51	m,b	(A) ÷ (M) → A, remainder → Q	7-39
DVAQ	57	m,b	(AQE) ÷ (M,M + 1) → AQ and remainder with sign extended to E. Divide fault halts operation and program advances to next instruction	7-42
EAQ	55.3		($E_U E_L$) → AQ	7-29
ECHA	11.0	z	z → A, lower 17 bits of A are used	7-15
ECHA,S	11.4	z	z → A, sign of z extended	7-15
EINT	77.74		Interrupt control is enabled. Allows one more instruction to be executed before interrupt	7-67

*The CILO instruction is not available on the 3104-A/B or 3114-A/B Computer.

TABLE 2. ALPHAMNEMONIC LISTING OF INSTRUCTIONS (CONTINUED)

MNEMONIC CODE	OCTAL OPERATION CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
ELQ	55.1		$(E_L) \rightarrow Q$	7-29
ENA	14.6	y	Clear A, enter y	7-15
ENA,S	14.4	y	Clear A, enter y, sign extended	7-15
ENI	14.1-3	y,b	Clear B^b, enter y	7-15
ENQ	14.7	y	Clear Q, enter y	7-15
ENQ,S	14.5	y	Clear Q, enter y, sign extended	7-15
EOJ	70.6	m	Jump to m if E overflows, otherwise RNI @ P + 1	7-49
EUA	55.2		$(E_U) \rightarrow A$	7-29
EXS	77.2	x,ch	Sense external status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-64
EZJ,EQ	70.4	m	(E) = 0, jump to m; (E) ≠ 0, RNI @ P + 1	7-49
EZJ,LT	70.5	m	(E) < 0, jump to m; (E) ≥ 0, RNI @ P + 1	7-49
FAD,I	60	m,b	Floating point addition of (M,M + 1) to (AQ) → AQ	7-43
FDV,I	63	m,b	Floating point division of (AQ) by (M,M + 1) → AQ Remainder with sign extended to E	7-44
FMU,I	62	m,b	Floating point multiplication of (AQ) and (M,M + 1) → AQ	7-44
FSB,I	61	m,b	Floating point subtraction of (M,M + 1) from (AQ)→AQ	7-44
HLT	00.0	m	Unconditional stop, RNI @ m upon restarting	7-30
IAI	53.(4+b)4	b	Add (A) to $(B^b) \rightarrow B^b$. Sign of B^b extended prior to addition	7-26
IAPR	77.57		Interrupt associated processor	7-66
IJD	02.4-7	m,b	If $(B^b) = 0$, RNI @ P + 1; if $(B^b) \neq 0$, $(B^b) - 1 \rightarrow B^b$ RNI @ m	7-34
IJI	02.1-3	m,b	If $(B^b) = 0$, RNI @ P + 1; if $(B^b) \neq 0$, $(B^b) + 1 \rightarrow B^b$ RNI @ m	7-33
INA	15.6	y	Increase (A) by y	7-16
INA,S	15.4	y	Increase (A) by y, sign of y is extended	7-16
INAC,INT	73 *	ch	(A) is cleared and a 6-bit character is transferred from a peripheral device to the lower 6 bits of A	7-80
INAW,INT	74 *	ch	(A) is cleared and a 12- or 24-bit word is read from a peripheral device into the lower 12 bits or all of A (word size depends on I/O channel)	7-82
INCL	77.50	x	Interrupt faults defined by x are cleared	7-65
INI	15.1-3	y,b	Increase (B^b) by y, signs of y and B^b are extended	7-16
INPC,INT,B,H	73 **	ch,r,s	A 6- or 12-bit character is read from a peripheral device and stored in memory at a given location	7-72
INPW,INT,B,N	74.0**	ch,m,n	Word Address is placed in bits 00-14, 12- or 24-bit words are read from a peripheral device and stored in memory	7-74
INQ	15.7	y	Increase (Q) by y	7-16
INQ,S	15.5	y	Increase (Q) by y, sign of y is extended	7-16
INS	77.3	x,ch	Sense internal status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-62
INTS	77.4	c,ch	Sense for interrupt condition; if "1" bits occur simultaneously in interrupt lines and in the interrupt mask, RNI @ P + 1; if not, RNI @ P + 2	7-61
IOCL	77.51	x	Clears I/O channel or search/move control as defined by bits 00-03, 08, and 11 of x.	7-63

*7-bit operation code, bit 17 in P = "1"
**7-bit operation code, bit 17 in P = "0"

TABLE 2. ALPHAMNEMONIC LISTING OF INSTRUCTIONS (CONTINUED)

MNEMONIC CODE	OCTAL OPERATION CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
ISD	10.4-7	y,b	If $(B^b) = y$, clear B^b and RNI @ P + 2; if $(B^b) \neq y$, $(B^b) - 1 \rightarrow B^b$, RNI @ P + 1	7-19
ISE	04.0 *	y	If y = 0, RNI @ P + 2, otherwise RNI @ P + 1	7-13
ISE	04.1-3	y,b	If $y = (B^b)$, RNI @ P + 2, otherwise RNI @ P + 1	7-13
ISG	05.0	y	If $y \geq 0$, RNI @ P + 2, otherwise RNI @ P + 1	7-14
ISG	05.1-3	y,b	If $(B^b) \geq y$, RNI @ P + 2, otherwise RNI @ P + 1	7-14
ISI	10.1-3	y,b	If $(B^b) = y$, clear B^b and RNI @ P + 2; if $(B^b) \neq y$, $(B^b) + 1 \rightarrow B^b$, RNI @ P + 1	7-19
LACH	22	r,	(R) → A; load lower 6 bits of A	7-20
LCA,I	24	m,b	$(\overline{M}) \rightarrow A$	7-21
LCAQ,I	26	m,b	$(\overline{M}) \rightarrow A$, $\overline{(M + 1)} \rightarrow Q$	7-21
LDA,I	20	m,b	(M) → A	7-20
LDAQ,I	25	m,b	(M) → A, (M + 1) → Q	7-21
LDE	64	r,1	Load E with up to 12 numeric BCD characters from storage. BCD field length is specified by (D) register. Characters are read consecutively from least significant character at address (R + (D) − 1) until the most significant character at address M is in E. (E) is shifted right as loading progresses. The sign of the field is acquired along with the least significant character	7-48
LDI,I	54	m,b	$(M_{00\text{-}14}) \rightarrow B^b$	7-22
LDL,I	27	m,b	(M) ∧ (Q) → A	7-21
LDQ,I	21	m,b	(M) → Q	7-22
LPA,I	37	m,b	(M) ∧ (A) → A	7-37
LQCH	23	r,2	(R) → Q; load lower 6 bits of Q	7-22
MEQ	06.0-7	m,i	$(B^1) - i \rightarrow B^1$; if (B^1) negative, RNI @ P + 1; if (B^1) positive, test (A) = (Q) ∧ (M); if true, RNI @ P + 2, if false, repeat sequence	7-54
MOVE, INT	72	c,r,s	Move c characters from r to s; $1 \leq c \leq 128_{10}$	7-58
MTH	07.0-7	m,i	$(B^2) - i \rightarrow B^2$; if (B^2) negative, RNI @ P + 1; if (B^2) positive, test (A) ≥ (Q) ∧ (M); if true, RNI @ P + 2; if false, repeat sequence	7-55
MUA,I	50	m,b	Multiply (A) by (M) → QA; lowest order bits of product in A	7-39
MUAQ,I	56	m,b	Multiply (AQ) by (M,M + 1) → AQE	7-42
OTAC,INT	75 *	ch	Character from lower 6 bits of A is sent to peripheral device, (A) retained	7-84
OTAW,INT	76 *	ch	Word from lower 12 bits or all of A (depending on type of I/O channel) sent to a peripheral device	7-86
OUTC, INT,B,H	75 **	ch,r,s	Storage words disassembled into 6 or 12-bit characters and sent to a peripheral device	7-76
OUTW, INT,B,H	76 **	ch,m,n	Words read from storage to peripheral device	7-78
PAUS	77.6	x	Sense busy lines. If "1" appears on a line corresponding to "1" bits in x, do not advance P. If P is inhibited for longer than 40 ms, read reject instruction from P + 1. If no comparison, RNI @ P + 2	7-64
QEL	55.5		$(Q) \rightarrow E_L$	7-29
QSE	04.7	y	If y = (Q), RNI @ P + 2, otherwise RNI @ P + 1; lower 15 bits of Q are used	7-13
QSE,S	04.5	y	If y = (Q), RNI @ P + 2, otherwise RNI @ P + 1 Sign of y is extended	7-13
QSG	05.7	y	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-14

*7-bit operation code, bit 17 = "1"

**7-bit operation code, bit 17 = "0"

TABLE 2. ALPHAMNEMONIC LISTING OF INSTRUCTIONS (CONTINUED)

MNEMONIC CODE	OCTAL OPERATION CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
QSG,S	05.5	y	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1 Sign of y is extended	7-14
RAD,I	34	m,b	Add (M) to (A) → (M)	7-38
RTJ	00.7	m	P + 1 → M (address portion) RNI @ m + 1, return to m for P + 1	7-32
SACH	42	r,2	$(A_{00-05}) \rightarrow R$	7-23
SBA,I	31	m,b	(A) minus (M) → A	7-39
SBAQ,I	33	m,b	(AQ) minus (M, M + 1) → AQ	7-40
SBCD	77.72		Set BCD fault logic	7-67
SBE	67	r,3	Up to twelve 4-bit characters (most significant character at address m) is subtracted from E. Difference appears in E. (D) register specifies field length.	7-47
SCA,I	36	m,b	Where (M) contains a "1" bit, complement the corresponding bit in A	7-37
SCAQ	13,4-7	y,b	Shift (AQ) left end around until upper 2 bits of A are unequal. Residue K = k-shift count. If b = 1, 2, or 3, $K \rightarrow B^b$; if b = 0, K is discarded	7-52
SCHA,I	46	m,b	$(A_{00-16}) \rightarrow (M_{00-16})$	7-25
SCIM	77.53	x	Selectively clear Interrupt Mask Register for each "1" bit in x. The corresponding bit in the mask register is set to "0"	7-66
SEL	77.1	x,ch	If channel ch is busy, read reject instruction from P + 1. If channel ch is not busy, a 12-bit function code is sent on channel ch with a function enable, RNI @ P + 2	7-70
SET	70.7	y	Set (D) with lower 4 bits of y	7-46
SFE	70.0-3	k,b	Shift (E) in one character (4-bit) steps. Left shift: bit 23 = "0", magnitude of shift = lower 4 bits of K = k + (B^b). Right shift: bit 23 = "1", magnitude of shift = lower 4 bits of complement of K = k + (B^b)	7-49
SFPF	77.71		Set floating point fault logic	7-67
SHA	12.0-3	y,b	Shift (A). Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-50
SHAQ	13.0-3	y,b	Shift (AQ) as one register. Shift count K = k + B^b (signs of k and B^b extended). If bit 23 of K = "1", shift right and complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left and lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
SHQ	12.4-7	y,b	Shift (Q), Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift ieft; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
SJ1	00.1	m	If jump key 1 is set, jump to m	7-31
SJ2	00.2	m	If jump key 2 is set, jump to m	7-31
SJ3	00.3	m	If jump key 3 is set, jump to m	7-31
SJ4	00.4	m	If jump key 4 is set, jump to m	7-31
SJ5	00.5	m	If jump key 5 is set, jump to m	7-31
SJ6	00.6	m	If jump key 6 is set, jump to m	7-31

TABLE 2. ALPHAMNEMONIC LISTING OF INSTRUCTIONS (CONTINUED)

MNEMONIC CODE	OCTAL OPERATION CODE	ADDRESS FIELD	INSTRUCTION DESCRIPTION	PAGE NO.
SLS	77.70		Program stops if Selective Stop switch is on; upon restarting RNI @ P + 1	7-31
SQCH	43	r,l	$(Q_{00-05}) \rightarrow R$	7-24
SRCE,INT	71*	c,r,s	Search for equality of character c in a list beginning at location r until an equal character is found, or until character location s is reached; $0 \leq c \leq 63_{10}$	7-56
SRCN,INT	71**	c,r,s	Same as SRCE except search condition is for inequality	7-56
SSA,I	35	m,b	Where (M) contains a "1" bit, set the corresponding bit in A to "1"	7-37
SSH	10.0	m	Test sign of (m), shift (m) left one place, end around and replace in storage. If sign negative, RNI @ P + 2; otherwise RNI @ P + 1	7-50
SSIM	77.52	x	Selectively set interrupt mask register for each "1" bit in x. The corresponding bit in the mask register is set to "1"	7-66
STA,I	40	m,b	$(A) \rightarrow (M)$	7-23
STAQ,I	45	m,b	$(AQ) \rightarrow (M,M+1)$	7-24
STE	65	r,2	Store up to 13 numeric BCD characters from E. Least significant character stored at R+(D)−1 continuing back to most significant character stored at R	7-48
STI,I	47	m,b	$(B^b) \rightarrow (M_{00-14})$	7-25
STQ,I	41	m,b	$(Q) \rightarrow (M)$	7-24
SWA,I	44	m,b	$(A_{00-14}) \rightarrow (M_{00-14})$	7-25
TAI	53.40-70	b	$(A_{00-14}) \rightarrow B^b$	7-27
TAM	53.42	v	$(A) \rightarrow v$	7-28
TIA	53.1-3	b	Clear (A), $(B^b) \rightarrow A_{00-14}$	7-27
TIM	53.(4+b)3	v,b	$(B^b) \rightarrow v_{00-14}$	7-28
TMA	53.02	v	$(v) \rightarrow A$	7-28
TMI	53.(0+b)3	v,b	$(v_{00-14}) \rightarrow B^b$	7-28
TMQ	53.01	v	$(v) \rightarrow Q$	7-27
TQM	53.41	v	$(Q) \rightarrow v$	7-27
USC	77.77		Unconditional stop. Upon restarting RNI @ P + 1.	7-31
UJP,I	01	m,b	Unconditional jump to M	7-32
XOA	16.6	y	y V (A) → A, no sign extension	7-17
XOA,S	16.4	y	y V (A) → A, sign of y is extended	7-17
XOI	16.1-3	y,b	y V $(B^b) \rightarrow B^b$	7-17
XOQ	16.7	y	y V (Q) → Q, no sign extension	7-17
XOQ,S	16.5	y	y V (Q) → Q, sign of y is extended	7-17

*7-bit operation code, bit 17 in P + 1 = "0"
**7-bit operation code, bit 17 in P + 1 = "1"

TABLE 3. FUNCTION LISTING OF INSTRUCTIONS

FUNCTION	MNEMONIC CODE	INSTRUCTION DESCRIPTION	PAGE NO.
Transfers	AEU†††	$(A) \rightarrow E_U$	7-29
	ANA†	$y \wedge (A) \rightarrow A$	7-18
	ANA,S	$y \wedge (A) \rightarrow A$ sign of y extended	7-18
	ANI†	$y \wedge (B^b) \rightarrow B^b$	7-18
	ANQ†	$y \wedge (Q) \rightarrow Q$, no sign extension	7-18
	ANQ,S	$y \wedge (Q) \rightarrow Q$, sign of y extended	7-18
	EAQ†††	$(E_U\ E_L) \rightarrow AQ$	7-29
	ELQ†††	$(E_L) \rightarrow Q$	7-29
	ENA	Clear A, enter y	7-15
	ENA,S	Clear A, enter y, sign extended	7-15
	ENI	Clear B^b, enter y	7-15
	ENQ	Clear Q, enter y	7-15
	ENQ,S	Clear Q, enter y, sign extended	7-15
	EUA†††	$(E_U) \rightarrow A$	7-29
	LCA,I†	$(\overline{M}) \rightarrow A$	7-21
	LCAQ,I†	$(\overline{M}) \rightarrow A$, $(\overline{M+1}) \rightarrow Q$	7-21
	LDA,I	$(M) \rightarrow A$	7-20
	LDAQ,I	$(M) \rightarrow A$, $(M+1) \rightarrow Q$	7-21
	LDE†	Load E with up to 12 numeric BCD characters from storage. BCD field length is specified by (D) register. Characters are read consecutively from least significant character at address (R + (D)-1) until the most significant character at address R is in E. (E) is shifted right as loading progresses. The sign of the field is acquired along with the least significant character	7-48
	LDI,I	$(M_{00\text{-}14}) \rightarrow B^b$	7-22
	LDL,I†	$(M) \wedge (Q) \rightarrow A$	7-21
	LDQ,I	$(M) \rightarrow Q$	7-22
	LPA,I†	$(M) \wedge (A) \rightarrow A$	7-37
	SSA,I†	Where (M) contains a "1" bit, set the corresponding bit in A to "1"	7-37
	STA,I	$(A) \rightarrow (M)$	7-23
	STAQ,I	$(AQ) \rightarrow (M, M+1)$	7-24
	STE†††	Store up to 13 numeric BCD characters from E. Least significant character stored at R + (D)-1 continuing back to most significant character stored in R	7-48
	STI,I	$(B^b) \rightarrow (M_{00\text{-}14})$	7-25
	STQ,I	$(Q) \rightarrow (M)$	7-24
	SWA,I	$(A_{00\text{-}14}) \rightarrow (M_{00\text{-}14})$	7-25
	TAI	$(A_{00\text{-}14}) \rightarrow B^b$	7-27
	TAM	$(A) \rightarrow v$	7-28
	TIA	Clear (A), $(B^b) \rightarrow A_{00\text{-}14}$	7-27
	TIM	$(B^b) \rightarrow v_{00\text{-}14}$	7-28
	TMA	$(v) \rightarrow A$	7-28
	TMI	$(v_{00\text{-}14}) \rightarrow B^b$	7-28
	TMQ	$(v) \rightarrow Q$	7-27
	TQM	$(Q) \rightarrow v$	7-27
	XOA†	$y \vee (A) \rightarrow A$, no sign extension	7-17
	XOA,S†	$y \vee (A) \rightarrow A$, sign of y is extended	7-17
	XOI†	$y \vee (B^b) \rightarrow B^b$	7-17
	XOQ†	$y \vee (Q) \rightarrow Q$, no sign extension	7-17

† Requires additional operation prior to transfer.

†† Trapped Instruction if optional floating point/48-bit precision hardware is absent.

††† Trapped Instruction if optional BCD hardware is absent.

TABLE 3. FUNCTION LISTING OF INSTRUCTIONS (CONTINUED)

FUNCTION	MNEMONIC CODE	INSTRUCTION DESCRIPTION	PAGE NO.
Transfers (Continued)	XOQ,St	y $\vee$ (Q) $\rightarrow$ Q, sign of y is extended	7-17
	MOVE,INT	Move c characters from r to s; $1 \leq c \leq 128_{10}$.	7-58
	QELttt	(Q) $\rightarrow E_L$	7-29
	SCA,I	Where (M) contains a "1" bit, complement the corresponding bit in A	7-37
	SETttt	Set (D) with lower 4 bits of y	7-46
Character Operation	ECHA	z $\rightarrow A_{00\text{-}16}$	7-15
	ECHA,S	z $\rightarrow$ A sign extended	7-15
	LACH	(R) $\rightarrow A_{00\text{-}05}$	7-20
	LQCH	(R) $\rightarrow Q_{00\text{-}05}$	7-22
	SACH	$(A_{00\text{-}05}) \rightarrow$ (R)	7-23
	SQCH	$(Q_{00\text{-}05}) \rightarrow$ (R)	7-24
	SCHA,I	$(A_{00\text{-}16}) \rightarrow (M_{00\text{-}16})$	7-25
Arithmetic	ADA,I	Add (M) to (A) $\rightarrow$ A	7-38
	ADAQ,I	Add (M,M + 1) to (AQ) $\rightarrow$ AQ	7-40
	ADEttt	Up to twelve 4-bit characters (most significant character at address R) is added to (E). Sum appears in E. (D) register specifies field length	7-47
	AIA	Add (A) to (B^b) $\rightarrow$ A	7-26
	AQA	Add (A) to (Q) $\rightarrow$ A	7-26
	AQEttt	(AQ) $\rightarrow (E_U\ E_L)$	7-29
	DVA,I	(A) $\div$ (M) $\rightarrow$ A, Remainder $\rightarrow$ Q	7-39
	DVAQtt	(AQE) $\div$ (M,M + 1) $\rightarrow$ AQ and remainder with sign extended to E. Divide fault halts operation and program advances to next instruction	7-42
	FADtt	Floating point addition of (M,M + 1) to (AQ) $\rightarrow$ AQ	7-43
	FDV,Itt	Floating point division of (AQ) by (M,M + 1) $\rightarrow$ AQ, remainder with sign extended to E	7-44
	FMU,Itt	Floating point multiplication of (AQ) and (M,M + 1) $\rightarrow$ AQ	7-44
	FSB,Itt	Floating point subtraction of (M,M + 1) from (AQ) $\rightarrow$ AQ	7-44
	IAI	Add (A) to (B^b) $\rightarrow$ B^b. Sign of B^b extended prior to addition	7-26
	INA	Increase (A) by y	7-16
	INA,S	Increase (A) by y, sign extended	7-16
	INI	Increase (B^b) by y, signs of y and B^b are extended	7-16
	INQ	Increase (Q) by y	7-16
	INQ,S	Increase (Q) by y, sign extended	7-16
	MUA,I	Multiply (M) by (A) $\rightarrow$ QA. Lowest order bits of product in A	7-39
	MUAQ,Itt	Multiply (AQ) by (M,M + 1) $\rightarrow$ AQE	7-42
	RAD,I	Add (M) to (A) $\rightarrow$ (M)	7-38
	SBA,I	(A) minus (M) $\rightarrow$ A	7-30
	SBAQ,I	(AQ) minus (M,M + 1) $\rightarrow$ AQ	7-40
	SBEttt	Up to twelve 4-bit characters (most significant character at address R) is subtracted from E. Difference appears in E. (D) register specifies field length	7-47
Jumps and Stops	HLT	Unconditional stop; RNI @ m upon restarting	7-30
	SJ1	If jump key 1 is set, jump to m	7-31
	SJ2	If jump key 2 is set, jump to m	7-31
	SJ3	If jump key 3 is set, jump to m	7-31
	SJ4	If jump key 4 is set, jump to m	7-31
	SJ5	If jump key 5 is set, jump to m	7-31
	SJ6	If jump key 6 is set, jump to m	7-31

TABLE 3. FUNCTION LISTING OF INSTRUCTIONS (CONTINUED)

FUNCTION	MNEMONIC CODE	INSTRUCTION DESCRIPTION	PAGE NO.
Jumps and Stops (Continued)	SLS	Program stops if Selective Stop switch is on; upon restarting, RNI @ P + 1	7-31
	UCS	Unconditional stop. Upon restarting, RNI @ P + 1	7-31
	UJP,I	Unconditional jump to m	7-32
	RTJ	P + 1 → m (address portion), RNI @ m + 1, return to m for P + 1	7-32
Decision	AQJ,EQ	If (A) = (Q), RNI @ m, otherwise RNI @ P + 1	7-36
	AQJ,GE	If (A) ≥ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
	AQJ,LT	If (A) < (Q), RNI @ m, otherwise RNI @ P + 1	7-36
	AQJ,NE	If (A) ≠ (Q), RNI @ m, otherwise RNI @ P + 1	7-36
	ASE	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1. Lower 15 bits of A are used	7-13
	ASE,S	If y = (A), RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-13
	ASG	If (A) ≥ y, RNI @ P + 2, otherwise @ P + 1	7-14
	ASG,S	If (A) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-14
	AZJ,EQ	If (A) = 0, RNI @ m, otherwise RNI @ P + 1	7-35
	AZJ,GE	If (A) ≥ 0, RNI @ m, otherwise RNI @ P + 1	7-35
	AZJ,LT	If (A) < 0, RNI @ m, otherwise RNI @ P + 1	7-35
	AZJ,NE	If (A) ≠ 0, RNI @ m, otherwise RNI @ P + 1	7-35
	CPR,I	(M) > (M), RNI @ P + 1 (Q) > (M), RNI @ P + 2 (A) ≥ (M) ≥ (Q) RNI @ P + 3 } (A) and (Q) are unchanged	7-33
	EOJttt	Jump to m if E overflows, otherwise RNI @ P + 1	7-49
	EZJ,EQttt	(E) = 0, jump to m; (E) ≠ 0, RNI @ P + 1	7-49
	EZJ,LTttt	(E) < 0, jump to m; (E) ≥ 0, RNI @ P + 1	7-49
	IJD	If (B^b) = 0, RNI @ P + 1; if (B^b) ≠ 0, (B^b) − 1 → B^b, RNI @ m	7-34
	IJI	If (B^b) = 0, RNI @ P + 1; if (B^b) ≠ 0, (B^b) + 1 → B^b, RNI @ m	7-33
	ISD	If (B^b) = y, clear B^b and RNI @ P + 2; if (B^b) ≠ y, (B^b) − 1 → B^b and RNI @ P + 1	7-19
	ISE	If y = 0, RNI @ P + 2, otherwise RNI @ P + 1	7-13
	ISE	If y = (B^b), RNI @ P + 2, otherwise RNI @ P + 1	7-13
	ISG	If y ≥ 0, RNI @ P + 2, otherwise RNI @ P + 1	7-14
	ISG	If (B^b) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-14
	ISI	If (B^b) = y, clear B^b and RNI @ P + 2; if (B^b) ≠ y, (B^b) + 1 → B^b, RNI @ P + 1	7-19
	SRCE,INT	Search for equality of character c in a list beginning at location r until an equal character is found, or until character location s is reached; $0 \leq c \leq 63_{10}$	7-56
	SRCN,INT	Same as SRCE except search condition is for inequality	7-56
	SSH	Test sign of (m), shift (m) left one place end around and replace in storage. If sign negative, RNI @ P + 2; otherwise RNI @ P + 1	7-50
	MEQ	(B^1) − i → B^1; if (B^1) negative, RNI @ P + 1; if (B^1) positive, test (A) ≥ (Q) ∧ (M), if true, RNI @ P + 2, if false, repeat sequence	7-54
	MTH	(B^2 − i → (B^2); if (B^2) negative, RNI @ P + 1; if (B^2) positive, test (A) ≥ (Q) ∧ (M); if true, RNI @ P + 2; if false, repeat sequence	7-55
	PAUS	Sense busy lines. If "1" appears on a line corresponding to "1" bits in x, do not advance P. If P is inhibited for longer than 40 ms, read reject instruction from P + 1. If no comparison, RNI @ P + 2	7-64

TABLE 3. FUNCTION LISTING OF INSTRUCTIONS (CONTINUED)

FUNCTION	MNEMONIC CODE	INSTRUCTION DESCRIPTION	PAGE NO.
Decision (Continued)	QSE	If y = (Q), RNI @ P + 2; otherwise RNI @ P + 1. Lower 15 bits of Q are used	7-13
	QSE,S	If y = (Q), RNI @ P + 2. Otherwise RNI @ P + 1. Sign of y is extended	7-13
	QSG	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1	7-14
	QSG,S	If (Q) ≥ y, RNI @ P + 2, otherwise RNI @ P + 1. Sign of y is extended	7-14
Shifts	SHA	Shift (A). Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-50
	SHAQ	Shift (AQ) as one register. Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
	SHQ	Shift (Q). Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = "1", shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = "0", shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off	7-52
	SCAQ	Shift (AQ) left end around until upper 2 bits of A are unequal. Residue K = k-shift count. If b = 1, 2, or 3, K → B^b; if b = 0, K is discarded	7-52
	SFE†††	Shift E in one character (4-bit) steps Left shift: bit 23 = "0", magnitude of shift = lower 4 bits of K = k + (B^b). Right shift: bit 23 = "1", magnitude of shift = lower 4 bits of complement of K = k + (B^b)	7-49
	SSH	Test sign of (m), shift (m) left one place, end around and replace in storage. If sign negative, RNI @ P + 2; otherwise RNI @ P + 1	7-50
Input/ Output	CILO*	Channel interrupt lockout	7-69
	CLCA	Clear channel activity, not peripherals	7-69
	CON	If channel ch is busy, read reject instruction from P + 1. If channel ch is not busy, 12-bit connect code sent on channel ch with connect enable, RNI @ P + 2	7-70
	COPY	External status code from I/O channel ch to lower 12-bits of A, contents of interrupt mask register to upper 12-bits of A. RNI @ P + 1	7-60
	CTI	Set Type In — Beginning character address must be preset in location 23 of register file and last character address + 1 must be preset in location 33 of the file.	7-71
	CTO	Set Type Out — Beginning character address must be preset in location 23 of register file and last character address + 1 must be preset in location 33 of the file.	7-71
	EXS	Sense external status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-60
	INAC,INT	(A) is cleared and a 6-bit character is transferred from a peripheral device to the lower 6 bits of A	7-80
	INAW,INT	(A) is cleared and a 12 or 24-bit word is read from a peripheral device into the lower 12 bits or all of A (Word size depends on I/O channel)	7-82
	INPC,INT,B,H	A 6 or 12-bit character is read from peripheral device and stored in memory at a given location	7-72
	INPW,INT,B,N	Word address is placed in bits 00-14; 12- or 24-bit words are read from a peripheral device and stored in memory	7-74
	IOCL	Clears I/O channel or search/move control as defined by bits 00-03, 08, and 11 of x.	7-63
	OTAC,INT	Character from lower 6 bits of A is sent to peripheral device, (A) retained	7-76

*The CILO instruction is not available on the 3104-A/B or 3114-A/B Computer.

TABLE 3. FUNCTION LISTING OF INSTRUCTIONS (CONTINUED)

FUNCTION	MNEMONIC CODE	INSTRUCTION DESCRIPTION	PAGE NO.
Input/ Output (Continued)	OTAW,INT	Word from lower 12 bits or all of A (depending on type of I/O channel) sent to a peripheral device	7-86
	OUTC,INT,B,H	Storage words disassembled into 6 or 12-bit characters and sent to a peripheral device	7-76
	OUTW,INT,B,H	Words read from storage to peripheral device	7-78
	SEL	If channel ch is busy, read reject instruction from P + 1. If channel ch is not busy, a 12-bit function code is sent on channel ch with a function enable, RNI @ P + 2	7-70
Interrupt	CINS	Interrupt mask and internal status to A	7-62
	DINT	Disable interrupt control	7-67
	EINT	Interrupt control is enabled, allows one more instruction to be executed before interrupt occurs	7-67
	IAPR	Interrupt associated processor	7-66
	INCL	Interrupt faults defined by x are cleared	7-65
	INS	Sense internal status if "1" bits occur on status lines in any of the same positions as "1" bits in the mask, RNI @ P + 1. If no comparison, RNI @ P + 2	7-62
	INTS	Sense for interrupt condition; if "1" bits occur simultaneously in interrupt lines and in the interrupt mask, RNI @ P + 1; if not, RNI @ P + 2	7-61
	SSIM	Selectively set Interrupt mask register, for each "1" bit in x. The corresponding bit in the mask register set to "1".	7-66
	SBCD	Set BCD fault logic	7-67
	SCIM	Selectively clear interrupt mask register for each "1" bit in x. The corresponding bit in the mask register is set to "0".	7-66
	SFPF	Set floating point fault logic	7-67

INDEX

A Register 1-7
Accumulator (See A Register)
Addressing
  Addressing Modes 7-4, 7-5
  Address Modification and Indexing 7-3
  Absolute Addresses 2-3
  Word-Character Conversions 7-2
Arithmetic
  Faults 4-1
  Fixed Point 7-38, 7-40
  Floating Point 7-43
  Supplementary Information . . . . . App. B
Assemblers (See COMPASS and BASIC Assemblers)
Assembly Listing Format 8-17
Auto Load/Auto Dump 3-3
  Examples 5-12
  Interim Subroutine 3-3
  Reserved Addresses 2-4
B[b] Registers 1-9
BASIC Assembler 8-6
BCD 7-46
Block Control and Interrupt Module 1-3
Block Control
  Section 1-11
  Clearing Mask 7-63
Breakpoint
  Address 5-13
  Mode 5-13
  Switch 5-7
C Register 1-8, 5-1
Character Address-Word Address
  Conversions 7-2
Character Positions 1-7
Character Set . . . . . App. A
Clear, Master 5-8, 5-9
Clock, Real-Time 1-12
COBOL 8-4
Codes
  Interrupt 4-5
  Error 8-18
  Typewriter Character 6-5
Coding
  Procedures 8-7
  Forms 8-19
Compare Instruction 7-53
Communication Register (See C Register)
COMPASS Assembler 8-2
  Coding Form 8-19
Computer Organization 1-7
Consoles 1-4, 5-1
  Switches and Indicators 5-2
Console Keyboard 5-8
Console Typewriter 6-1
Control Instructions 7-63
Conversions
  Word Address-Character Address 7-2
  Numbers B-5
  Octal-Binary C-6, C-10
D Register 1-9
Data Bus 1-11
  Register 1-9
Data Processing Package 8-3
E Register 1-9, 5-1
Error Codes 8-18
F Register 1-8
Faults, Arithmetic 4-1
Field Length Register (See D Register)
Fixed Point Arithmetic B-8
Floating Point Arithmetic B-9
FORTRAN 8-5
  Coding Form 8-19
Indexing
  Address Modification 7-3
  Examples 7-5
Index Registers (See B [b] Registers)
Input/Output
  Channels 1-3
  Characteristics 3-1
  Instructions 7-68, 7-69
  Interface Signals 3-1
  Parity 1-14, 3-2
Instructions (See also specific instructions) . 7-1
  Execution Times 7-11
  Format 7-1, 7-14, 7-45, 8-7
  Interpretations 8-8
  Index 7-8
  Listings 7-7
    (Also see back of book for Octal, Alphamnemonic and Function Listings on pp. 7, 12, and 17)
  Macro 8-12
  Pseudo- 8-9
  Symbols . . . . . pp. 7, 12 (in back of book)
  Trapped 7-6, 7-7 (See also pp. 7-42 to p. 7-49)
Interface Signals 3-1
Inter-Register Transfer
  24-Bit Precision 7-26
  48-Bit Precision 7-29
Interrupt
  Clearing and Sensing 4-4
  Codes 4-5
  Control 4-4
  Instructions 7-65

Internal 4-1
I/O 4-3
Mask Register 4-3
Power Failure 4-2
Priority 4-4
Processing 4-5
Real time 4-2
System 4-1
Trapped Instruction 4-2
Jump Instructions 7-30
Keyboard (See Console Keyboard)
Load Instructions 7-20
Logical Instructions
(with Storage Reference) 7-37
Loudspeaker, Console 5-4
Macro Instructions 8-12
Main Control & Arithmetic Module 1-3
Master Clear 5-8, 5-9
Memory
Configurations, Optional 1-3
Protection 2-3, 2-5
Meters, Elapsed Time 5-15
Modularity 1-1
Move Instruction 7-58
Number Systems B-1
Octal-Decimal Conversion Table
Integer C-6
Fraction C-10
Optional Arithmetic 1-4
P Register 1-8
Pause Sensing Mask 7-64
Parameters, Instruction (See Symbol Definitions)
Parity 1-14
Storage 1-14
I/O 1-15, 3-2
Peripheral Equipment 1-1, 1-15
Power Control Panel 1-7, 5-15
Program Address Counter (See P Register)
Programming Reference Table . . . . App. C
Pseudo-Instructions 8-9
Q Register 1-8
Radix Arithmetic B-6
Read/Write Characteristics 2-2
Single-Character Mode 2-2
Double-Character Mode 2-2
Triple-Character Mode 2-2
Full-Word Mode 2-2
Address Mode 2-2
Real-Time Clock 1-12
Registers (See also specific registers)
Characteristics 1-10
Descriptions 1-7
Displays 5-1
Operations 7-12
Shifts 7-50
Register File 1-11
Register File Assignments 1-11
S Bus 1-11
S Register 1-9
Satellite Configurations 3-5
Search Instructions 7-50, 7-56
Sense Instructions 7-60
SCOPE 8-1
Software Systems 8-1
SORT/MERGE Program, Generalized 8-5
Status
Checking (Typewriter) 6-2
Indicators 5-4, 5-5
Internal Status Sensing Mask 7-62
Stop Instructions 7-30
Storage
Addressing 2-3
Characteristics 2-1
Control Panel 2-1
Module 1-3
Parity 1-13
Protection 2-3
Registers 2-2
Sharing 2-3
Shift 7-50
Word Format 1-7
Store Instructions 7-23
Switches
Console 5-7, 5-10
Keyboard 5-9
Power Control Panel 5-15
Storage Protection 2-5
Typewriter Console 6-3
Symbol Definitions 7-3
System Description 1-1
Temperature Warning 5-6
Typewriter, Console 6-1
Codes 6-5
Status Checking 6-2
Switches and Indicators 6-4
Tabs, Margins and Spacing 6-2
Type Dump 6-3
Type In 6-3
Type Load 6-3
Type Out 6-3
Utility Package 8-4
Word Address 7-1
Word Address – Character Address
Conversions 7-2
Word Format 1-7
X Register 1-8
Z Register 1-9

COMMENT SHEET

MANUAL TITLE CONTROL DATA 3100 COMPUTER SYSTEM

Reference Manual

PUBLICATION NO. 60108400 REVISION D

FROM: NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

FORM CA231 REV.1-67

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Technical Publications Department
4201 North Lexington Avenue
Arden Hills, Minnesota 55112

CUT ALONG LINE

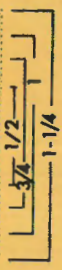
FOLD

FOLD

3100 CHARACTERISTICS

- Stored-program, solid-state, general-purpose computer.
- Diode logic.
- Parallel mode of operation.
- Single address logic.
- Programmed inter-register transfers.
- Address modification (indexing).
- Indirect addressing.
- Character and word addressing (4 characters per word).
- 28-bit storage word (24 data bits and 4 parity bits).
- Nonvolatile magnetic core storage. Standard memory: 4,096 words/16,384 characters.
- Selected storage protection.
- Storage sharing.
- Complete cycle time: 1.75 microseconds
- Access time: 1.0 microsecond.
- 24-bit accumulator register and auxiliary accumulator register.
- Binary arithmetic: 2^{24}-1 modulus, one's complement for all single precision (24-bit) operations and double precision (48-bit) addition and subtraction.
- Instruction repertoire compatible with the 3200 and 3300 Computers.
- Trapped instruction processing: executes double precision multiplication and division, floating point, binary coded decimal (BCD) and an optional register transfer instruction if optional arithmetic logic is not present in a system.
- Integrated register file.
- Complete interrupt system.
- Block control operations.
- Logical and sensing operations.
- Masked storage searches.
- Three 15-bit index registers.
- Real-time clock (1.0 millisecond incrementation)
- Integrated upright console featuring:
  - Binary register displays.
  - Internal and external status displays.
  - Instruction step control.
  - Breakpoint thumbwheel control.
  - Auto step control.
  - Auto Load.
  - Auto Dump.
  - Detachable keyboard for manual entry and control of the computer.
- Standard 3000 Series type 12-bit bidirectional data channel.
- Data transfer rate up to 3.3 megabits/second.
- Compatible I/O mediums include magnetic tape, disk file, punched cards, paper tape and printed forms.
- Options include:
  - Memory expansion to 8,192; 16,384 or 32,768 words.
  - 3 additional 12-bit data channels, or 1 additional 12-bit data channel and 1 24-bit data channel.
  - Floating point and 48-bit precision multiply and divide hardware logic.
  - BCD arithmetic hardware logic.
  - On-line I/O monitor typewriter.
  - Desk-sized console with octal and decimal displays, or
  - Desk-sized console equipped with nondetachable keyboard, binary displays and on-line electric typewriter.
  - Complete selection of peripheral equipment.

CONTROL DATA



>>> CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.