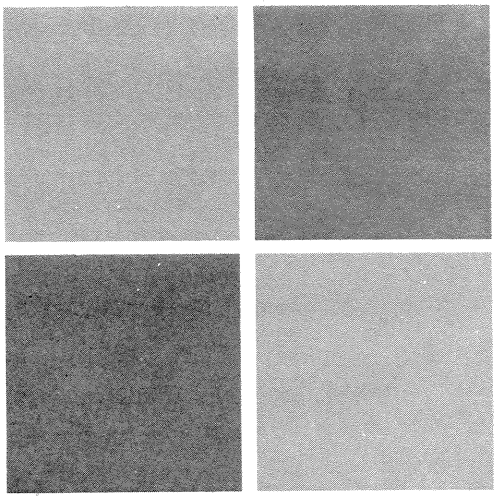
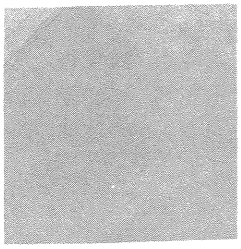
SCC 4700 COMPUTER



REFERENCE MANUAL



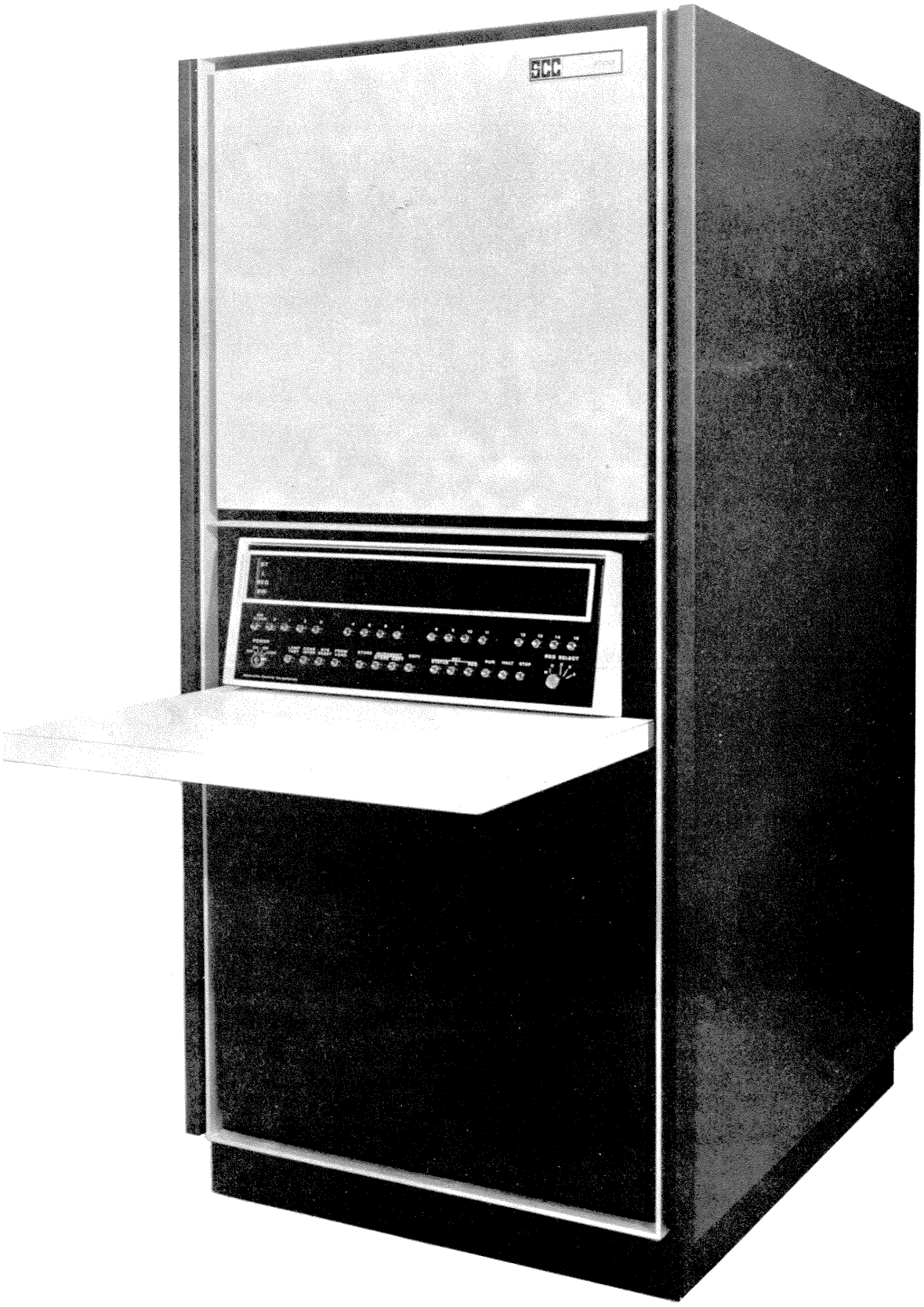
SCIENTIFIC CONTROL CORPORATION

SCC 4700 COMPUTER
REFERENCE MANUAL

SCIENTIFIC CONTROL CORPORATION

1215 W. Crosby Rd., Carrollton, Texas 214-214-6555

SCC 52 10–46–91



SCC 4700 Computer

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I
INTRODUCTION

The Scientific Control Corporation Model 4700 represents a new design approach to the small computer field. Advanced hardware design techniques provide a cost performance ratio unmatched in any other small computer. The SCC 4700 implements several features, such as memory mapping and floating point hardware, found only in much larger machines to provide low cost configurations in real time, time sharing, scientific, and general purpose applications not formerly available.

High-speed logic forms and advanced construction techniques have been used throughout to create a "state-of-the-art" machine which will have a high degree of reliability and avoid premature obsolescence.

The SCC 4700 is a general purpose, high-speed, binary computer with a single address type of instruction. It features a high-speed magnetic core memory module consisting of 4,096 sixteen-bit words, with a 920-nanosecond cycle time, which permits a wide variety of real time applications.

The SCC 4700 has outstanding design features such as:

1. A microprogrammed read-only memory for flexible and economical internal logic
2. Fully integrated circuitry using the most advanced TTL integrated circuits
3. An etched circuit back-plane board eliminating "bird nest" wiring
4. "Register slice" internal organization for easy maintainability
5. Programmable memory protection (optional) which provides flexible read-only write-only, or execute-only protection
6. Memory mapping (optional) for implementation of multiprogramming techniques
7. Byte addressing for efficient processing of character strings, particularly those in ASCII or EBCDIC code
8. Real time byte-oriented I/O structure utilizing low cost multiplexor and high-speed selector channels for data transfer
9. Powerful multilevel priority interrupt system which minimizes program overhead and provides quick real time responses
10. Extensive instruction set which simplifies system and application programming while providing outstanding flexibility and power.

These and other advanced concepts make the SCC 4700 faster and more flexible than any other 16-bit machine.

CONFIGURATION

The main modules of the basic SCC 4700 (Figure 1) are as follows:

4K Memory Module

Central Processing Unit

Multiplexor Channel

Control/Display Panel.

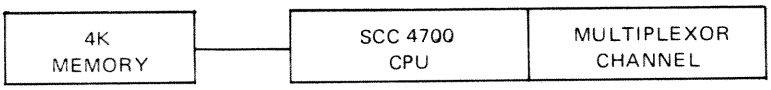


Figure 1. Basic Configuration

The SCC 4700 may be expanded as shown in Figure 2 to include the following:

Four 16K Memory Modules

Memory Mapping with up to 32 Associative Registers

Memory Gating with up to 8 DMA Ports

Central Processing Unit with Multiply/Divide, Double Precision, and Floating Point Hardware (Additional CPU's may be added for multiprocessing).

Four I/O Channels in any combination of multiplexor and/or selector channels

Peripheral Devices (up to 64 per channel).

GENERAL CHARACTERISTICS

The advanced characteristics of Scientific Control Corporation's Model 4700 contributing to its outstanding efficiency for general purpose, real time, and time sharing applications are as follows:

1. **Large Memory Capacity**

   The basic memory module of the SCC 4700 consists of 4,096 sixteen-bit words. The memory capability is expandable to 65,536 words in 4,096-word increments.

2. **High-Speed Memory**

   The 4K, 8K, and 16K memory modules are 2-1/2 dimensional with a 920-nanosecond full cycle time.

3. **Memory Parity**

   Logic circuits are provided if the optional memory parity checking feature is desired. Memory modules with an extra parity bit in each word are available for use with this feature.

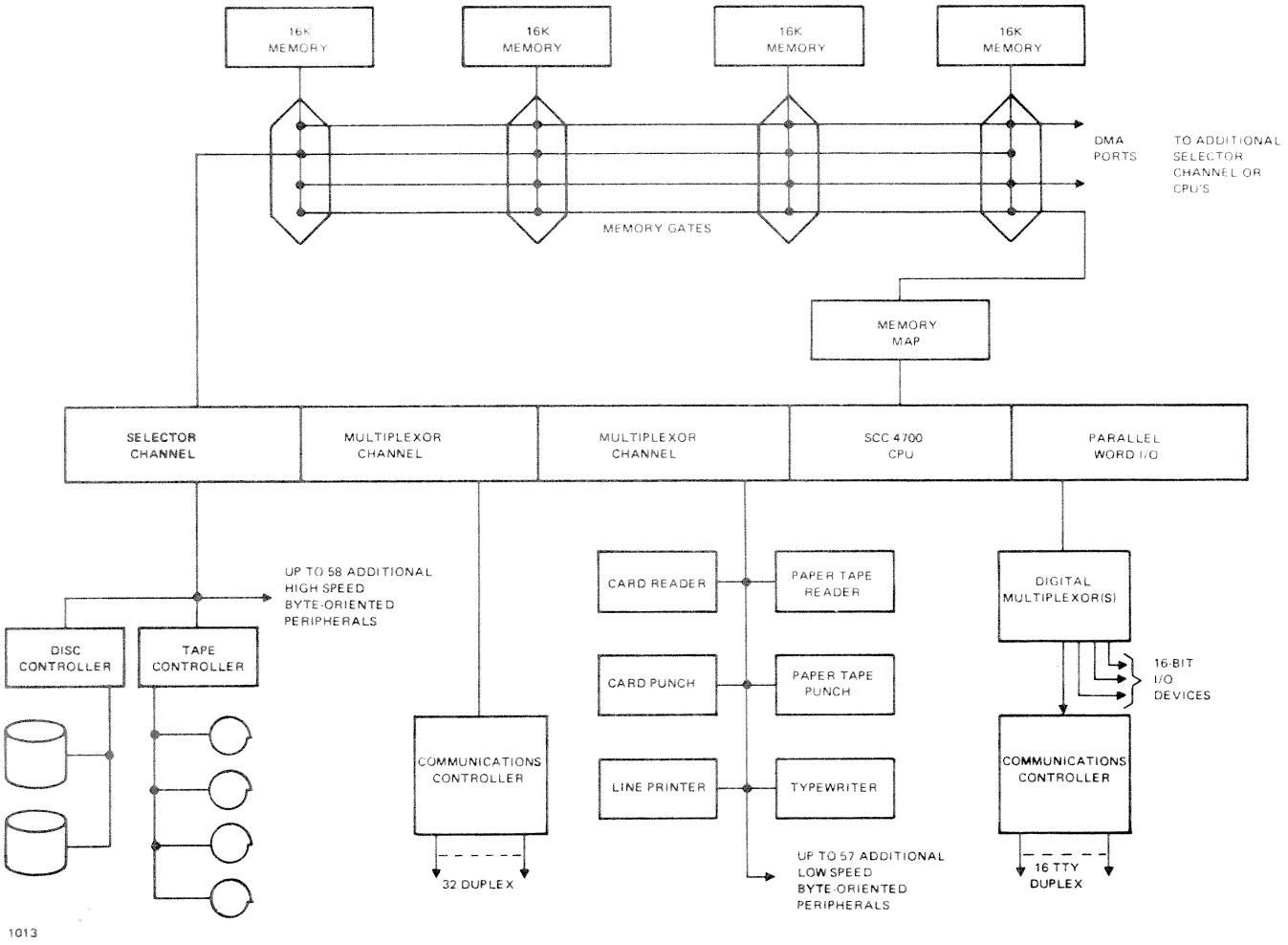


Figure 2. Typical Expanded Configuration

4. **Byte Addressing**

Byte, word, double word, and triple word addressing are provided for maximum efficiency.

5. **Memory Accessing**

The capability of asynchronous memory access by either the Central Processing Units, Selector Channel Units, or special system components is provided by the four-port memory gate. The memory gates permit independent memory module accessing and allow complete overlapping of input, processing, and output operations. Simultaneous requests of the same module are handled on a priority basis with the priority assignment being a system variable.

6. **Memory Mapping and Protection**

This optional unit contains a set of associative registers through which program addresses are transformed to real core locations. The memory mapping system divides all available memory into "pages." For each page, the programmer may specify one of three levels of memory protection: read, write, or execute.

Accidental or unauthorized use of a memory location is prevented by checking the memory access code for the page being referenced before execution. Attempts to violate the specified protection causes a trap. In this manner, the protection feature permits the maintenance of system integrity in a multiprogramming environment and complete security of system and application programs.

7. **Microprogrammed Hardware**

The instruction repertoire is microprogrammed to provide a highly flexible and reliable instruction set. The design and organization of the microprogrammed instruction repertoire permits the addition of hardware arithmetic or special function instructions to be implemented economically.

8. **Extensive Instruction Set**

The powerful instruction set allowing efficient and flexible programming includes:

a. **Addressing Modes**

The addressing modes available to the programmer permit several modes of memory addressing: Indirect, Indexed, Direct, and Relative. Both pre-indexing and post-indexing are provided with a hardware index register.

b. **Load and Store Instructions**

Direct loading and storing of major programmable registers including byte and multi-register operations may be performed in a single instruction.

c. **Literal Instructions**

Literal instructions which utilize the last 9 bits of the instruction as a signed operand are included in the instruction set. This feature permits high-speed arithmetic and logical operations involving a single character operand.

d. **Inter-Register Instructions**

A special set of instructions called the Operate Instructions provide logical and arithmetic operations between the A, B, X, and E registers. This feature also provides the capability of performing a skip, depending on the result of the logical or arithmetic operation performed upon the registers. Data may also be exchanged between two registers in only 1.1 microsecond.

e. **Shift Instructions**

The instruction repertoire includes 16 indexable shift instructions. Shifts may be made to the left or right on a single 16-bit word in the A register or on two 16-bit words in the A and B registers.

f. **System Instructions**

System calls and returns permit up to 64 direct access entry points to monitor functions and provide efficient context switching by saving and restoring machine status and programmable registers.

g. **Privileged Instructions**

These instructions prevent unauthorized changing of machine status by user programs.

Also included in the instruction set are several special instructions which reduce total system cost and program overhead by providing efficient handling of special functions.

9. **Hardware Arithmetic**

Three out of four optional high-speed hardware arithmetic instruction sets may be added to the standard instruction repertoire.

a. Integer Multiply and Divide

b. Fractional Double Precision to provide greater accuracy through additional significant digits.

c. Floating Point or Double Floating Point to provide greater precision and versatility in scientific application areas.

10. **System Control**

System control and autonomy in multiprogramming environments are maintained and simplified through hardware design which facilitates transfer of control, parameters, and status information between the system and the user.

11. **Fully Parallel Operation**

Fully parallel internal processing of all instructions together with full parallel data transfer between memory and the CPU provide a high-speed system performance capability.

12. **Versatile I/O**

The SCC 4700 may use from one to four byte-oriented input/output channels to communicate with memory either directly or through the Central Processing Unit. The channel units may be multiplexor and/or selector channels, each of

which can interface up to 64 device controllers. These channels provide a wide variety of input/output capability from single byte (character) data transfer controlled directly by the CPU to transfer of data by blocks asynchronously and independently of the CPU.

Through the use of pointers, the input/output system permits data chaining to provide scatter read/gather write techniques and the use of separate control and data areas.

The I/O channels are compatible with all of the SCC 700 series configurations.

Data transfer of full 16-bit words directly to and from the accumulator is accomplished through the program controlled parallel I/O, standard with the SCC 4700. Special high-speed full word transfers directly to memory are performed by the optional Direct Memory Access (DMA) ports.

13. **Automatic Initial Program Loading**

For operator convenience, automatic initial program loading is provided.

14. **Watchdog Timer**

Program hangup due to I/O failure is prevented by a watchdog timer.

15. **Priority Interrupt System**

The SCC 4700 interrupt capability allows the normal execution of a program to be interrupted in order to execute a program of higher priority. The priority structure resolves contention problems arising from the simultaneous occurrence of interrupt conditions and permits servicing of interrupts according to priority.

Two interrupts are standard: console and I/O channel. In addition, two conditions cause traps to reserved locations: real time clock = 0 and unimplemented instructions.

Up to 256 external interrupts in 16 levels, power up, power down, three additional channel interrupts, memory parity, privileged instruction, floating point over/under flow, and memory protection may be added to the SCC 4700 to fulfill specialized requirements.

16. **Real Time Clock**

Standard in the SCC 4700 is a real time clock which decrements a fixed location and causes a trap when the contents of the location reach zero.

17. **Power Failure Protection**

The design of the optional power failure feature ensures program integrity in the event of power system failures. If power failure occurs, this feature activates an interrupt which stores the active registers in memory and initiates software routines to provide an orderly system shutdown. Reserved power permits an excess of 5 milliseconds of normal operation after detection of power failure. When power returns to the system, the active registers are restored and a power up interrupt is generated.

18. **Remote Control Console**

The control console may be installed at a location remote to the CPU to provide ready access for the operator.

19. **All Integrated Circuits**

All circuits utilize the most advanced, highly reliable, fast-switching, low noise TTL integrated circuit components. These components are mounted on extra large printed circuit plug-in boards in a "register slice" arrangement.

20. **Comprehensive Software Package**

The basic software package is designed to run on the SCC 4700 with a 4K memory and an ASR 33. Included in the basic software package are: a macroassembler, a relocatable loader, the SCC Basic FORTRAN system, one fixed point and two floating point math subroutine packages, a basic diagnostic program package, and a comprehensive utility system. The utility system includes an on-line debugging package, I/O drivers, and conversion routines.

Software for an expanded configuration of the SCC 4700 includes the Real Time Monitor and FORTRAN IV.

The optional peripheral devices available with the SCC 4700 consist of the following:

1. **Teletype Input/Output Device**

Two units are available which operate at a rate of 10 characters per second. The ASR 33 and ASR 35 both contain paper tape reader and paper tape punch in addition to the normal typewriter keyboard.

2. **Selectric Typewriter**

The IBM Selectric unit operates at a maximum speed of 15 characters per second and is equipped with a 15 inch pin feed platen, allowing use of continuous form paper.

3. **Paper Tape Reader with Controller**

This 8-level unidirectional unit operates at a nominal speed of 300 characters per second. A bidirectional paper tape spooler is also available.

4. **Paper Tape Punch with Controller**

Two 8-level units are available for punching at a rate of 50 or 120 characters per second.

5. **Disc Storage Unit and Controller**

Three models of the disc storage unit are available with an access time of 17 milliseconds and a memory capacity of 128K, 256K, or 512K bytes. The controller accommodates from one to four disc units.

6. **Disc Pack Drive and Controller**

Three models of the disc pack drive are available: two in non-IBM compatible format with 29- or 54-million bit capacity, and one in IBM compatible format with a capacity of 54-million bits. The controller accommodates from one to four disc pack drives and is available in IBM and non-IBM compatible format.

7. **Magnetic Tape Unit and Controller**

Nine magnetic tape unit models are available: five for 7-track tape and four for 9-track tape. The 7-track units can read or write in densities of 200, 556, or 800 bits per inch at speeds ranging from 25 to 150 inches per second. The 9-track units read/write 800 bits per inch at speeds ranging from 37.5 to 150 inches per second. All tape units are IBM compatible. The controllers for the 7- and 9-track tape units accommodate from one to four tape units.

8. **Line Printer and Controller**

Three models are available with printing rates of 300, 600, or 1000 lines per minute. Each model prints a selection of 64 characters with 132 characters per line.

9. **Card Reader and Controller**

One model is available for reading a 256-character set at a rate of 200 cards per minute.

10. **Card Punch and Controller**

One model is available for punching from a 256-character set at a rate of 100 cards per minute.

11. **Digital to Analog Converter**

The digital to analog converter is an addressable, multichannel, binary, two's complement converter which may be 7-bit plus sign bipolar or 8-bit unipolar.

12. **Multiplexor-Encoder**

This unit permits sequential sampling of 64 analog channels of 0 to 5 volts dc. The number of channels can vary in groups of 8 up to the maximum of 64. The multiplexor-encoder contains a buffer amplifier and a sample and hold amplifier. The unit outputs a 12-bit unipolar or 11-bit plus sign digital data word with a digitizing range of up to 40 kc. The computer can address the multiplexor-encoder for sampling of the analog channels sequentially.

13. **Communications Equipment**

Three communications controllers are available. One provides line termination and multiplexing for one to three full-duplex line terminals. The second model provides for multiplexing of 1 to 32 full-duplex line terminals. The third model is a bit-oriented communication controller for 1 to 16 teletype circuits with selectable clock rates.

SECTION II

MACHINE ORGANIZATION

The basic SCC 4700 Central Processor Unit contains nine 16-bit registers and may be expanded to include two additional registers and the memory map unit. The organization of the CPU provides extremely fast data and arithmetic capability.

Figure 3 below is a block diagram of the SCC 4700 Central Processor Unit. All arrows indicate data paths for 16-bit parallel transfers. The optional registers and the memory map unit are enclosed in heavy broken lines.

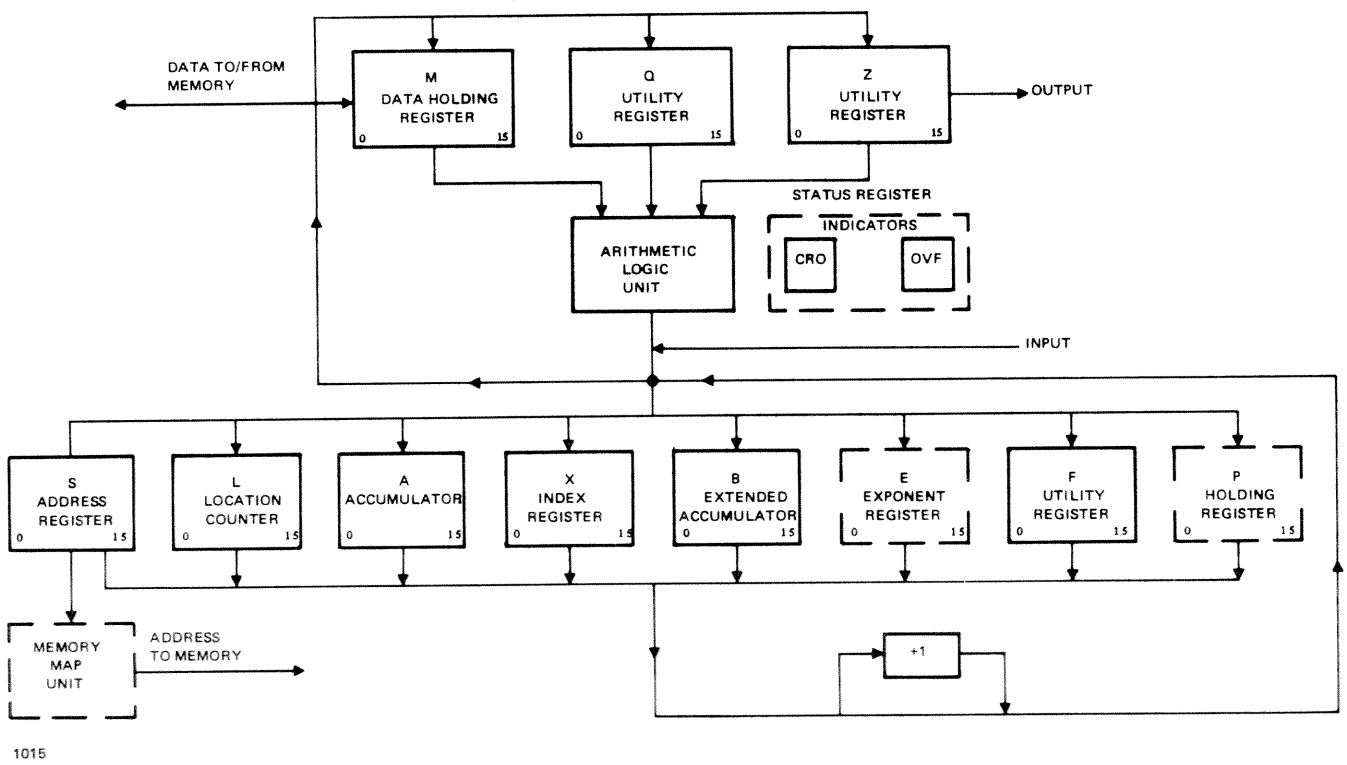


Figure 3. SCC 4700 Central Processing Unit, Registers, and Data Paths

The registers of the CPU are as follows:

M Register — The M register accepts and transfers data to and from memory. The M register also serves as a holding register for instructions and data.

Q Register — The Q register is a utility register which serves as a holding register for the arithmetic logic unit.

Z Register — The Z register is a utility register which also serves as a holding register for the arithmetic logic unit.

S Register — The S register is the memory address register which supplies a 16-bit address to the memory unit (or to the memory map unit, if implemented).

L Register — The L register is the location counter (program address register) which contains the address of the instruction being fetched.

A Register — The A register is the main accumulator. Most arithmetic and logical operations use this register to hold the result of the operation.

B Register — The B register serves as a second accumulator and extension of the A register for double precision and floating point operations.

X Register — The X register is used for indexing and address modification.

E Register (optional) — The E register is utilized during floating point instructions to hold the exponent of a number in floating point format.

F Register — The F register is a utility register which serves as a holding register to provide high-speed, internal operations for double precision and floating point instructions.

P Register (optional) — The P register is a holding register used to save necessary data during memory mapping, memory protection, and memory parity operations.

Status Register — The Status Register is a composite of six indicators which indicate the mode of operation of the machine and the occurrence of certain special conditions. The state of these indicators (0 or 1) can be changed from the control console, under program control, or by the CPU in response to changes in processing or the operating environment.

1. **Indirect Mode Indicator** — This indicator is meaningful only when memory mapping is implemented. The Indirect Mode Indicator allows the system to use the user map on indirect memory references. Bit 10 is set (=1) by a SIUM instruction and reset (=0) by a System Call instruction, the SYSTEM RESET button on the console, or a SISM instruction.

   This indicator affects the interpretation of the Mode Indicator when indirect addressing is specified and the Mode Indicator is set to system mode. (Refer to Mode Indicator below).

2. **Mode Indicator** — The Mode Indicator is meaningful only when the memory map unit is implemented. If bit 11 of the Status Register is set (=1), the CPU is in the user mode. Bit 11 is reset (=0) when executing instructions in the system mode; however, if indirect addressing is indicated, bit 11 is given the value of bit 10 above. Privileged instructions may be executed only when this bit indicates system mode.

   If memory mapping is not implemented, the machine will always be in the system mode.

3. **Interrupt Indicator** — The interrupt system may be enabled or disabled under program control. If bit 12 of the Status Register is set (=1), the interrupt system is disabled and any armed interrupts which occur are remembered but do not interrupt program execution. If bit 12 is reset (=0), the interrupt system is enabled and armed interrupts are allowed to go active according to their priority.

4. **Halt Indicator** — This indicator specifies whether the CPU is in run or halt status. The CPU is halted if bit 13 of the Status Register is set (=1) through use of the Halt instruction, the HALT button, or the SYSTEM RESET button on the console. (Only single step execution of instructions can be performed while this indicator is set.) Bit 13 is reset (=0) by certain interrupts, some traps, and the RUN button on the console.

5. **Overflow Indicator** — The overflow indicator (bit 14 of the Status Register) is set if the result of the arithmetic operation exceeds the maximum signed magnitude quantity which can be contained in the accumulator. Bit 14 is not reset (=0) if an overflow does not occur.

6. **Carryout Indicator** — This indicator (bit 15 of the Status Register) is set (=1) if a carry occurs in the adder from the high order position (bit 0). Bit 15 is reset (=0) if a carry from bit position 0 does not occur.

   An arithmetic operation may result in both an overflow and carryout.

Memory Map Unit (optional) — The Memory Map Unit contains 8 to 32 associative registers and 2 map table pointer registers. The number of associative registers may be increased in groups of four. (Refer to Section VI for the operation of this unit and its registers.)

SECTION III

INSTRUCTION REPERTOIRE

CONVENTIONS AND DEFINITIONS

The conventions used in this chapter are as follows:

1. A register name or an address enclosed in parentheses ( ) denotes the contents of the register or address.

2. A location enclosed within two sets of parentheses (()) indicates an indirect address. Each additional set of parentheses indicates another level of indirect addressing.

3. Subscription denotes bit positions within a register or instruction. Example: INS_{0-6} indicates bits 0 through 6 of the instruction INS.

4. Abbreviations and labels indicate the following:

1	Set (equal to 1)
0	Reset (equal to 0)
A	Accumulator (A register)
B	Extended accumulator (B register)
BA	Base address
CNT	Count, the number of places to be shifted
CRO	Carryout or carryout indicator
D	Destination register
E	Exponent register (E register)
EA	Effective address
I	Indirect address bit
INS	Instruction
L	Location counter or length of shift
OVF	Overflow or overflow indicator
PA	Primary address
R	Relative addressing bit
S	Sign bit or source register
ST	Status register
X	Index register or index bit
Y	Address field

5. An arrow (→) is used to denote copying information from a register, address, or instruction to another location.

   Example: (A) → B indicates the contents of the A register are copied into the B register.

6. The symbol ∩ denotes the Logical AND operation defined in the following table.

   Example: (A) ∩ (B)

(A) \ (B)	0	1
0	0	0
1	0	1

   The Logical OR operation is designated by the character ∪ and is defined in the following table.

   Example: (A) ∪ (B)

(A) \ (B)	0	1
0	0	1
1	1	1

   The symbol ⊕ denotes the Exclusive OR operation which is defined in the following table.

   Example: (A) ⊕ (B)

(A) \ (B)	0	1
0	0	1
1	1	0

7. A horizontal bar placed over a register name enclosed in parentheses indicates the one's complement of the contents of the register.

   Example: $(\overline{A})$ denotes each bit of the A register is inverted.

   Two's complement is indicated by adding +1.

   Example: $(\overline{A}+1)$ designates each bit of the A register is inverted and one is added to the result.

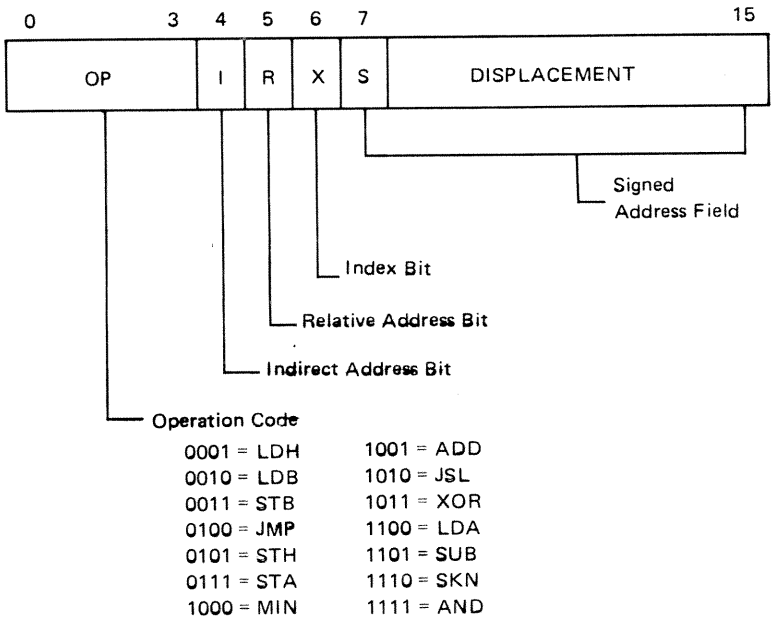
FORMATS

INSTRUCTION FORMATS

The three types of formats used in forming instructions are (1) Basic, (2) Literal, and (3) Extended Op Code. A description of each of these formats and their variations is given below. (Descriptions of each instruction available on the SCC 4700 is covered in the section "Instruction Set" of this section.)

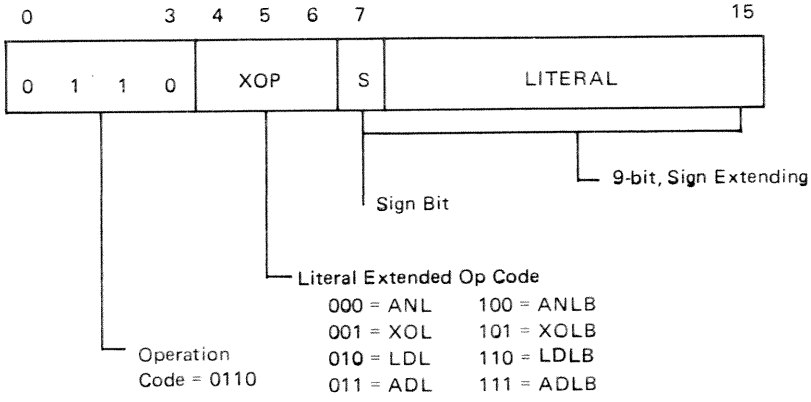
1. Basic Format

The most commonly used instructions for memory referencing operations are those written in the basic format. (Refer to "Addressing" for the various modes available.)



2. Literal Format

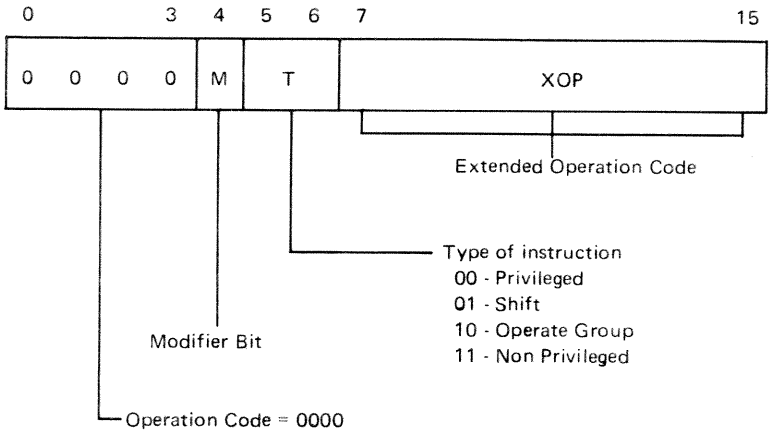
This format applies to all literal instructions, such as LDL (Load A, Literal) and LDLB (Load B, Literal). Literal instructions are assigned an operation code of 0110 and follow the format below.



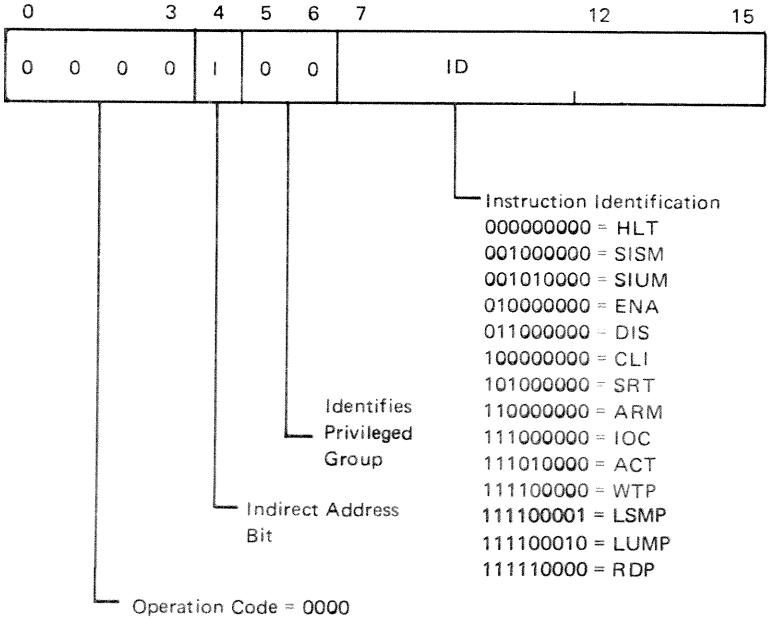
The least significant nine bits of the literal instruction become a 16-bit operand when the sign bit (bit 7) is propagated; i.e., locations 0 through 6 are made equal to bit 7.

3. Extended Op Code Format

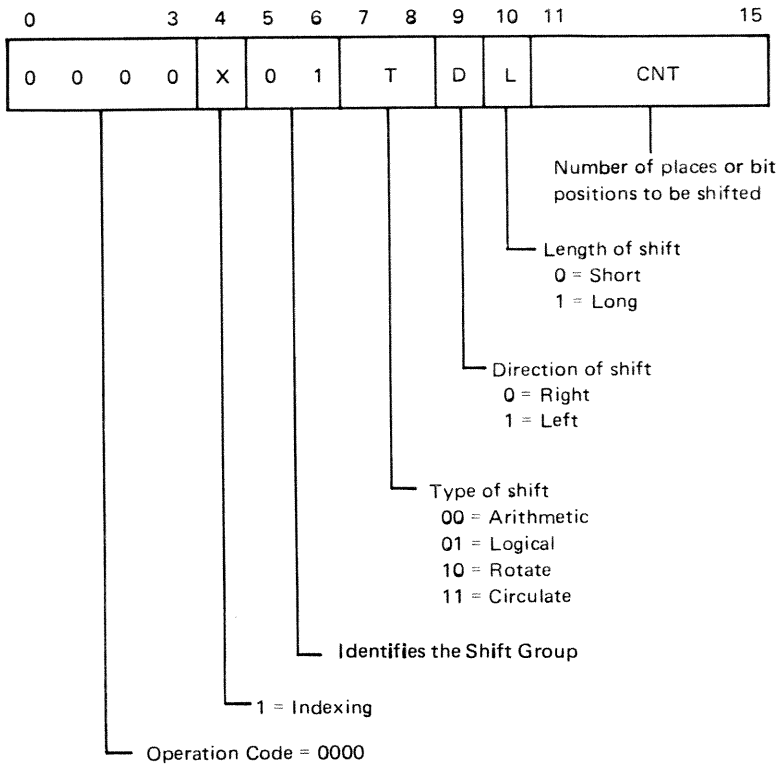
This format is used for instructions which fall into the categories of (a) Privileged, (b) Shift, (c) Operate, or (d) Non-Privileged. The general Extended Op Code format is shown below and is followed by a breakdown of each of the above categories.



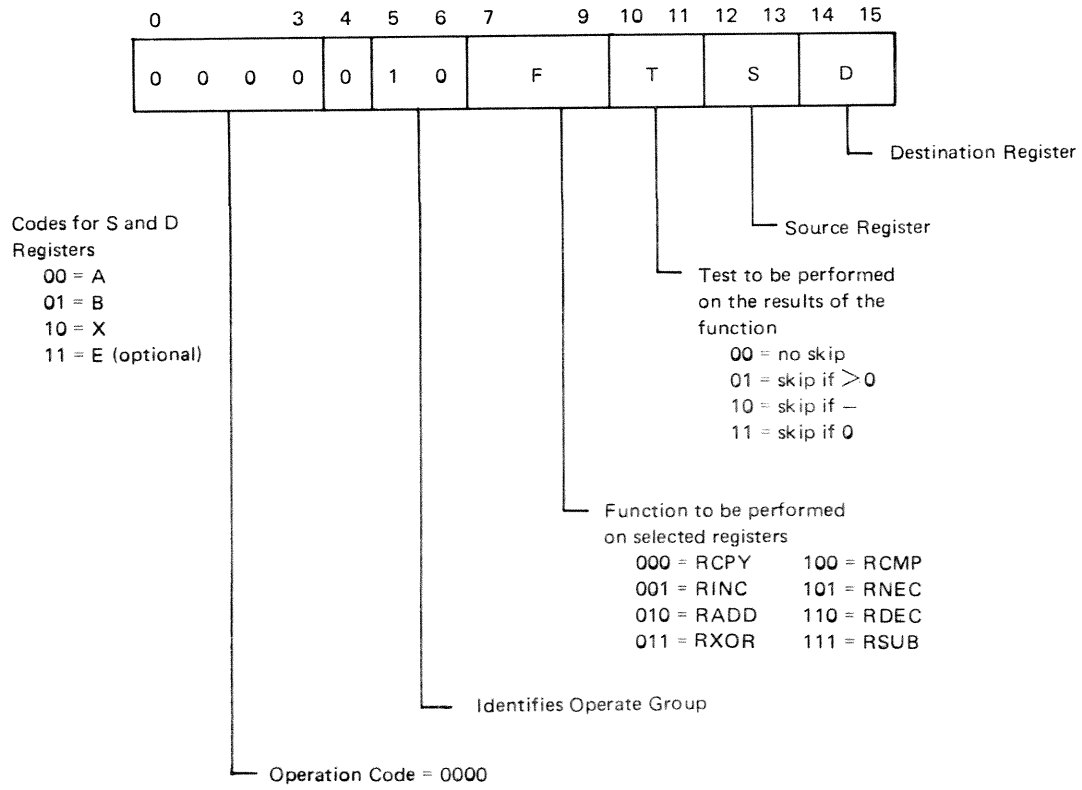
a. Privileged



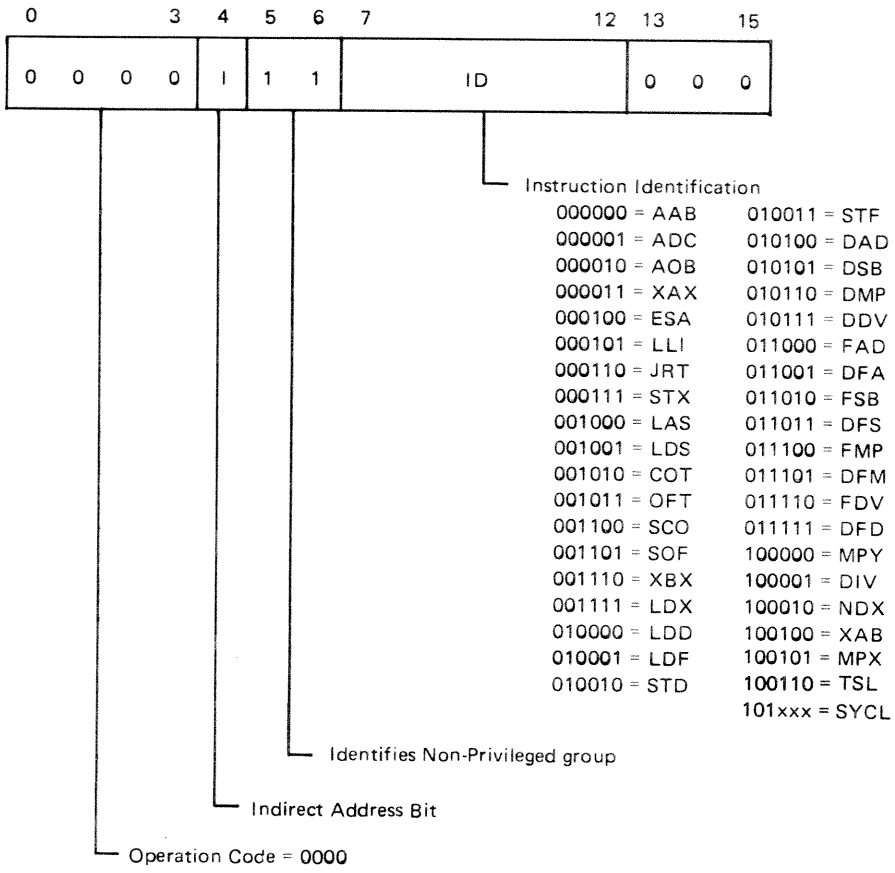
b. Shift



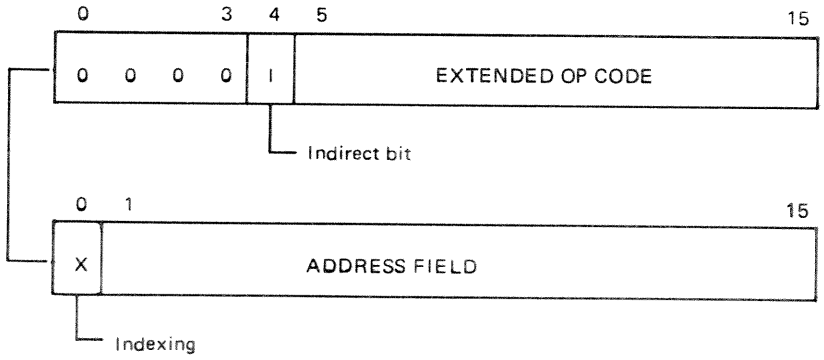
c. Operate



d. Non-Privileged



Double Word



Certain instructions require two words. The first word is in the Extended Op Code format and the second has the format of an indirect address. There is no relative bit, because the 15-bit address of the second word makes relative addressing unnecessary.

The instructions may be indexed and directly addressed. Setting bit 0 of the second word specifies pre-indexing. If indirect addressing (bit 4) is specified, then the second word becomes a pointer (which may be indexed) and the indirect address will be post-indexed if bit 0 of the indirectly addressed location is also set.

Depending on the type of instruction, therefore, the second word will contain an address, a pointer, or data. For instance:

1. The optional arithmetic instructions require address information in the second word.
2. Input/output control commands will contain control information for the channels and devices unless indirect addressing is specified.

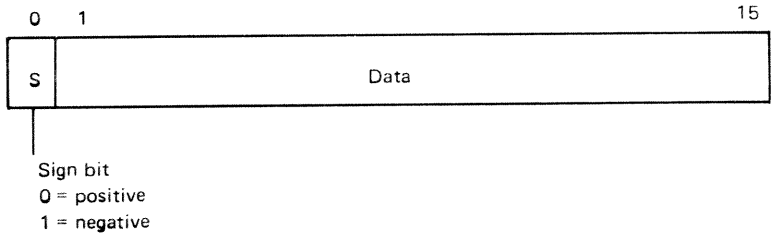
The address portion of this type of instruction and the word to which it points are assembled in PAR statements. One level of indirect addressing and two levels of indexing may be specified.

DATA FORMATS

Four types of formats are used to represent numerical data: (1) Integer, (2) Double Precision, (3) Floating Point (Short), and (4) Double Floating Point (Long).

1. Integer

The basic data format is a 16-bit binary integer with the sign located in bit 0.

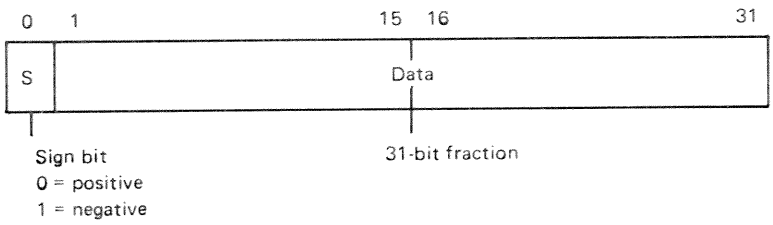


The number represented is defined as a binary or a hexadecimal integer, with bit 1 being the most significant position and bit 15 the least significant.

The maximum range of signed integers which may be represented by a single word is $8000_{16} \leq i \leq 7FFF_{16}$ (decimal: $-32768 \leq i \leq +32767$). Negative quantities are expressed in two's complement form. (The two's complement of a number is obtained by inverting each bit of the binary number and adding one.)

2. Double Precision

Double precision arithmetic utilizes two machine words to represent a 31-bit, signed binary fraction with the following format:

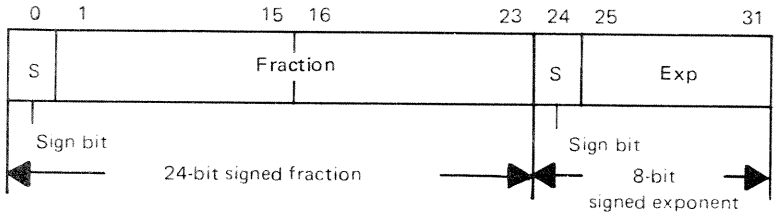


Numbers up to 2^{31} may be represented in this format.

The minimum range is $80000000_{16} \leq i \leq 7FFFFFFF_{16}$ (decimal: -1.0000000000 $\leq i \leq$ 0.9999999996). Negative double precision numbers are represented in two's complement form.

3. Floating Point (Short)

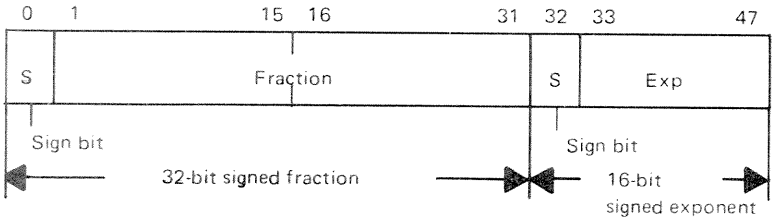
The short floating point format is one of two optional formats available for floating point numbers on the SCC 4700. The short format uses two machine words to represent the floating point number:



The fraction occupies bits 0-23 and the exponent uses bits 24-31. The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit.

4. Double Floating Point (Long)

The double floating point format is the second optional format available for expressing a floating point number on the SCC 4700. The long format utilizes three machine words in the following format:



The fraction occupies bits 0-31 and the exponent uses bits 32-47. The radix point of the fraction is assumed to be immediately to the right of the sign bit (bit 0).

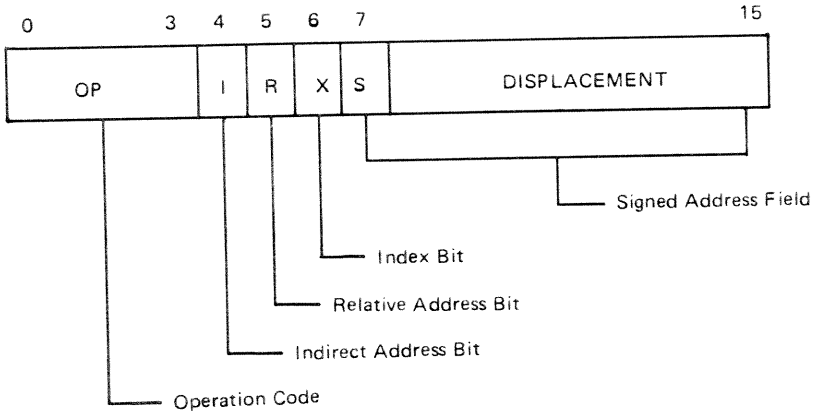
ADDRESSING

Address modification in the SCC 4700 is based on two concepts:

Primary Address: The intermediate address which is determined before indirect addressing and post-indexing are applied. It becomes the effective address if indirect addressing is not applied.

Effective Address: The final address which is formed after all address modification and indexing have been performed.

Modes of Addressing



There are five possible modes available for instructions of basic format, each of which results in a different effective address when implemented either singly or in combination. They are:

1. **Direct (Bit 5 = 0)**

   The primary address is determined by the address field of the instruction. In the direct mode, the primary address always refers to the first 512 locations of memory, unless indexed.

2. **Relative (Bit 5 = 1)**

   The primary address is the sum of the address field with sign extended of the instruction and the contents of the location counter; i.e., $(L) \pm (INS)_{8-15}$. (For byte addressing: $2(L) \pm (INS)_{8-15}$.)

3. **Primary Indexed (Bit 6 = 1)**

   Indexing may be applied to the primary address to form the primary indexed address.

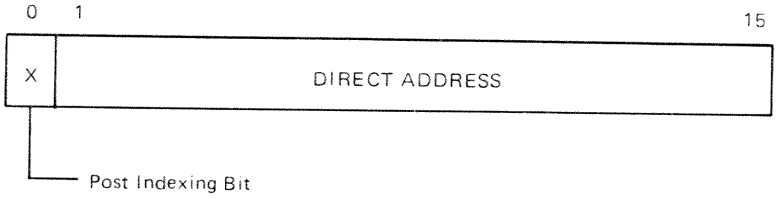
   Direct Indexed — The index register becomes a base register and the effective address is $(X) + (INS)_{7-15}$.

   Relative Indexed — The index register is added to the relative address; i.e., $[(L) \pm (INS)_{8-15}] + (X)$. (For byte addressing: $[2(L) \pm (INS)_{8-15}] + (X)$.)

4. **Indirect (Bit 4 = 1)**

   The indirect address bit is always applied after the contents of the primary address have been obtained. If the indirect address bit is zero, the contents of the location specified by the primary address are used as the operand of the instruction.

If the indirect address bit is a one, the contents of the location specified by the primary address is interpreted not as an operand, but as a 15-bit operand address. (Bit 0 of the indirectly addressed location is tested for post-indexing). Indirect addressing requires one additional memory cycle (920 nanoseconds) on all instructions.



If the primary address of a basic instruction is the current location plus one, the location counter is incremented to skip (L+2) the next word in the instruction sequence. In this way, both operands and full addresses may be included "in-line". This indirect address technique makes possible addressing up to 32K ($7FFF_{16}$).

5. **Post-Indexed (Bit 0 = 1)**

In this mode, bit 0 is checked after the contents of the indirectly addressed location specified by the basic instruction are obtained. If bit 0 is equal to one, the contents of the index register are added to the other 15 bits to form the effective operand address.

Figure 4 illustrates the flow of an instruction through the basic modes of addressing.

Byte Addressing

Address modification of instructions that are byte addressable (such as LDH, Load Halfword) is accomplished in the same manner as word-oriented instructions with the following exceptions:

1. Post-indexing is not permitted.
2. The range (in the number of words) is half that of word-oriented instructions unless indirect addresses are used. In this case, the full 16 bits are used as the byte address.
3. The location counter is treated as a word address while the index register is considered a byte address.

All conversion of the byte address to align the data in the proper half of the word is performed automatically by the CPU.

The even numbered byte locations will be contained in the most significant eight bits of each word. For example:

Location	Byte Address	
0	0	1
1	2	3
2	4	5

Figure 5 is a flow diagram of the halfword addressing mode.

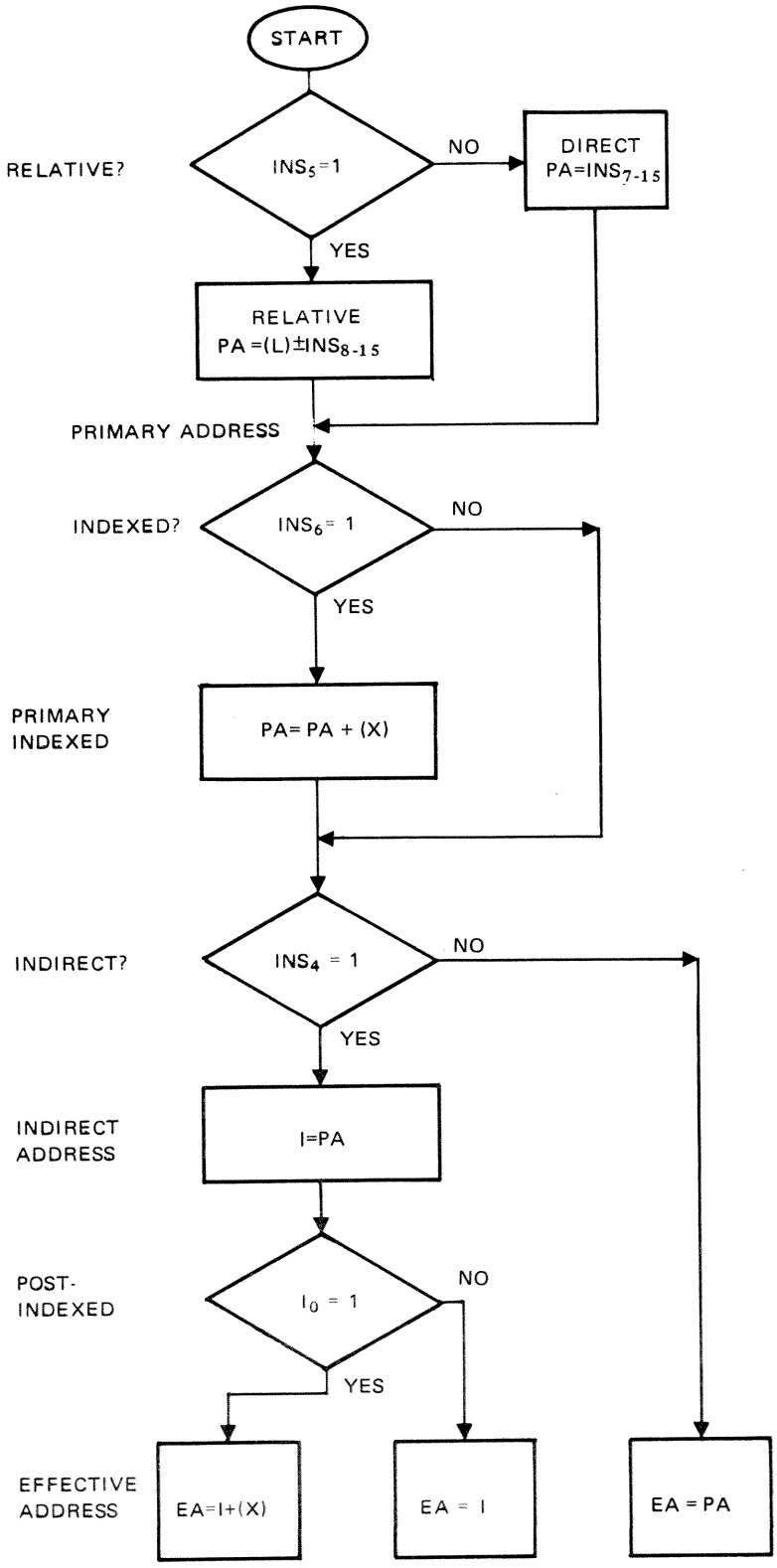


Figure 4. Basic Addressing Mode Flow Diagram

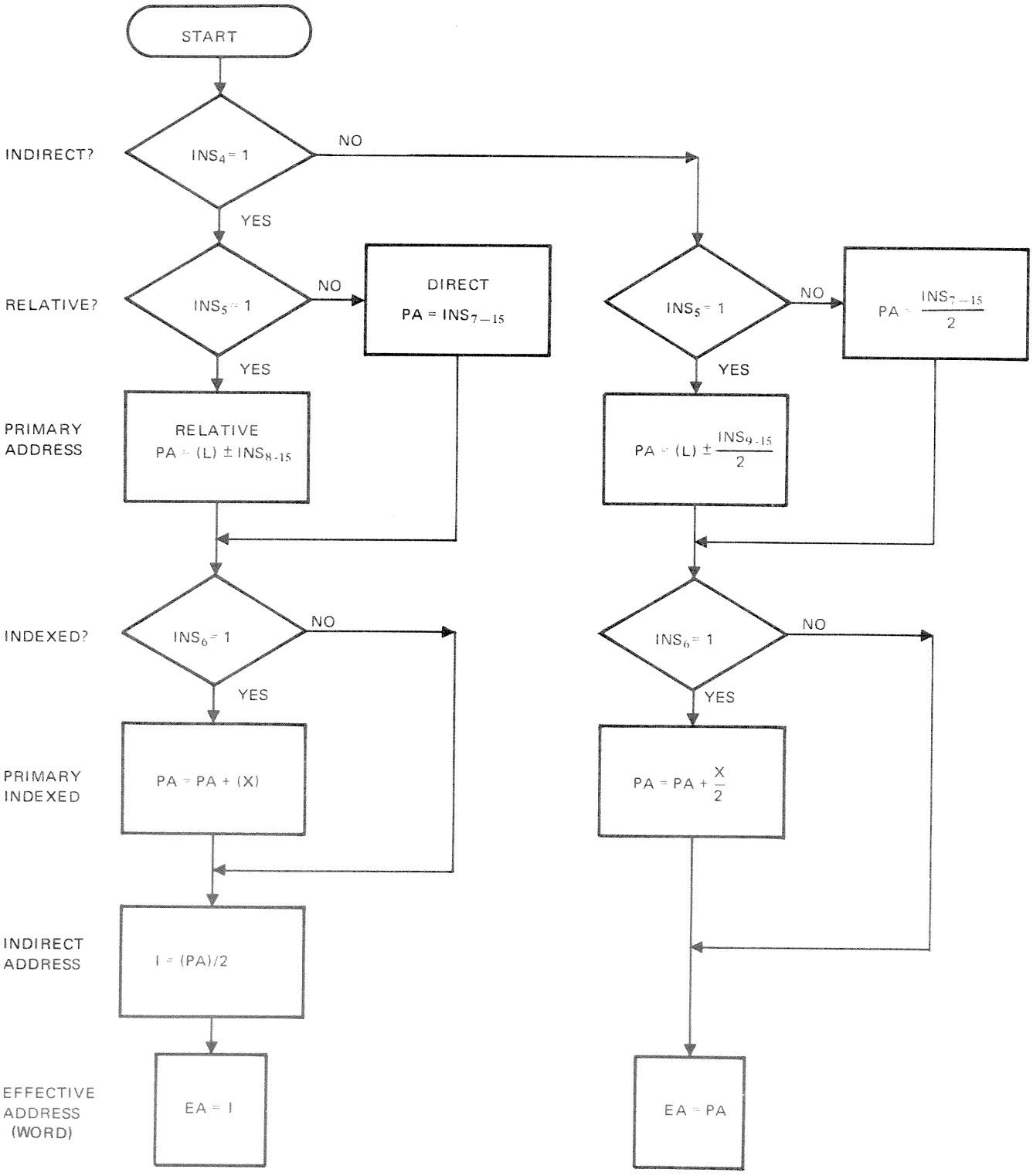


Figure 5. Halfword Addressing Mode Flow Diagram

INSTRUCTION SET

This section describes the function and format of each instruction available with the SCC 4700. The optional instruction set consists of instructions which are classified as follows:

Multiply–Divide

Double Precision

Floating Point

Double Floating Point

Note: The conventions listed at the beginning of this chapter will be used extensively in the remainder of this section and therefore should be reviewed.

General Operation Procedures

The location counter will normally be incremented by one for basic and extended operation code instructions; however, if the primary address of a basic instruction is L + 1, (L + 2 or L + 3 for halfword instructions) the location counter will be incremented by two, skipping the location containing the indirect address or operand.

Double word instructions will increment the location counter by two to skip the address field unless otherwise noted.

Skip instructions increment the location counter as indicated, depending on the result of the test of the contents of the location or indicator.

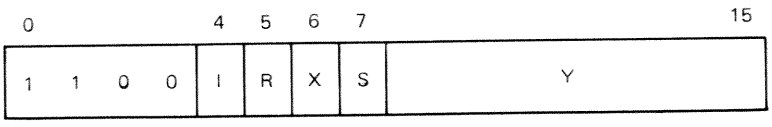
Indirect addressing will be allowed only where indicated by "I" in bit position 4.

Pre-indexing is allowed when indicated by an "X" in the instruction format. In double word instructions, this will be bit 0 of the second word. Post-indexing is available only when indirect addressing is specified and will be indicated by setting (=1) bit 0 of the indirect address.

Typical instruction execution time is indicated for the basic SCC 4700 without memory map or memory gate.

The address mode of the instruction is determined by the setting of the I, R, and X bits. For example:

LDA, Load Accumulator

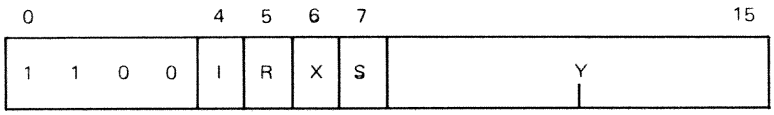


Bits 4-7	Address Mode
0000	Load A Direct
0010	Load A Direct Indexed
0100	Load A Relative
0110	Load A Relative Indexed
1000	Load A Indirect from Direct Address
1010	Load A Indirect from Indexed Direct Address
1100	Load A Indirect from Relative Address
1110	Load A Indirect from Relative Indexed Address

When the basic instruction format is implemented, the effective address is formed. Post-indexing is implemented when bit 0 of the indirectly addressed word is set.

LOAD AND STORE INSTRUCTIONS

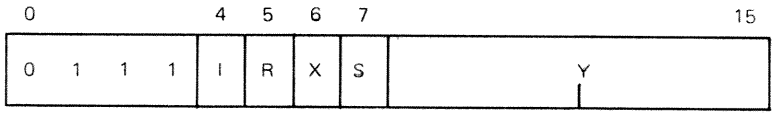
LDA — LOAD ACCUMULATOR — 1.84 μsec



(EA) → A

The contents of the effective address are copied into the accumulator. The contents of the location specified by the effective address are unchanged.

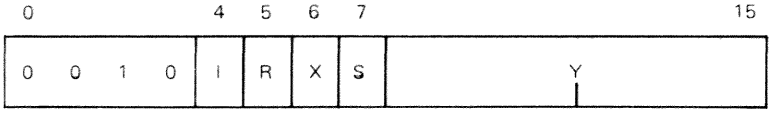
STA — STORE ACCUMULATOR — 1.84 μsec



(A) → EA

The contents of the accumulator are copied into the memory location specified by the effective address. The contents of the accumulator are unchanged.

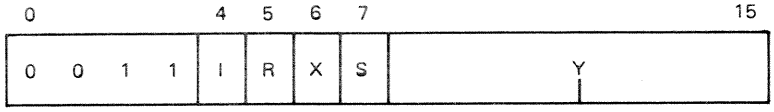
LDB — LOAD B REGISTER — 1.84 μsec



(EA) → B

The contents of the effective address are copied into the B register. The contents of the location specified by the effective address are unchanged.

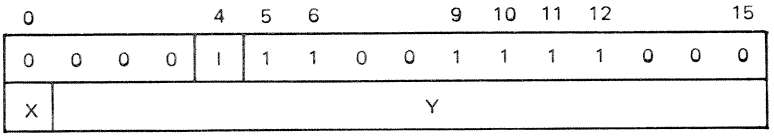
STB STORE B REGISTER **1.84 μsec**



(B) → EA

The contents of the B register are copied into the memory location specified by the effective address. The contents of the B register are unchanged.

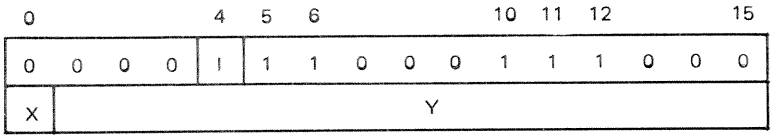
LDX LOAD INDEX **2.81 μsec**



(EA) → X

The contents of the effective address are copied into the index register. The contents of the memory location specified by the effective address are unchanged.

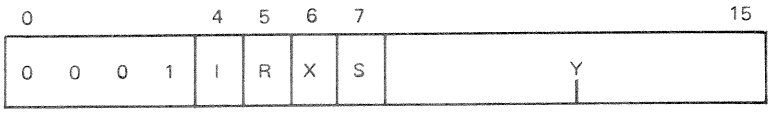
STX STORE INDEX **2.81 μsec**



(X) → EA

The contents of the index register are copied into the memory location specified by the effective address. The contents of the index register are unchanged.

LDH LOAD HALFWORD **2.59 μsec**
3.01 μsec**



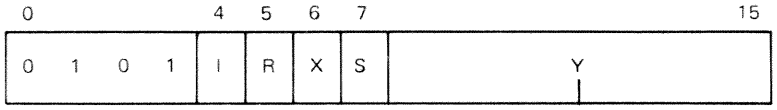
$0 \rightarrow A_{0-7}$; (EA) $\rightarrow A_{8-15}$

Bits 0-7 of the accumulator are set to zero and the contents of the effective address (specified as a byte address) are loaded into bits 8-15 of the accumulator. The contents of the location specified by the effective address are unchanged.

Post-indexing is not permitted.

**Indirect Address Time.

STH — STORE HALFWORD — 2.89 μsec / 3.31 μsec**

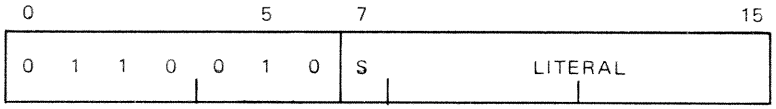


$(A)_{8-15} \rightarrow EA$

The contents of bits 8-15 of the accumulator are copied into the effective address (specified as a byte address). The contents of the accumulator and the other half of the effective address word are unchanged.

Post-indexing is not permitted.

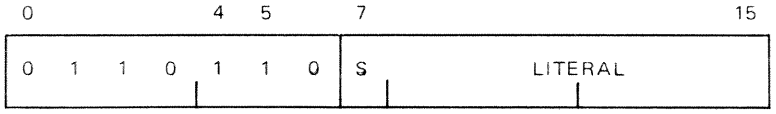
LDL — LOAD A, LITERAL — 1.05 μsec



$(INS)_7 \rightarrow A_{0-7}$
$(INS)_{8-15} \rightarrow A_{8-15}$

The sign (bit 7) is copied into bits 0-7 of the accumulator. Bits 8-15 of the instruction are copied into bits 8-15 of the accumulator.

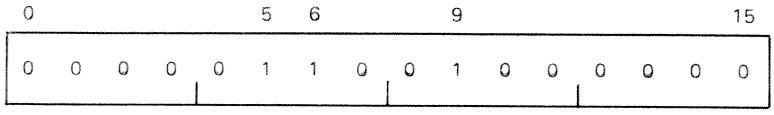
LDLB — LOAD B, LITERAL — 1.05 μsec



$(INS)_7 \rightarrow B_{0-7}$
$(INS)_{8-15} \rightarrow B_{8-15}$

The sign bit (bit 7) is copied and extended into bits 0-7 of the B register. Bits 8-15 of the instruction are copied into bits 8-15 of the B register.

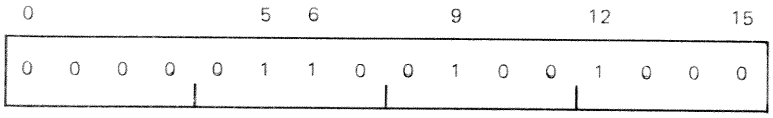
LAS — LOAD ACCUMULATOR FROM SWITCHES — 1.00 μsec



(Switches) → A

The contents of the switch register are placed into the accumulator. Data may be entered in the switch register by the operator using the control panel. The contents of the switch register are unchanged.

LDS — LOAD STATUS — 1.00 μsec

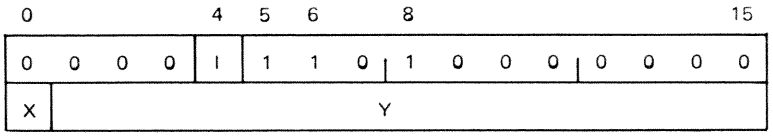


(ST) → A

**Indirect Address Time.

The contents of the status register are copied into the accumulator. The status register is a composite register which includes the carryout, overflow, halt, mode, indirect mode, and interrupts disabled indicators. The contents of the status register are not changed.

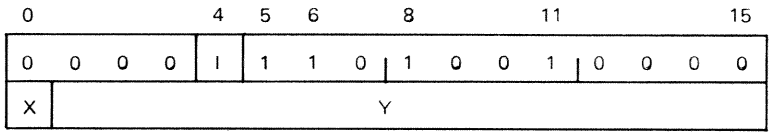
LDD* **LOAD DOUBLE** **3.68 μsec**



(EA) → A; (EA + 1) → B

The contents of the effective address are copied into the accumulator. The contents of the effective address plus one are copied into the B register. The contents of memory are unchanged.

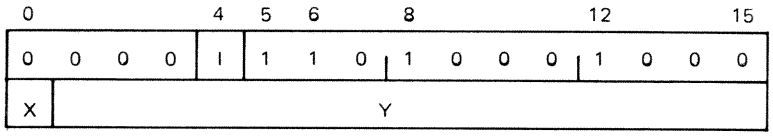
STD* **STORE DOUBLE** **3.68 μsec**



(A) → EA; (B) → EA + 1

The contents of the accumulator are copied into the effective address. The contents of the B register are copied into the effective address plus one. The contents of the registers are unchanged.

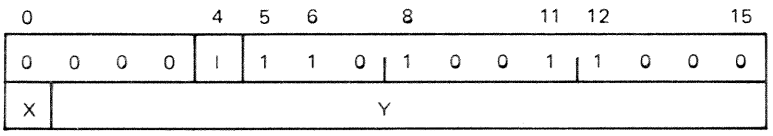
LDF* **LOAD FLOATING** **4.60 μsec**



(EA) → A; (EA + 1) → B; (EA + 2) → E

The contents of the effective address are copied into the accumulator. The contents of the effective address plus one are placed in the B register. The contents of the effective address plus two are transferred to the E register. The contents of memory are unchanged.

STF* **STORE FLOATING** **4.60 μsec**

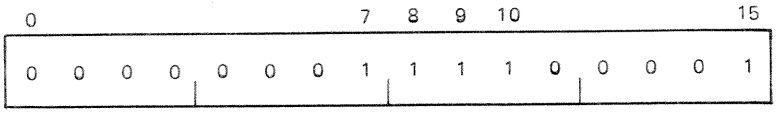


(A) → EA; (B) → EA + 1; (E) → EA + 2

The contents of the accumulator are placed into the effective address. The contents of the B register are copied into the effective address plus one. The contents of the E register are transferred into the effective address plus two. The contents of the registers are unchanged.

*Optional.

LSMP* LOAD SYSTEM MAP POINTER 1.95 μsec

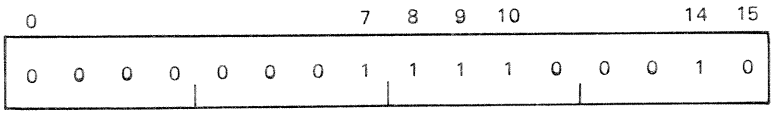


α_s → SMTP register
0 → Associative register

LSMP is a privileged instruction which loads the starting address (α) of the system map table from the accumulator into the SMTP register and clears the associative registers. Note: The starting address must be a real core address.

The location counter is incremented by 2, skipping the next instruction.

LUMP* LOAD USER MAP POINTER 1.95 μsec



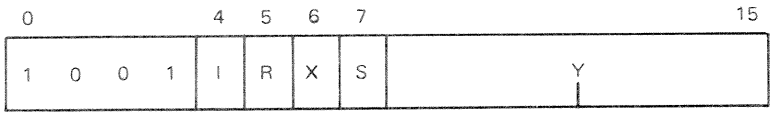
α_u → UMTP register
0 → Associative register

LUMP is a privileged instruction which loads the starting address (α) of the user map table from the accumulator into the UMTP register and clears the associative registers. Note: The starting address must be a real core address.

The location counter is incremented by two, skipping the next instruction.

ARITHMETIC INSTRUCTIONS

ADD ADD TO ACCUMULATOR 1.84 μsec

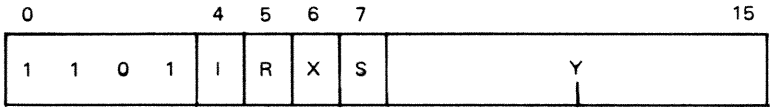


(A) + (EA) → A

The contents of the effective address (16-bit operand) are added to the current value of the accumulator and the result is placed in the accumulator. The carryout indicator is set by the carry from bit position 0. If the result is greater than the maximum size of the register, the overflow indicator is set. If both numbers have the same sign, but the sign of the result is different, an overflow has occurred. If overflow does not occur, the overflow indicator is not altered.

*Optional.

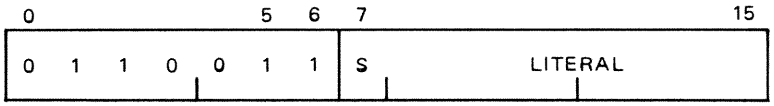
SUB SUBTRACT FROM ACCUMULATOR 1.84 μsec



(A) − (EA) → A

The contents of the effective address (16-bit operand) are subtracted from the current value of the accumulator and the result is placed in the accumulator. The carryout indicator is set by the carry from bit position 0. If the result is greater than the maximum size of the register, the overflow indicator is set. If overflow does not occur, the overflow indicator is not altered. If the numbers have different signs, and the sign of the result equals the sign of the memory operand, an overflow has occurred.

ADL ADD TO A, LITERAL 1.15 μsec

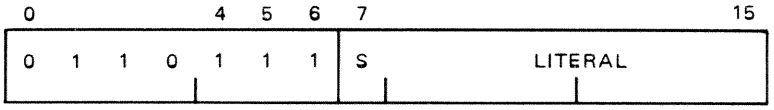


$(INS)_7 \rightarrow INS_{0-6}$; (INS) + (A) → A

This instruction provides a convenient way to add or subtract quantities in the range $-256 \leq X \leq +255$. The last nine bits (7-15) of this instruction are interpreted as a 9-bit, two's complement number.

Bit 7 of the instruction is extended through bits 0-6, converting the 9-bit operand to a 16-bit, two's complement operand which is added to the contents of the accumulator. The carryout indicator is set by the carryout of bit position zero. If overflow does not occur, the overflow indicator is not altered.

ADLB ADD TO B, LITERAL 1.15 μsec

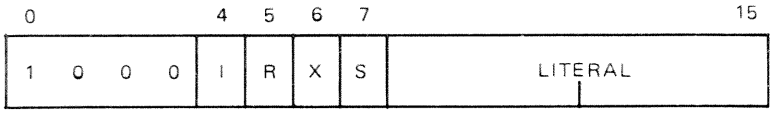


$(INS)_7 \rightarrow INS_{0-6}$; (INS) + (B) → B

This instruction provides a convenient way to add or subtract quantities in the range $-256 \leq X \leq +255$. The last nine bits (7-15) of this instruction are interpreted as a 9-bit, two's complement number.

Bit 7 of the instruction is extended through 0-6, converting the 9-bit operand to a 16-bit, two's complement operand which is added to the contents of the B register. The carryout indicator is set by the carryout of bit position zero. If overflow does not occur, the overflow indicator is not altered.

MIN **MEMORY INCREMENT; SKIP ON ZERO** **2.14 μsec**

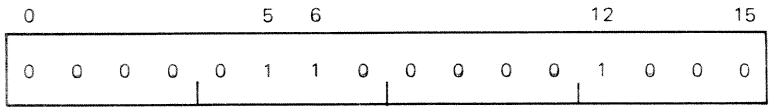


(EA) + 1 → EA
If (EA) = 0, (L) + 2 → L or, if PA = L + 1, (L) + 3 → L.
If (EA) ≠ 0, (L) + 1 → L or, if PA = L + 1, (L) + 2 → L.

One is added to the contents of the memory location at the effective address. The result is tested, and if the value equals zero, the location counter is incremented by two. If the value is not zero, the location counter is incremented by one. The carryout and overflow indicators are not affected.

If the primary address is L+1, the location counter will be incremented by two if (EA) ≠ 0; or by three if (EA) = 0.

ADC **ADD CARRY** **1.25 μsec**

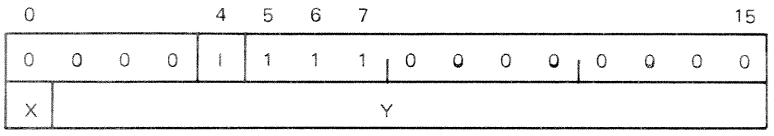


(A) + (CRO) → A

The carryout indicator is added to the accumulator and the result is placed in the accumulator.

The carryout indicator is set by the carryout of bit zero. If the results exceed the size of the accumulator, the overflow indicator is set. If overflow does not occur, the overflow indicator is not altered.

MPY* **MULTIPLY** **8.44 μsec**



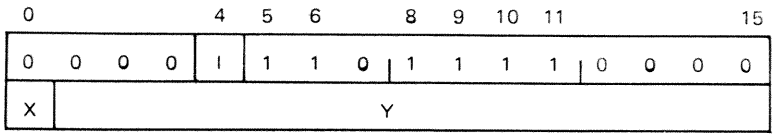
(A) * (EA) → B, A

This operation is a single-precision integer multiplication, which leaves the most significant half of the 32-bit product in the B register and the least significant half in the A register. The multiplicand is placed in the accumulator before the instruction is executed. The multiplier is the contents of the effective address of the double word multiply instruction. The multiplier and multiplicand must be right-adjusted. The product will be right adjusted in the B and A registers after execution. Overflow is not affected; but the carryout indicator may be set if carryout occurs on the last operation performed. The contents of memory at the location specified by the effective address remain unchanged.

*Optional.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

FDV* **FLOATING POINT DIVIDE** **37.16 μsec (min)**
44.41 μsec (max)



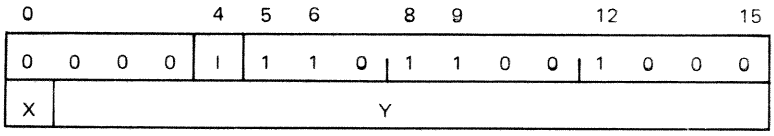
(A, B)/(EA, EA+1) → A, B

This instruction is to be used with the short (or two word) floating point format. The contents of the effective address (EA) and (EA) + 1 are subtracted from the A and B registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

Both numbers must be normalized ($A_0 \neq A_1$) before division; the answer will be normalized when returned. Rounding will be performed on the 25th bit of the answer, before combining with the exponent.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur. The carryout is not significant.

DFA* **DOUBLE FLOATING ADD** **7.88 μsec (min)**
9.03 μsec (max)**

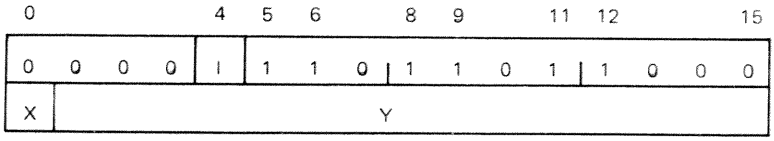


(A, B, E) + (EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2 is added to the floating point quantity in the A, B, and E registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFS* **DOUBLE FLOATING SUBTRACT** **7.88 μsec (min)**
9.03 μsec (max)**



(A, B, E) − (EA, EA+1, EA+2) → A, B, E

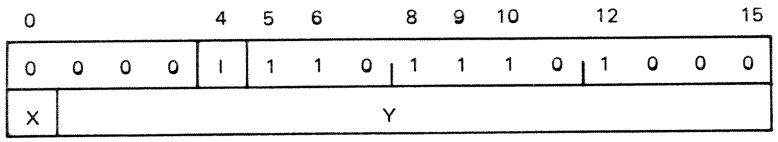
*Optional.

** +.25N for alignment
+.55N for normalize.

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2 is subtracted from the floating point quantity in the A, B, and E registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFM* **DOUBLE FLOATING MULTIPLY** **26.28 μsec (min)**
27.63 μsec (max) **



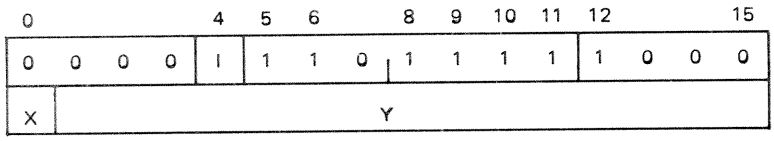
(A, B, E) * (EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1 and (EA) + 2 is multiplied times the floating point quantity in the A, B, and E registers. The product replaces the multiplicand in the A, B, and E registers.

Both numbers must be normalized ($A_0 \neq A_1$) before multiplication; the normalized answer will be returned.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFD* **DOUBLE FLOATING DIVIDE** **34.68 μsec (min)**
39.58 μsec (max) **



(A, B, E)/(EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity in the A, B, and E registers is divided by the floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2. The quotient replaces the dividend in the A, B, and E registers.

Both numbers must be normalized ($A_0 \neq A_1$) before division; the answer will be normalized when returned.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

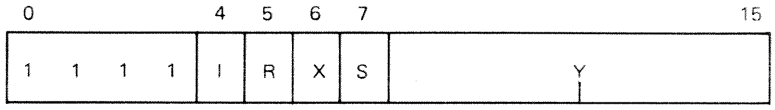
---

*Optional.

**+.21N for alignment.

LOGICAL INSTRUCTIONS

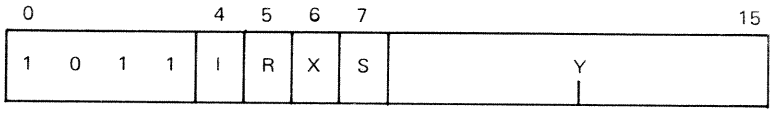
AND — AND MEMORY WITH ACCUMULATOR — 1.84 μsec



(A) ∩ (EA) → A

The contents of the effective address are ANDed with the contents of the accumulator. The result is placed in the accumulator. The contents of the memory location remain unchanged.

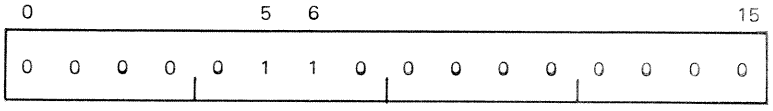
XOR — EXCLUSIVE OR WITH ACCUMULATOR — 1.84 μsec



(A) ⊕ (EA) → A

The contents of the effective address are Exclusive ORed with the contents of the accumulator. The result is placed in the accumulator. The memory location is unchanged.

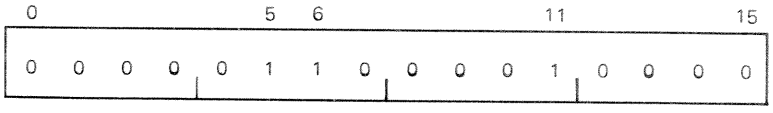
AAB — AND ACCUMULATOR WITH B REGISTER — 1.15 μsec



(A) ∩ (B) → A

The contents of the accumulator are ANDed with the contents of the B register. The result is placed in the accumulator. The contents of the B register are unchanged.

AOB — OR ACCUMULATOR WITH B REGISTER — 1.15 μsec



(A) ∪ (B) → A

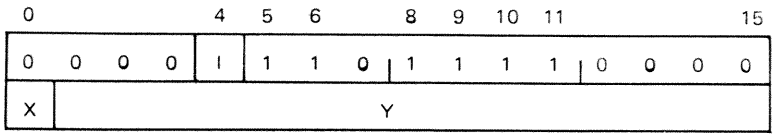
The contents of the accumulator are ORed with the contents of the B register. The result is placed in the accumulator. The contents of the B register are unchanged.

---

*Optional.
** For alignment +.21N

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

FDV* **FLOATING POINT DIVIDE** **37.16 μsec (min)**
44.41 μsec (max)



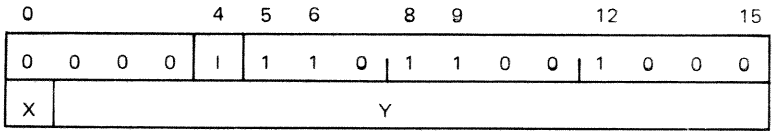
(A, B)/(EA, EA+1) → A, B

This instruction is to be used with the short (or two word) floating point format. The contents of the effective address (EA) and (EA) + 1 are subtracted from the A and B registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

Both numbers must be normalized ($A_0 \neq A_1$) before division; the answer will be normalized when returned. Rounding will be performed on the 25th bit of the answer, before combining with the exponent.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur. The carryout is not significant.

DFA* **DOUBLE FLOATING ADD** **7.88 μsec (min)**
9.03 μsec (max)**

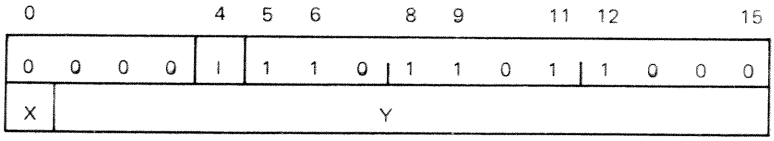


(A, B, E) + (EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2 is added to the floating point quantity in the A, B, and E registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFS* **DOUBLE FLOATING SUBTRACT** **7.88 μsec (min)**
9.03 μsec (max)**



(A, B, E) − (EA, EA+1, EA+2) → A, B, E

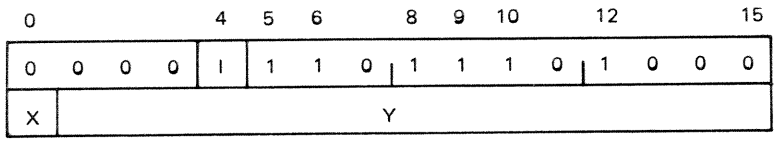
*Optional.

** +.25N for alignment
+.55N for normalize.

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2 is subtracted from the floating point quantity in the A, B, and E registers after alignment of the binary point. (Alignment occurs when the exponents of both numbers are set equal.)

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFM* **DOUBLE FLOATING MULTIPLY** **26.28 μsec (min)**
27.63 μsec (max) **



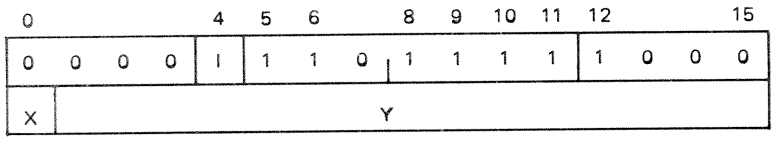
(A, B, E) * (EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity at the effective address (EA), (EA) + 1 and (EA) + 2 is multiplied times the floating point quantity in the A, B, and E registers. The product replaces the multiplicand in the A, B, and E registers.

Both numbers must be normalized ($A_0 \neq A_1$) before multiplication; the normalized answer will be returned.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

DFD* **DOUBLE FLOATING DIVIDE** **34.68 μsec (min)**
39.58 μsec (max) **



(A, B, E)/(EA, EA+1, EA+2) → A, B, E

This instruction is to be used with the long (or three word) floating point format. The floating point quantity in the A, B, and E registers is divided by the floating point quantity at the effective address (EA), (EA) + 1, and (EA) + 2. The quotient replaces the dividend in the A, B, and E registers.

Both numbers must be normalized ($A_0 \neq A_1$) before division; the answer will be normalized when returned.

If exponent overflow or underflow occurs, the system trap indicator will be set and a system trap will occur.

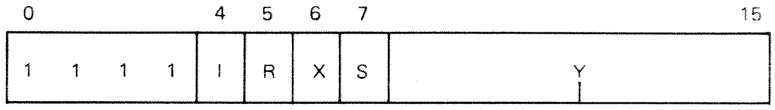
---

*Optional.

**+.21N for alignment.

LOGICAL INSTRUCTIONS

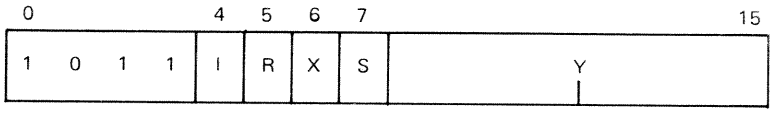
AND — AND MEMORY WITH ACCUMULATOR — 1.84 μsec



(A) ∩ (EA) → A

The contents of the effective address are ANDed with the contents of the accumulator. The result is placed in the accumulator. The contents of the memory location remain unchanged.

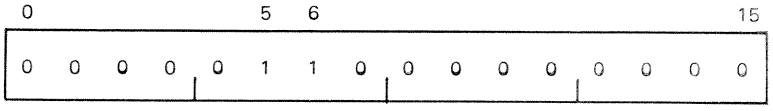
XOR — EXCLUSIVE OR WITH ACCUMULATOR — 1.84 μsec



(A) ⊕ (EA) → A

The contents of the effective address are Exclusive ORed with the contents of the accumulator. The result is placed in the accumulator. The memory location is unchanged.

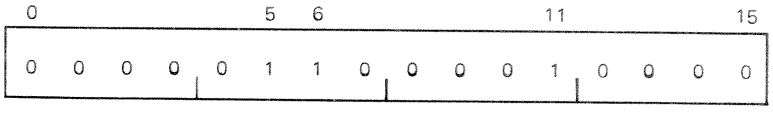
AAB — AND ACCUMULATOR WITH B REGISTER — 1.15 μsec



(A) ∩ (B) → A

The contents of the accumulator are ANDed with the contents of the B register. The result is placed in the accumulator. The contents of the B register are unchanged.

AOB — OR ACCUMULATOR WITH B REGISTER — 1.15 μsec



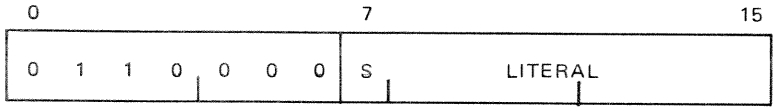
(A) ∪ (B) → A

The contents of the accumulator are ORed with the contents of the B register. The result is placed in the accumulator. The contents of the B register are unchanged.

---

*Optional.
** For alignment +.21N

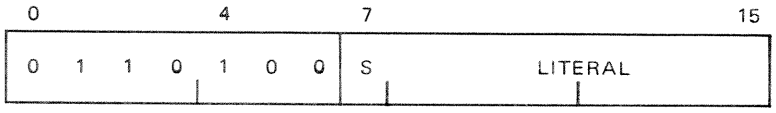
ANL **AND THE ACCUMULATOR, LITERAL** **1.15 μsec**



$(INS)_7 \rightarrow INS_{0-6}$; $(INS) \cap (A) \rightarrow A$

Bit 7 of the instruction is extended through bits 0-6. The resulting 16-bit operand is ANDed with the contents of the accumulator. The result is placed in the accumulator.

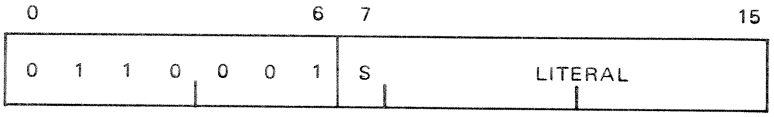
ANLB **AND THE B REGISTER, LITERAL** **1.15 μsec**



$(INS)_7 \rightarrow INS_{0-6}$; $(INS) \cap (B) \rightarrow B$

Bit 7 of the instruction is extended through bits 0-6. The resulting 16-bit operand is ANDed with the contents of the B register. The result is placed in the B register.

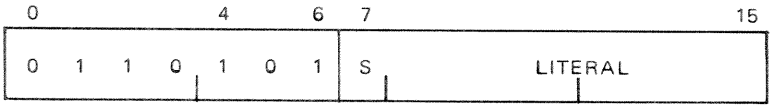
XOL **EXCLUSIVE OR THE ACCUMULATOR, LITERAL** **1.15 μsec**



$(INS)_7 \rightarrow INS_{0-6}$; $(INS) \oplus (A) \rightarrow A$

Bit 7 of the instruction is extended through bits 0-6, and the 16-bit operand is Exclusive ORed with the contents of the accumulator. The result is placed in the accumulator.

XOLB **EXCLUSIVE OR THE B REGISTER, LITERAL** **1.15 μsec**

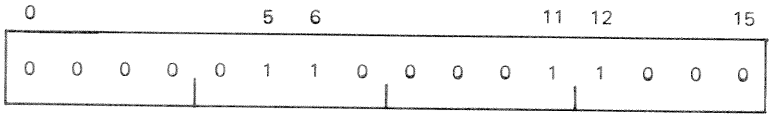


$(INS)_7 \rightarrow INS_{0-6}$; $(INS) \oplus (B) \rightarrow B$

Bit 7 of the instruction is extended through bits 0-6, and the 16-bit operand is Exclusive ORed with the contents of the B register. The result is placed in the B register.

REGISTER MANIPULATION INSTRUCTIONS

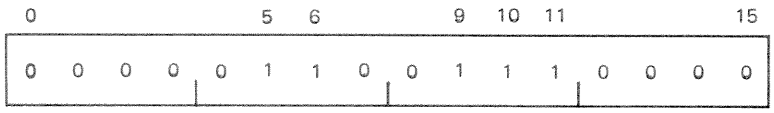
XAX **EXCHANGE ACCUMULATOR AND INDEX** **1.15 μsec**



$(A) \rightarrow X$; $(X) \rightarrow A$

The contents of the accumulator are placed in the index register and the contents of the index register are placed in the accumulator in a single operation.

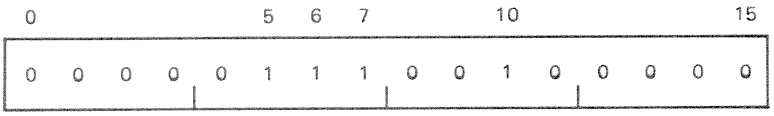
XBX **EXCHANGE B AND X** **1.15 μsec**



(B) → X; (X) → B

The contents of the B register are placed in the index register and the contents of the index register are placed in the B register in one operation.

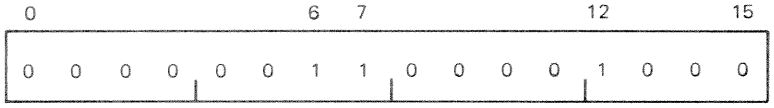
XAB **EXCHANGE A AND B** **1.15 μsec**



(A) → B, (B) → A

The contents of the accumulator are placed in the B register and the contents of the B register are placed in the accumulator in one operation.

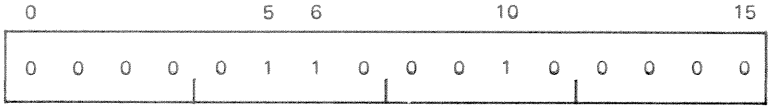
XHA **EXCHANGE HALVES OF A** **3.25 μsec**



$(A)_{0-7} \rightarrow A_{8-15}$
$(A)_{8-15} \rightarrow A_{0-7}$

The contents of bits 0-7 of the accumulator are placed in bits 8-15 and the contents of bits 8-15 are placed in bits 0-7 in one operation.

ESA **EXTEND SIGN OF ACCUMULATOR** **1.05 μsec**

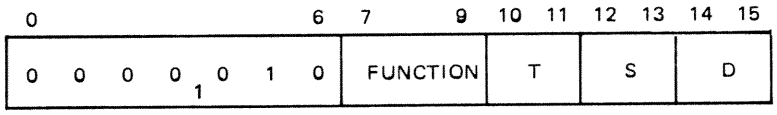


$(A)_0 \rightarrow B_{0-15}$

The sign (bit 0) of the accumulator is extended through the B register (B_{0-15}). The accumulator is unchanged.

OPERATE INSTRUCTIONS

This is a special group of instructions which performs inter-register manipulations. Arithmetic and logical functions can be performed on a register or between two registers, asynchronously of memory. Testing of the results is accomplished after the function is complete. (The test, if specified, does not require any additional time.) The register(s), function, and test to be performed are specified by the following format and codes.



The codes for the T, S, and D fields are as follows.

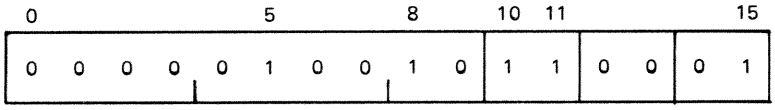
Bits 10–11 T specifies the test to be performed on the destination register after the indicated function has been performed.

00 = no skip
01 = skip if positive (>0)
10 = skip if negative
11 = skip if zero

Bits 12–13 S specifies the source register.
14–15 D specifies the destination register.

00 = A register
01 = B register
10 = X register
11 = E register (optional)*

Example: **RADD** **REGISTER ADD**



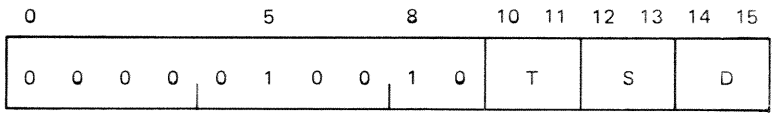
(A) + (B) → B, skip if B = 0

The contents of the A register are added to the contents of the B register; the result is placed in the B register. A skip is performed if the result is zero.

---

*This optional register is available only when floating point hardware has been implemented; however, the E register may be addressed. As a source, the E register will be all zeros; as a destination register, the results sent to the E register will be lost except for the test.

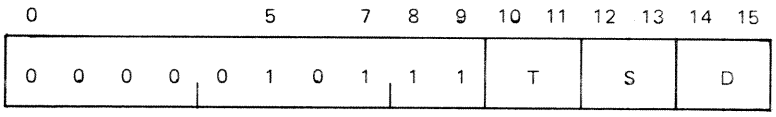
RADD REGISTER ADD 1.45 μsec



(S) + (D) → D

The contents of the source register are added to the contents of the destination register and the result is placed in the destination register. Carryout and overflow are not affected.

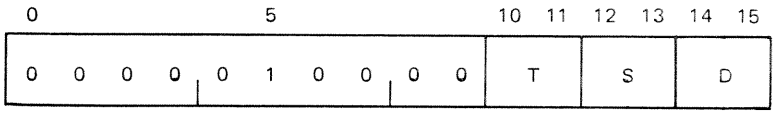
RSUB REGISTER SUBTRACT 1.45 μsec



(S) − (D) → D

The contents of the destination register are subtracted from the contents of the source register and the result is placed in the destination register. Carryout and overflow are not affected.

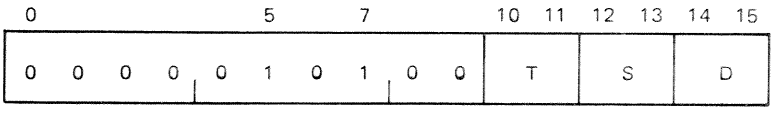
RCPY REGISTER COPY 1.25 μsec



(S) → D

The contents of the source register are transferred to the destination register.

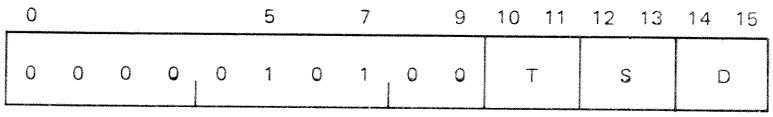
RCMP REGISTER COMPLEMENT 1.25 μsec



$(\overline{S}) \rightarrow D$

The one's complement of the contents of the source register are transferred to the destination register.

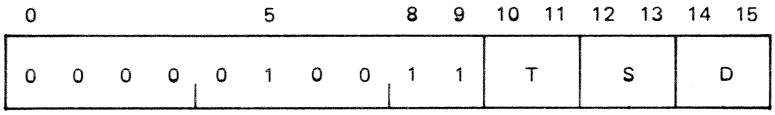
RNEG REGISTER NEGATE 1.25 μsec



$(\overline{S} + 1) \rightarrow D$

The two's complement of the contents of the source register are transferred to the destination register.

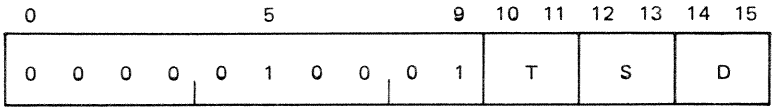
RXOR REGISTER EXCLUSIVE OR 1.45 μsec



(S) ⊕ (D) → D

The contents of the source register are Exclusive ORed with the contents of the destination register; the result is placed in the destination register.

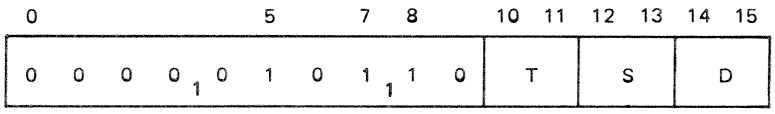
RINC REGISTER INCREMENT 1.25 μsec



(S) + 1 → D

One is added to the contents of the source register and the result is placed in the destination register. Carryout and overflow are not affected.

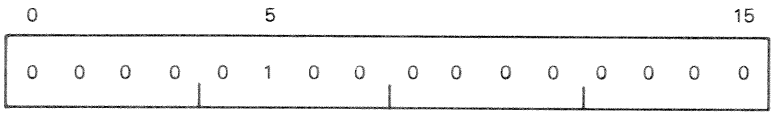
RDEC REGISTER DECREMENT 1.25 μsec



(S) − 1 → D

One is subtracted from the contents of the source register and the result is placed in the destination register. Carryout and overflow are not affected.

NOP NO OPERATION 1.25 μsec



(A) → A

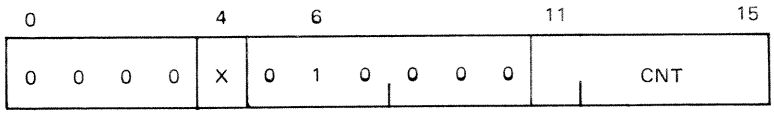
This instruction causes the machine to proceed to the next instruction without affecting any information.

SHIFT INSTRUCTIONS

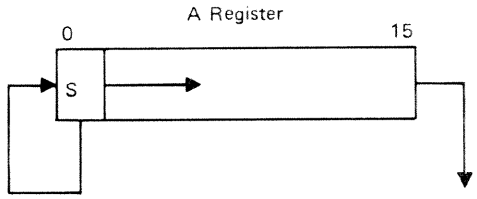
The shift instructions are conditional shifts; i.e., the number of binary positions to be shifted is determined by the value of CNT (the count field). Shifts may be made to the left or right and may be either "short" or "long". A "short" shift is performed on a single 16-bit word in the A register; a "long" shift is performed between the A and B registers on two 16-bit words.

Shifts may be indexed by setting bit 4 (=1) of the instruction, thus permitting a change in count, length, direction, and type of shift.

SAR — SHORT ARITHMETIC RIGHT SHIFT — 1.25 + .25N μsec



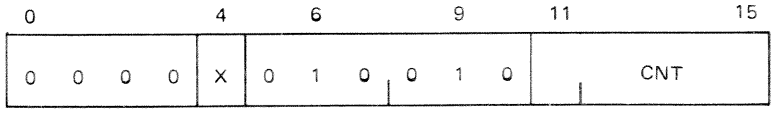
$(A)_0 \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)



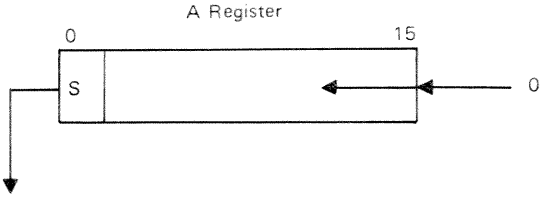
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the right the number of places designated by CNT ($0 \leq CNT < 32$). The sign bit, S, is propagated from one position to the next, CNT number of places. The data shifted out of A_{15} is lost.

SAL — SHORT ARITHMETIC LEFT SHIFT — 1.25 + .25N μsec



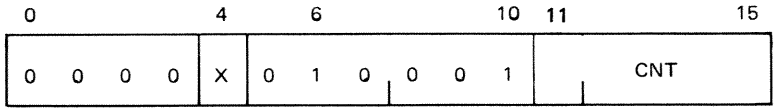
$0 \rightarrow A_{15}$
$(A)_i \rightarrow A_{i-1}$, I = 0 to CNT ($0 \leq CNT < 32$)



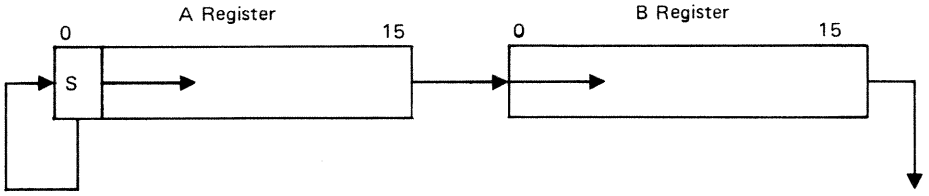
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the left the number of places designated by CNT ($0 \leq CNT < 32$). Zeros are inserted CNT number of places into A_{15}. The sign bit, S, is lost; however, the overflow indicator is set if the sign changes.

LAR — LONG ARITHMETIC RIGHT SHIFT — 1.25 + .25N μsec



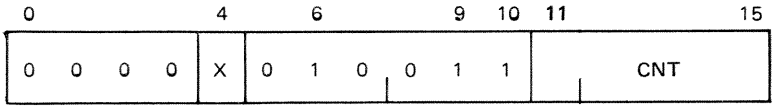
$(A)_0 \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_{15} \rightarrow B_0$



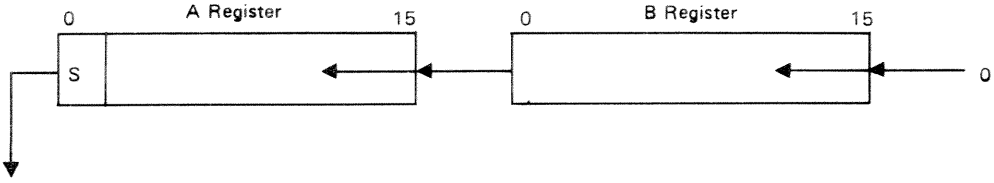
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the right the number of places designated by CNT ($0 \leq CNT < 32$). The sign bit is propagated from one position to the next, CNT number of places. The data shifted out of B_{15} is lost.

LAL — LONG ARITHMETIC LEFT SHIFT — 1.25 + .25N μsec



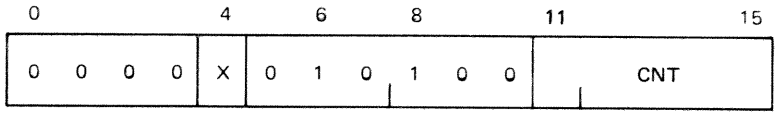
$(A)_i \rightarrow A_{i-1}$, $(B)_i \rightarrow B_{i-1}$, i = 0 to CNT
$(0 \leq CNT < 32)$
$(B)_0 \rightarrow A_{15}$
$0 \rightarrow B_{15}$



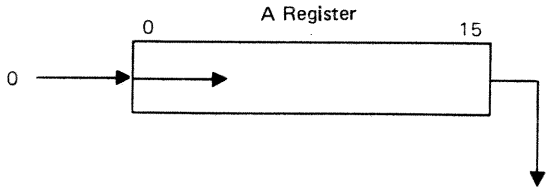
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the left the number of places designated by CNT ($0 \leq CNT < 32$). Zeros are inserted CNT number of places into B_{15}. The sign bit, A_0, is lost; however, the overflow indicator is set if the sign changes.

SLR **SHORT LOGICAL RIGHT SHIFT** **1.25 + .25N μsec**



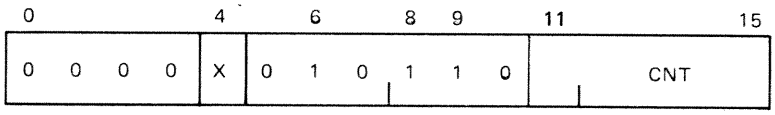
$0 \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, i = 0 to CNT
$(0 \leq CNT < 32)$



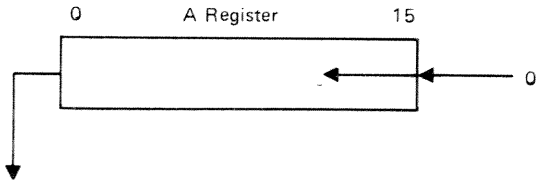
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the right the number of places specified by CNT ($0 \leq CNT < 32$). CNT number of zeros are inserted into A_0 and data is lost from A_{15}.

SLL **SHORT LOGICAL LEFT SHIFT** **1.25 + .25N μsec**



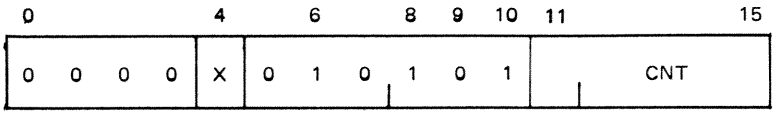
$(A)_i \rightarrow A_{i-1}$, i = 0 to CNT
$(0 \leq CNT < 32)$
$0 \rightarrow A_{15}$



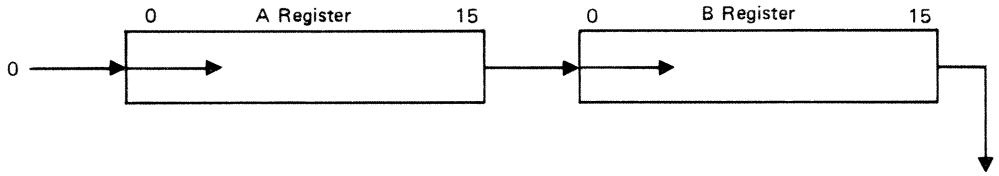
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the left the number of places designated by CNT ($0 \leq CNT < 32$). Zeros are inserted CNT number of places into A_{15} and data is lost from A_0.

LLR LONG LOGICAL RIGHT SHIFT 1.25 + .25N μsec



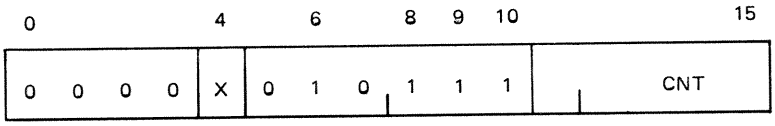
$0 \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}, (B)_i \rightarrow B_{i+1}$, i = 0 to CNT
$(0 \leq CNT < 32)$
$(A)_{15} \rightarrow (B)_0$



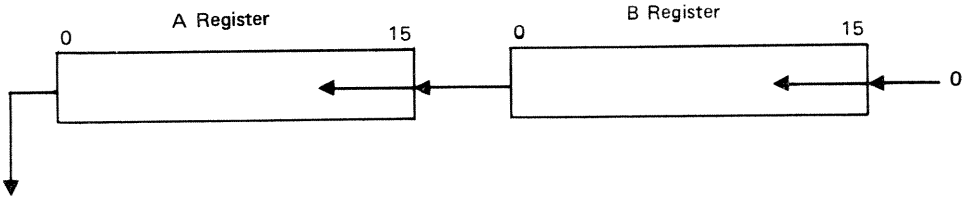
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B register to the right the number of places designated by CNT ($0 \leq CNT < 32$). Zeros are inserted into A_0 and data is lost from B_{15}.

LLL LONG LOGICAL LEFT SHIFT 1.25 + .25N μsec



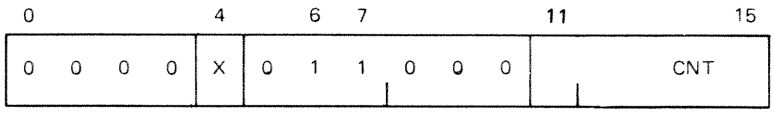
$(A)_i \rightarrow A_{i-1}, (B)_i \rightarrow B_{i-1}$, i = 0 to CNT $(0 \leq CNT < 32)$
$(B)_0 \rightarrow A_{15}$
$0 \rightarrow B_{15}$



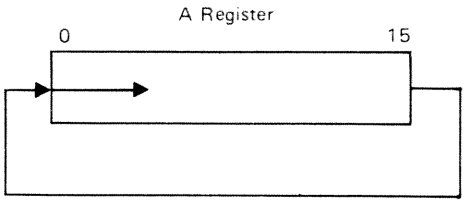
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B register to the left the number of places designated by CNT ($0 \leq CNT < 32$). Zeros are inserted into B_{15} and data is lost from A_0.

SRR **SHORT ROTATE RIGHT** 1.25 + .25N μsec



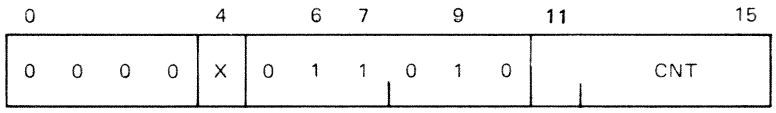
$(A)_{15} \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)



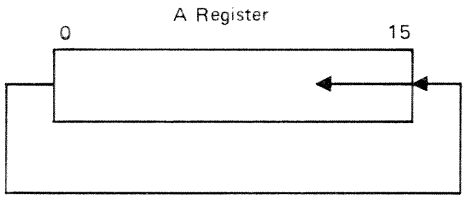
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the right the number of places designated by CNT ($0 \leq CNT < 32$). The content of A_{15} is inserted in A_0 and the shift continues in a circular manner without losing any data.

SRL **SHORT ROTATE LEFT** 1.25 + .25N μsec



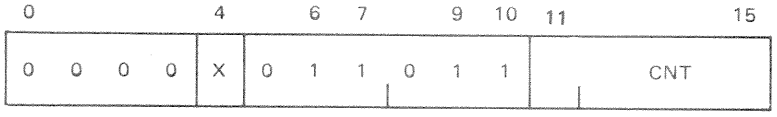
$(A)_i \rightarrow A_{i-1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_0 \rightarrow A_{15}$



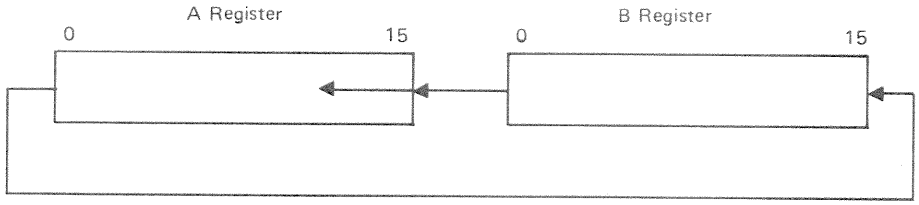
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the left the number of places designated by CNT ($0 \leq CNT < 32$). The contents of A_0 is inserted into A_{15} and the shift continues in a circular manner without losing any data.

LRL LONG ROTATE LEFT 1.25 + .25N μsec



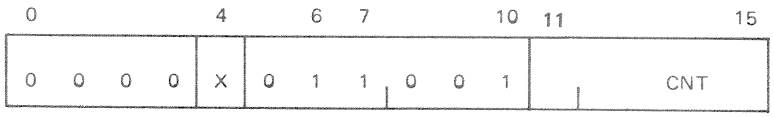
$(A)_i \rightarrow A_{i-1}$, $(B)_i \rightarrow B_{i-1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(B)_0 \rightarrow A_{15}$
$(A)_0 \rightarrow B_{15}$



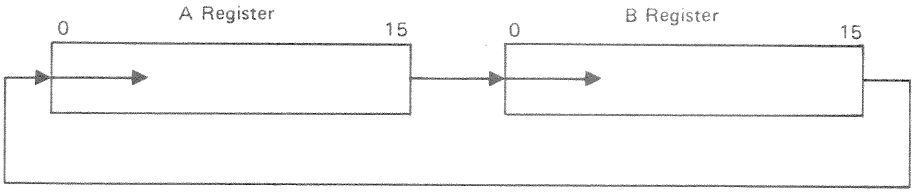
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the left the number of places designated by CNT ($0 \leq CNT < 32$). The content of A_0 is inserted into B_{15} and B_0 is transferred to A_{15}; the shift continues in a circular manner without losing data.

LRR LONG ROTATE RIGHT 1.25 + .25N μsec



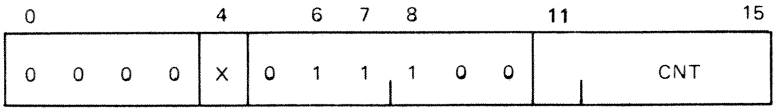
$(B)_{15} \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_{15} \rightarrow B_0$



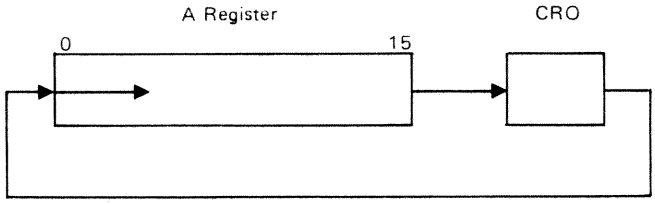
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the right the number of places designated by CNT ($0 \leq CNT < 32$). The content of B_{15} is inserted into A_0 and A_{15} is transferred to B_0; the shift continues in a circular manner without losing data.

SCR **SHORT CIRCULATE RIGHT** **1.25 + .25N μsec**



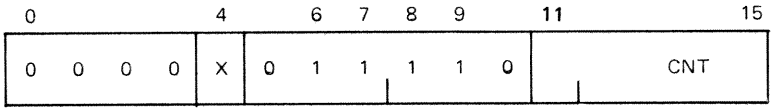
$(CRO) \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_{15} \rightarrow CRO$



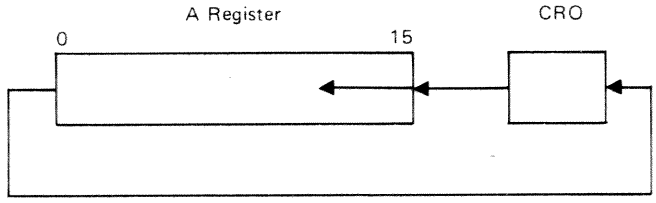
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the right the number of places determined by CNT ($0 \leq CNT < 32$). The content of A_{15} is inserted into the carryout indicator and the carryout indicator is transferred to A_0; the shift continues in a circular manner without losing data.

SCL **SHORT CIRCULATE LEFT** **1.25 + .25N μsec**



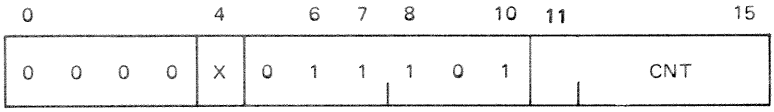
$(A)_i \rightarrow A_{i-1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_0 \rightarrow CRO$
$(CRO) \rightarrow A_{15}$



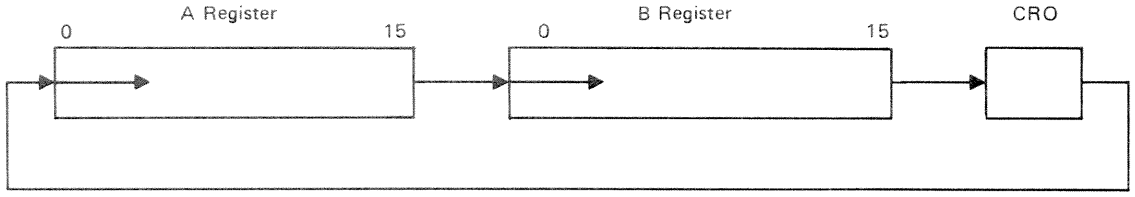
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the accumulator to the left the number of places designated by CNT ($0 \leq CNT < 32$). The content of the carryout indicator is inserted in A_{15}, and A_0 is transferred to the carryout; the shift continues in a circular manner without losing data.

LCR **LONG CIRCULATE RIGHT** **1.25 + .25N μsec**



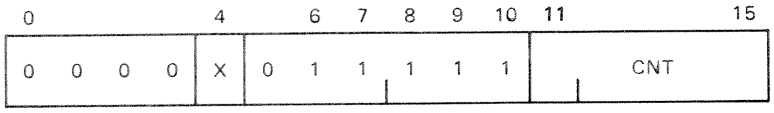
$(CRO) \rightarrow A_0$
$(A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(A)_{15} \rightarrow B_0$
$(B)_0 \rightarrow CRO$



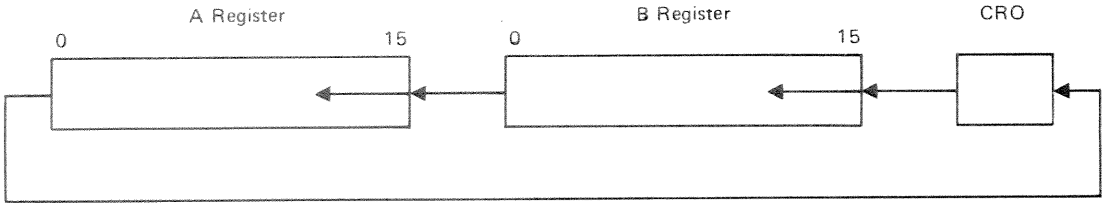
If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the right the number of places designated by CNT ($0 \leq CNT < 32$). The content of the carryout indicator is inserted into A_0, A_{15} is transferred into B_0, and B_{15} is moved to the carryout indicator; the shift continues in a circular manner without losing any data.

LCL **LONG CIRCULATE LEFT** **1.25 + .25N μsec**



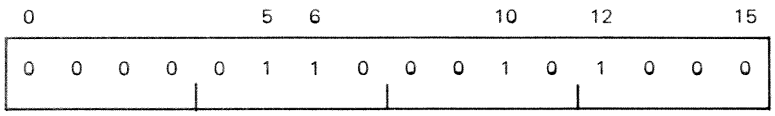
$(A)_i \rightarrow A_{i-1}$, $(B)_i \rightarrow B_{i-1}$, i = 0 to CNT ($0 \leq CNT < 32$)
$(CRO) \rightarrow B_{15}$
$(B)_0 \rightarrow A_{15}$
$(A)_0 \rightarrow CRO$



If bit 4 = 1, the contents of the index register are added to the instruction; i.e., $(INS)_{0-15} + (X)_{0-15} \rightarrow INS_{7-15}$.

This instruction then shifts the contents of the A and B registers to the left the number of places specified by CNT ($0 \leq CNT < 32$). The content of the carryout indicator is inserted into B_{15}, B_0 is transferred into A_{15}, and A_0 is brought into the carryout indicator; the shift continues in a circular manner without losing data.

LLO LOCATE LEADING ONE



If (A) = 0, (L) + 1 → L.
If (A) ≠ 0, $(A)_i \rightarrow (A)_{i-1}$ and (X) + 1 → X until $A_0 = 1$.
When $A_0 = 1$, (L) + 2 → L and $0 \rightarrow A_0$.

If the contents of the accumulator are equal to zero, the location counter is incremented by one and the next sequential instruction is executed.

If the contents of the accumulator are not equal to zero, bit 0 is checked. If bit 0 is not equal to one, the contents of the accumulator are shifted to the left one binary position and the index register is incremented. This procedure continues until bit 0 of the accumulator is found equal to one. The location counter is then incremented by two and bit 0 is reset (=0).

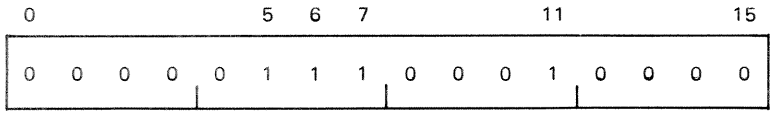
Timing for LLO is determined by the following:

If (A) = 0, 1.25 μsec

If (A) ≠ 0, 1.85 + .20N μsec (N = bit position which is equal to one).

One use for this instruction is to determine what interrupt has occurred in an external interrupt service routine. LLO will quickly indicate which bit in the A register was set (=1).

NDX NORMALIZE AND DECREMENT INDEX



If (A) and (B) = 0, 0 → X
If (A) or (B) ≠ 0, $(A)_i \rightarrow A_{i-1}$, $(B)_i \rightarrow B_{i-1}$, $(B)_0 \rightarrow A_{15}$, (X)-1 → X
until $(A)_0 \neq (A)_1$

If the contents of both the accumulator and B register are equal to zero, the index register is reset (=0) and the next sequential instruction is executed.

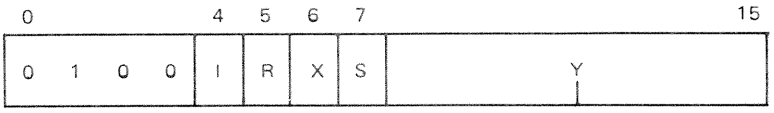
If the contents of either the accumulator or B register are not equal to zero, bit 0 and bit 1 of the accumulator are compared. If bit 0 equals bit 1, then the two registers are linked, their contents are shifted left one position, and the index register is decremented by one. This procedure continues until bit 0 is found not equal to bit 1.

If (A) and (B) = 0, the time of execution is 1.45 μsec; otherwise, 1.65 + .4N μsec.

The normalize instruction is used in arithmetic subroutines to give greater precision for quantities represented in floating point notation. Usually, the exponent is placed in the X register and the normalize instruction is then executed.

JUMP AND SKIP INSTRUCTIONS

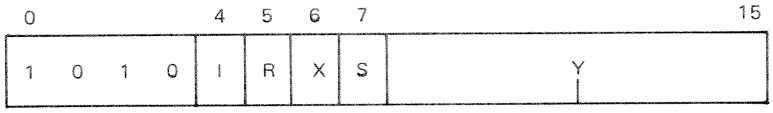
JMP JUMP 0 .92 μsec



EA → L

After all modifications (as indicated by the address control bits) are completed, the effective address is placed in the location counter. Control is then transferred to the instruction sequence at that location. JMP is not affected by the IMODE bit of the Status Register.

JSL JUMP AND STORE LOCATION 2.76 μsec



(L) + 1 → EA, (EA) + 1 → L

The contents of the location counter plus one are stored at the *address specified by the contents of the effective address.* Then, the effective address value plus one is placed in the location counter, and control is transferred to that location.

Example:

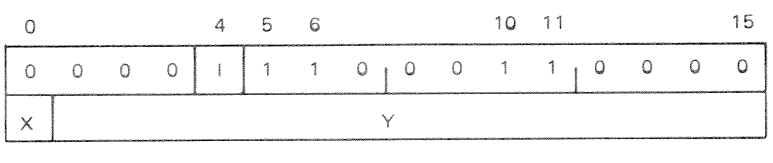
```
500          JSL     SUB
501          LDA     TAX
 .           .       .
 .           .       .
 .           .       .
600   SUB    PAR     SAVE
601          RINC    A,A,Z

Location     500     before
Counter      601     after

SAVE         501     after
```

If JSL specifies indirect addressing and the IMODE and MODE bits of the Status Register indicate user map and system mode, respectively, then the indirect address pointer and the contents of the location specified by the pointer are mapped in user mode; all other information is mapped in the system mode.

JRT JUMP RETURN 2.81 μsec



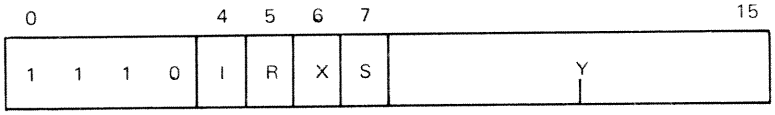
(EA) → L

The contents of the address specified by the effective address are copied into the location counter and control transferred to that location.

Example:

```
600    SUB    PAR     SAVE
601           RINC    A,A,Z
 .             .       .
 .             .       .
 .             .       .
610           JRT
                      PAR     SAVE
Location              610     before
Counter               501     after
SAVE                  501
```

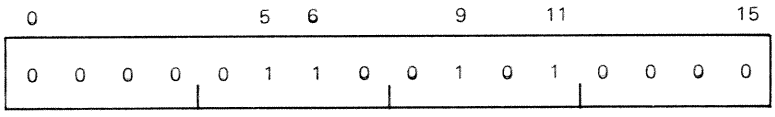
SKN **SKIP IF A ≠ MEMORY** **1.84 μsec**



If (A) = (EA), (L) + 1 → L
If (A) ≠ (EA), (L) + 2 → L

The contents of the A register are compared to the contents of the effective address. If they are equal, the location counter is incremented by one. If they are not equal, the location counter is incremented by two.

COT **CARRYOUT TEST** **1.25 μsec**

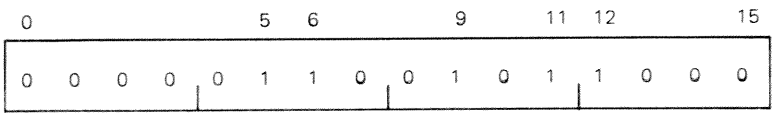


If CRO = 1, (L) + 1 → L; 0 → CRO
If CRO = 0, (L) + 2 → L

The carryout indicator is tested. If it is set (=1), the location counter is incremented by one and the carryout indicator is reset.

If the carryout indicator is reset (=0) when tested, the location counter is incremented by two.

OFT **OVERFLOW TEST** **1.25 μsec**



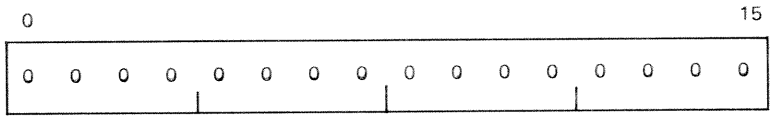
If OVF = 1, (L) + 1 → L; 0 → OVF
If OVF = 0, (L) + 2 → L

The overflow indicator is tested. If it is set (=1), the location counter is incremented by one and the overflow indicator is reset.

If the overflow indicator is reset (=0) when tested, the location counter is incremented by two.

CONTROL INSTRUCTIONS

HLT — HALT



The machine decodes the halt instruction and switches from the run to halt mode. To return it to the run mode, the operator presses the RUN button on the console. The SCC 4700 can also switch to the run mode as a result of an interrupt. All of the following interrupts will cause it to be placed in the run mode.

Power Off

Power On

Real Time Clock = 0

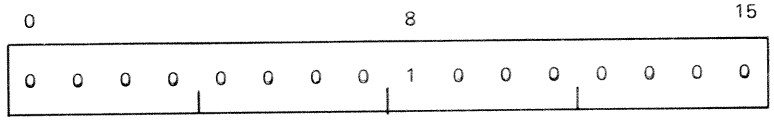
Channel Interrupt

Console Interrupt

Any external interrupt.

Halt is a privileged instruction.

ENA — ENABLE INTERRUPTS — 0 .92 μsec



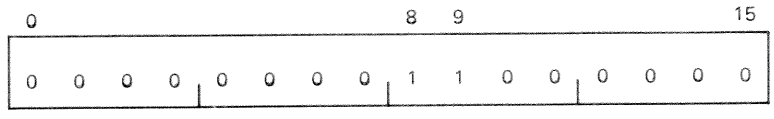
This instruction activates the interrupt system.

Until it is executed, no interrupt will be honored; however, any armed interrupts which occur will be "remembered" and serviced in priority sequence when the system is enabled.

Once the interrupt system is activated, it can be deactivated only by the Disable Interrupts instruction or by pressing the SYSTEM RESET button on the console, which will disable the interrupt system and disarm the interrupts.

Enable Interrupts is a privileged and non-interruptable instruction.

DIS — DISABLE INTERRUPTS — 0 .92 μsec



This instruction deactivates the interrupt system.

After it is executed, interrupts will not be serviced, even if armed. Any armed interrupts which occur will be "remembered" and serviced in priority sequence when the system is enabled.

It is a privileged and non-interruptable instruction.

The enable/disable instructions affect the following interrupts as a group:

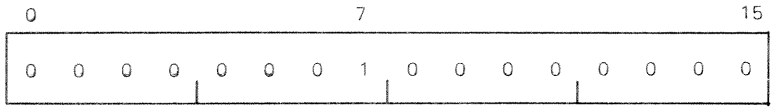
Interrupts for channels 1 through 4

External interrupts

Console interrupt

Real time clock

CLI CLEAR INTERRUPT 0.92 μsec

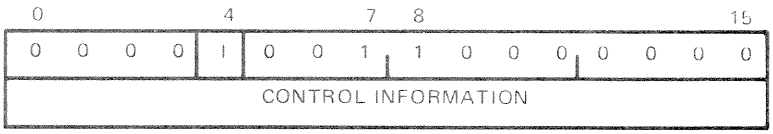


This instruction clears the active interrupt which has highest priority. All lower priority interrupts requesting service remain in the waiting state until this instruction is executed; then, the next highest becomes active.

This instruction resets the active and waiting flip-flops of an interrupt being serviced. (Refer to Section V, "Interrupt System".)

CLI is a privileged and non-interruptable instruction.

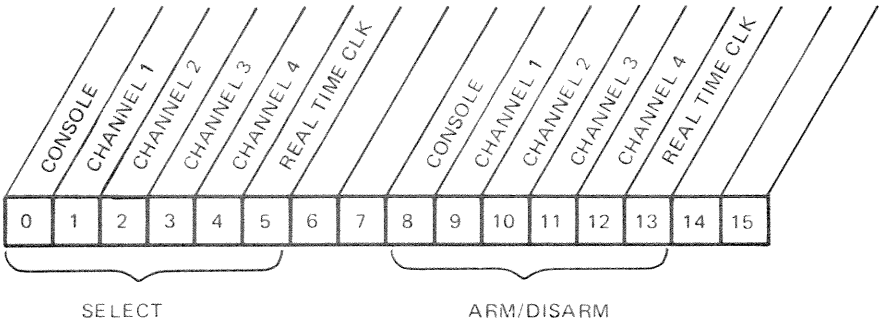
ARM ARM INTERRUPTS 1.92 μsec



ARM is a non-interruptable, privileged instruction which controls the individual arming of programmable internal interrupts. Control information is transmitted to the internal interrupt system by execution of the ARM instruction.

ARM is a double word instruction, the first word of which is in the Extended Op Code format. If bit 4 = 0, the second word contains the control information. If bit 4 = 1, the second word becomes a pointer to the control information. This pointer may be indexed and is written in the following format:

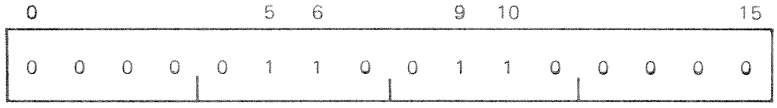
The format of the control information is:



The SELECT and ARM/DISARM bits for a given channel are interpreted as follows:

SELECT	ARM/DISARM	Action
0	X	unchanged
1	0	DISARM
1	1	ARM

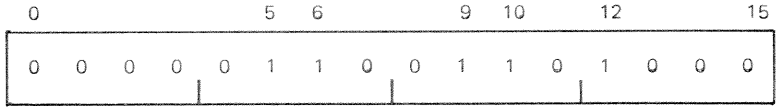
SCO SET CARRY OUT **1.00 μsec**



1 → CRO

This instruction causes the carryout indicator to be set (=1).

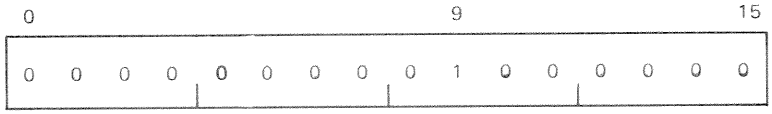
SOF SET OVERFLOW **1.00 μsec**



1 → OVF

This instruction causes the overflow indicator to be set (=1).

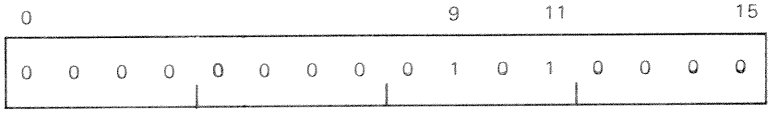
SISM SET INDIRECT SYSTEM MODE **0.92 μsec**



0 → ST_{10}

This privileged instruction resets (=0) the indirect mode bit (bit 10) of the Status Register, thereby preventing the system from using the user map on indirect memory references.

SIUM SET INDIRECT USER MODE **0.92 μsec**

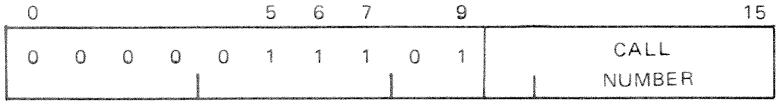


1 → ST_{10}

This privileged instruction sets (=1) the indirect mode bit (bit 10) of the status register, thereby allowing the system to utilize the user map on indirect memory references.

SYSTEM INSTRUCTIONS

SYCL — SYSTEM CALL — 7.64 μsec



$(L) \rightarrow ((BA + INS_{10\text{-}15}))$
$(ST) \rightarrow ((BA + INS_{10\text{-}15})) + 1$
$(X) \rightarrow ((BA + INS_{10\text{-}15})) + 2$
$(A) \rightarrow ((BA + INS_{10\text{-}15})) + 3$
$(B) \rightarrow ((BA + INS_{10\text{-}15})) + 4$
$0 \rightarrow CRO, OVF$
$(BA + INS_{10\text{-}15}) + 1 \rightarrow L$

System Call is a non-interruptable instruction which causes a trap to a reserved area of 64 locations starting at the base address (284_{10}). The call number in bits 10 to 15 of the instruction plus the base address give the location through which control is transferred.

Control is transferred to the address stored in the reserved location. The instruction automatically stores the contents of the L, ST, X, A, and B registers in five locations, starting with the address specified in the first word of the routine being called.

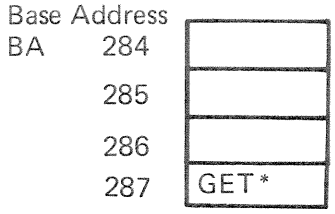
The mode, indirect mode, carryout, and overflow indicators of the status register will be reset (=0) after storing the status and before exiting from the instruction.

Example:

User Program

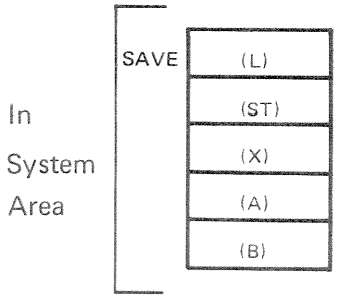
```
READ   SYCL   3
        .
        .
```

Base Address

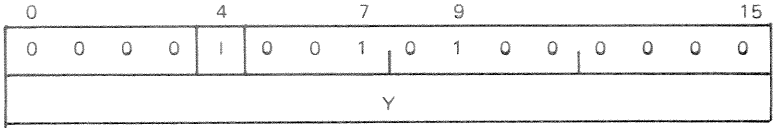


System Monitor (in system map)

```
GET    PAR    SAVE
        .
        .
        .
```



SRT — SYSTEM RETURN — 6.44 μsec



$(EA) \rightarrow L$
$(EA + 1) \rightarrow ST$
$(EA + 2) \rightarrow X$
$(EA + 3) \rightarrow A$
$(EA + 4) \rightarrow B$

---

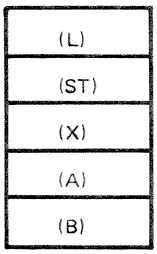
*GET is mapped in system mode and is non-indexable.

The System Return instruction is used to return control to the interrupted program after an interrupt or system call. The instruction restores the contents of the L, ST (except Halt), X, A, and B registers at the time of the interrupt or system call, by returning on the same "save" location assembled in the first word of the subroutine and reloading the registers with the values stored in the five reserved locations.

Example:

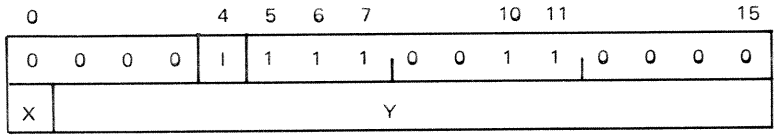
```
SUB     PAR     SAVE
          .
          .
          .
        SRT
        PAR     SAVE
          .
          .
          .

SAVE
```



The System Return instruction is a privileged instruction which cannot be pre- or post-indexed. SRT is not affected by the IMODE bit of the Status Register.

TSL **TEST AND SET LOCK** **3.12 μsec**



If (EA) = 0, (L) + 1 → L
If (EA) ≠ 0, (L) + 2 → L; 0 → EA

For systems operating in a multiple processor environment, this instruction provides the capability of restricting access to routines which might be executed by both processors at the same time. In such situations, each CPU must perform this instruction prior to entering the restricted sequence of instructions. Thus, protection is accomplished by individual routines and not by physical core locations.

TSL tests the effective location for zero (the locked condition). If the effective location is zero, the location counter is incremented by one.

If the effective location is not zero (not locked), the location counter is incremented by two and the lock location is set to zero.

This instruction may be indexed and indirectly addressed.

Example:

```
                        TSL
                        PAR     LOCK
CPU–1                   JMP     *–2
                        JSL     COMMON
                        STA     LOCK
```

```
                COMMON  PAR     XSAV
                        .
COMMON                  .
ROUTINE                 .
                        LDL     –1
                        JRT
                        PAR     XSAV
                LOCK    PAR     –1
```

```
                        TSL
                        PAR     LOCK
CPU–2                   JMP     *–2
                        JSL     COMMON
                        STA     LOCK
```

SECTION IV

INPUT/OUTPUT SYSTEM

The capabilities of the SCC 4700 input/output system are:

1. Data chaining which permits scatter read, gather write techniques
2. Mixed mode operations with different devices on same multiplexor channel
3. Servicing of multiple slow devices on same channel
4. High data transfer rates for fast devices
5. Full word or byte data transfer.

The I/O system uses one to four channel units to communicate with memory either directly or through the Central Processing Unit. The channel units may be multiplexor and/or selector channels, each of which can interface up to 64 device controllers.

Data transfers in the I/O system may be byte- or word-oriented. Byte transfers utilize either the multiplexor or selector channel and may be in either single byte or block transfer mode. Full word data transfer is performed through the parallel I/O interface and is controlled by the Read Parallel (RDP) and Write Parallel (WTP) instructions.

I/O control operations are separated into the following categories to direct the functions of three kinds of control equipment:

Instructions to be executed by the CPU

Commands to be executed by a channel

Orders to be executed by a device.

INPUT/OUTPUT CONTROL

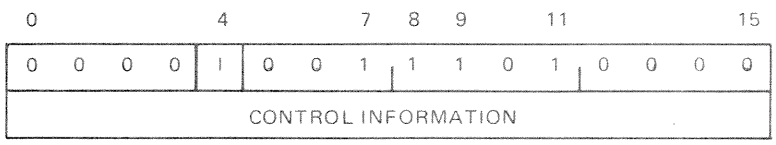
The SCC 4700 input/output system utilizes four instructions: three for parallel I/O and one for channel I/O.

Parallel I/O Instructions

A 16-bit, word-oriented, parallel I/O interface is used to read and control all devices on the parallel I/O bus, such as interrupts and special devices.

The device to be used is determined by the address data contained in the Activate (ACT) instruction. Data transfer between a device and the accumulator is accomplished by the Read Parallel I/O (RDP) and Write Parallel I/O (WTP) instructions.

ACT ACTIVATE PARALLEL I/O **1.84 μsec.**



ACT is a non-interruptable, privileged instruction used to activate any I/O device which utilizes the parallel I/O interface. It is necessary to give an ACT instruction if more than one device is connected to the parallel interface.

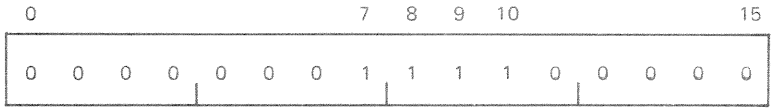
ACT is a double word instruction, the first word of which is in the Extended Op Code format. The second word contains the control information or the address of the control information. This double word instruction must occupy contiguous memory locations but may start on even or odd numbered locations.

The interpretation of the control information in the second word depends on the device and the user's system requirements. If bit 4 of ACT is set (=1), the second word becomes a pointer to the control information. This pointer is in the indirect address format and may be indexed.

2nd word if I = 1

The I/O control information is then output directly from memory and the location counter is incremented by one.

WTP WRITE PARALLEL I/O **1.95 μsec.**



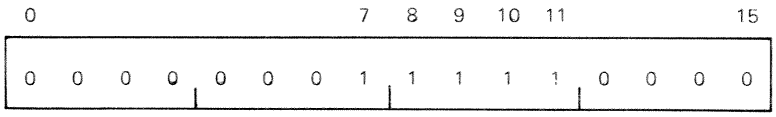
If device ready: (A) → Device, (L) + 2 → L.

WTP is a privileged instruction which transfers a 16-bit data word from the accumulator to the active device on the parallel I/O interface.

If the receiving device is ready, transfer is performed and the location counter is incremented by two.

If the device is not ready when tested by the WTP, the transfer is not performed and the machine proceeds to the next instruction after incrementing the location counter by one.

RDP READ PARALLEL I/O 1.95 μsec.



If device ready: (Device) → A, (L) + 2 → L.

RDP is a privileged instruction which transfers a 16-bit data word from the active device on the parallel I/O interface to the accumulator.

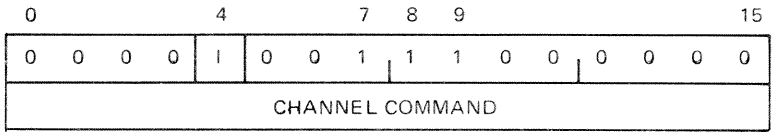
The transfer occurs only if the device is ready; the location counter is then incremented by two.

If the device is not ready when tested by RDP, the transfer is not performed and the machine proceeds to the next instruction after incrementing the location counter by one.

Channel I/O Instruction

The operation of the 8-bit, byte-oriented channels are controlled by an Input/Output Control (IOC) instruction.

IOC INPUT/OUTPUT CONTROL *



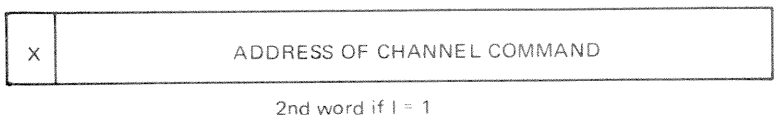
IOC is a privileged instruction which transmits the control information (channel command) required by either the multiplexor or selector channel controller.

IOC is a double word instruction, the first word of which is in the Extended Op Code format. The second word contains the channel command or the address of the channel command, depending on the value of bit 4.

If bit 4 is set (=1), the second word becomes a pointer to the channel command. This pointer is in the indirect address format and may be indexed.

---

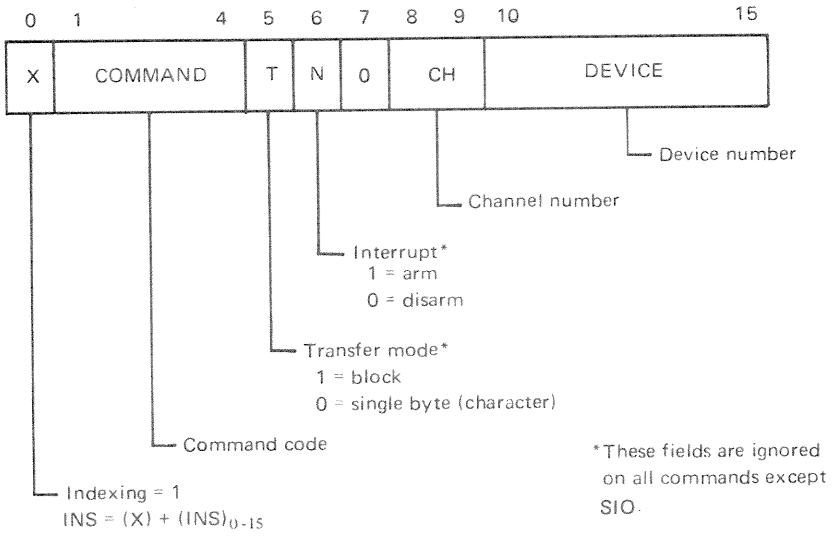
*The IOC is normally executed in 2.87 μsec (except Start I/O), providing an acknowledge from the device is received immediately; otherwise, it delays and checks for an acknowledge at 1.35 μsec intervals. If an acknowledge is not transmitted after 10 μsec, the "watchdog timer" (not the device) generates an acknowledge and the instruction proceeds. (An optional feature provides an external interrupt when the "watchdog timer" generates the acknowledge.)



The channel command is then output directly to the channel controller, which changes it to a sequence of signals acceptable to the control unit. (Refer to the section "Channel Commands" for commands available to the programmer.)

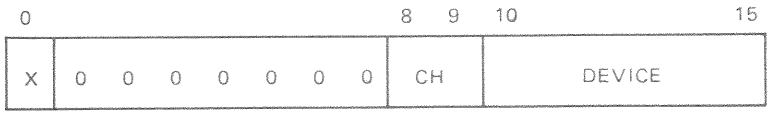
Channel Commands

The I/O channel commands are sent directly to the channel controller in the following single word format.



Each channel command must be immediately preceded by an IOC instruction or contained in the location specified by the address of the IOC.

HIO HALT I/O*

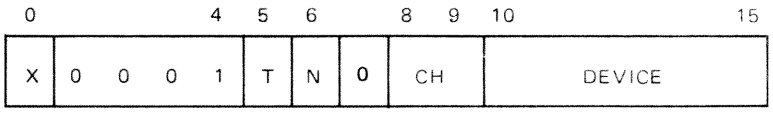


This command causes an immediate termination. The addressed device is halted and the settings of the entire device controller, including the status information, are cleared. HIO inhibits interrupts from the device and may cause information to be lost.

---

* HIO, TWC, system reset, and local reset all cause termination. A device also terminates on any error requiring operator intervention (such as card jam) or at the end of a record if a data error has been detected.

SIO — START I/O

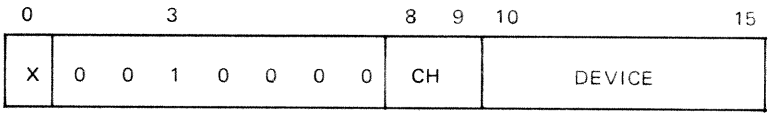


This command causes the device controller to be selected. If bit 5 = 1, SIO initiates channel operation for block transfers by starting automatic data transmission under control of the multiplexor or selector channel. The transfer mode (bit 5) and interrupt arm/disarm (bit 6) are transferred to the controller.

The device must be available when issuing the SIO command.

SIO clears previous status information.

XMT — TRANSMIT



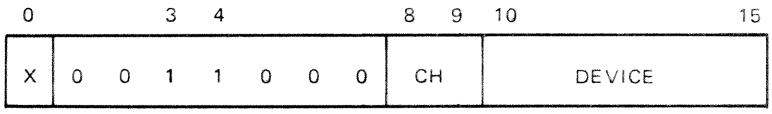
This command inputs or outputs a character to or from the least significant half of the accumulator (the most significant half is cleared on input). Direction of transfer is determined by the device number for the selected device:

even = input

odd = output

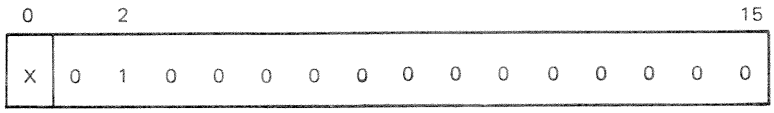
The device must be ready before executing a XMT command. XMT will then reset the ready bit and remove the device interrupt if in single byte mode.

EOA — EXECUTE ORDER IN ACCUMULATOR



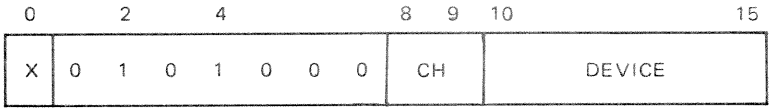
This command transmits an order (contained in the least significant half of the A register) to the device controller to initiate the desired operation. The action initiated by the selected device depends on the order format (which varies with each device).

IDN — INPUT DEVICE NUMBER



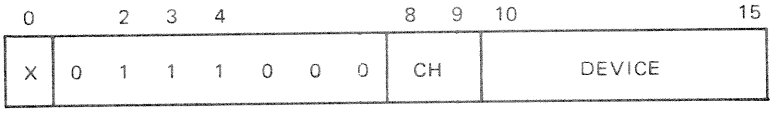
This command is not available to the programmer. It is used by the microcode of the CPU to identify an interrupting device when servicing the multiplexor channel for block transfers.

OUS — OUTPUT UNIT STATUS



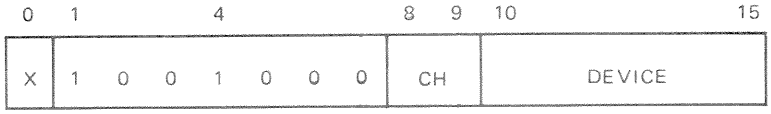
This command transmits 8 bits of data from the least significant half of the A register to the status register of the device controller. Bits of the status register may be set only; the device status cannot be reset or cleared by this command. The ready and available bits of the status register are not affected.

TWC* — TERMINATE WHEN COMPLETE



This is the usual command given to halt a device. Devices with predetermined record length will immediately stop data transfer; however, other devices, such as paper tape, may transmit the character currently being read before stopping transfer. When the device completes its current operation, the device and controller are terminated. The device gives a channel interrupt if armed when terminated. The status register remains valid.

SDR — SKIP IF DEVICE IS READY

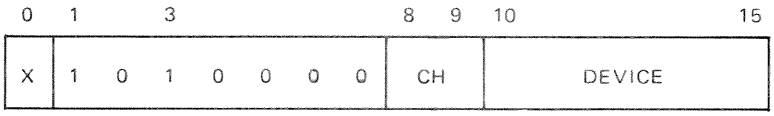


This command tests the status register of the device and, if the device is ready, increments the location counter by two.

---

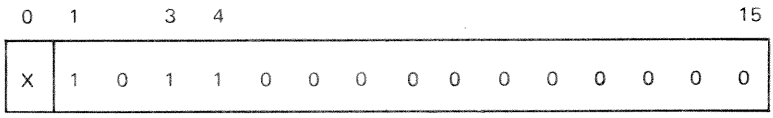
* HIO, TWC, system reset, and local reset all cause termination. A device also terminates on any error requiring operator intervention (such as card jam) or at the end of a record if a data error has been detected.

SDA SKIP IF DEVICE IS AVAILABLE



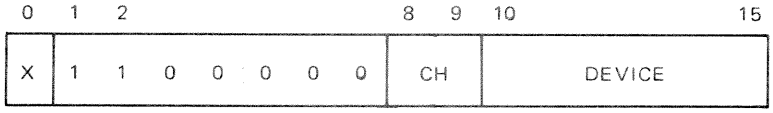
This command tests the status register of the device and, if the device is available, increments the location counter by two.

IIU INPUT INTERRUPTING UNIT



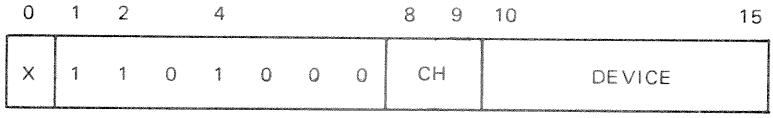
This command inputs the device number of the interrupting device into the least significant 6 bits of the index register and clears the high order bits.

IUS INPUT UNIT STATUS



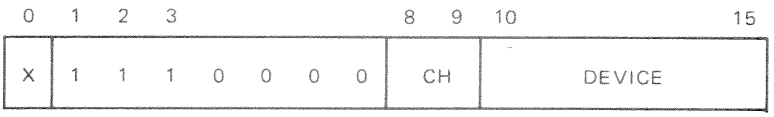
The command transmits the status register of the device being addressed into the least significant 8 bits of the A register and clears the most significant bits. IUS also clears the interrupt from the device.

ISB1 INPUT STATUS BYTE 1



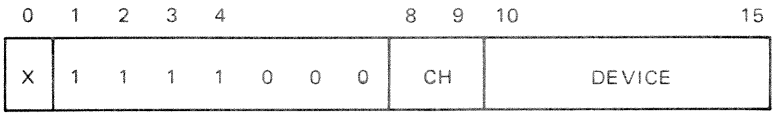
This command transmits additional device status information into the least significant 8 bits of the accumulator and clears the high order bits. The status information varies with each type of device and controller.

ISB2 INPUT STATUS BYTE 2



This command transmits another byte of device status information (different from the byte transferred by ISB1) to bits 8-15 of the accumulator and resets bits 0-7. The status information varies with each type of device and controller.

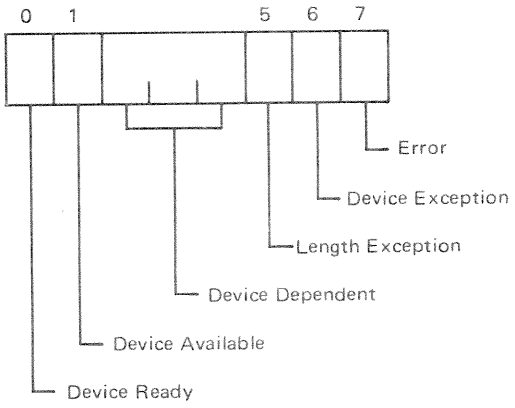
ISB3 INPUT STATUS BYTE 3



This command transmits another byte of device status information (different from the bytes transferred by ISB2 and ISB3) to bits 8-15 of the accumulator and resets bits 0-7. The status information varies with each type of device and controller.

Device Status

The format of the 8-bit device status register is as follows.



Bit 0	This bit set to a one indicates the device is ready and waiting for data transfer in character mode. If data transfer is in block mode, this bit will cause a transfer request to be generated.
Bit 1	This bit set to a one indicates the device is available for selection and use.
Bits 2-4	Device controller dependent.*
Bit 5	This bit set to a one indicates an error in the anticipated block count or record length.*
Bit 6	This bit set to a one indicates an unusual condition in the device completion, such as tape mark on tape, special characters, end of tape, out of paper, etc.*
Bit 7	This bit set to a one indicates an error condition which may have caused the data to be transferred incorrectly, such as parity error, rate overrun, card jam, etc.*

*Bits 2-7 may be set by channel command OUS (Output Unit Status). The status byte is cleared by channel commands SIO and HIO.

Device Orders

To initiate an input/output operation by a device, the device controller must transmit control information called a device order to the device in question. Due to the number of optional devices available for the SCC 4700, the codes, format, and interpretation of the device orders are not included in this manual. However, this information is included in the manual concerning the individual peripheral device.

INPUT/OUTPUT CHANNELS

The input/output system of the SCC 4700 utilizes from one to four data channels. The basic SCC 4700 includes one multiplexor channel and is expandable to include up to four channels in any combination of multiplexor and/or selector channels.

Multiplexor Channel

The multiplexor channel is under direct control of the Central Processing Unit and uses CPU data and address paths to memory.

This channel is capable of transferring data to and from up to 64 devices on a multiplex or "party-line" basis. Data transfer may be performed in single byte and block modes with each active device controller operating independently; i.e., some device controllers may be engaged in data transfer in the single byte mode, while other device controllers are transferring data in the block mode.

Selector Channel

Although activated by the Central Processing Unit, the selector channel operates independently of the CPU in providing direct memory access at a high data rate between a device and memory. Up to 64 byte-oriented device controllers may be serviced by the selector channel; however, only one device may be active at one time.

Block data transfer is accomplished through the command and control information sent directly from memory to the channel. When data transfer is in the block mode, "chaining" is permitted; i.e., the selector channel can sequentially execute a series of block transfers.

INPUT/OUTPUT SYSTEM OPERATION

Word Transfers

Full word data transfer is performed through the parallel I/O or the Direct Memory Access (DMA) ports.

1. Parallel I/O — The parallel I/O interface is controlled by the Activate Parallel I/O (ACT), Read Parallel (RDP), and Write Parallel (WTP) instructions. (Refer to the section "Parallel I/O Instructions" of this chapter.)

2. DMA Ports — The DMA Port provides a high speed, 16-bit, word-oriented interface between main memory and an external device. Data is routed to and from memory directly between the external device and the memory gating module.

   Since the DMA is designed for use with special device configurations, the reference manual concerning the device in question should be consulted.

Single Byte Transfers

Both the multiplexor and selector channels accept data in the single byte (character) mode. There is no difference in the operation of either channel during data transfer; i.e., data rate is the same, multiple devices may be serviced, etc.

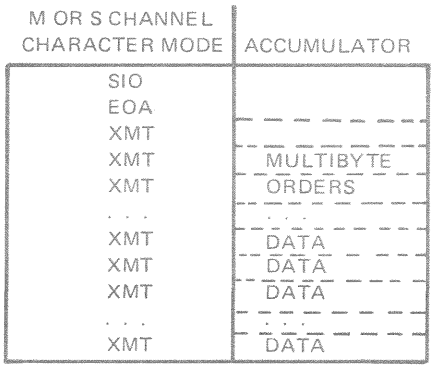
Data transfer is accomplished under program control, using the channel commands discussed previously, with data being output or input to or from the low order 8 bits of the accumulator. This mode of channel operation allows great flexibility to the programmer, placing the full capabilities of the channel under his complete control. The following illustrates a typical application of the single byte mode.

The device should first be tested for availability. If it is available, it may be selected with an SIO command and the device interrupt enabled, if operation under interrupt control is prefered. Next, an order is transmitted to the device with an EOA command to initiate the desired action.

If operation is not under interrupt control, the device is then tested to be ready. When it becomes ready, data is transmitted to or from it. This test for ready status and the subsequent data transmission sequence is repeated until the desired number of bytes have been transferred. The device operation is then terminated with a TWC command. The final status for the operation may then be tested when the device again becomes available.

If operation is under interrupt control, the interrupting unit may be determined by using the IIU command. The data is then transmitted after checking the device status if desired. When the data transfer is complete, the device is terminated with a TWC command. The device status is finally tested upon receiving its termination interrupt.

If there are multibyte orders, the first order byte is transmitted with an EOA command followed by a XMT command for each additional order byte. For example:



Block Transfers

The block transfer process to or from memory can be performed through either the multiplexor or selector channel. Transfers of multiple bytes of data through the selector channel occur asynchronously and independently of the Central Processing Unit; however, memory cycles may be stolen by the selector channel if the same memory module is accessed simultaneously.

All block transfers through the multiplexor and selector channel are initiated by execution of the Start I/O channel command. Transfer of data then proceeds automatically until completion, without further attention from the program.

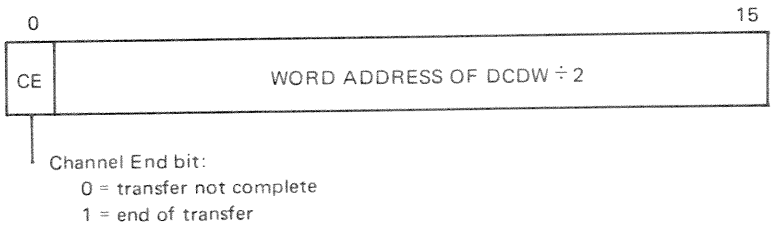
Both channels are programmed in exactly the same way; the only difference between the two channels occur in the manner the hardware handles the block transfer. Those to be considered are:

1. Faster data rate on selector channel.
2. Address and byte count are stored in core memory for the multiplexor channel, rather than in registers as with selector channel.
3. The microcode of the CPU handles the interrupt and data transfer which results in the multiplexor channel stealing some CPU time and memory cycles (the selector channel steals only memory cycles.)
4. Multiple devices can be active on the multiplexor channel at the same time, but only one device at a time can be active on the selector channel.

To perform a block transfer on either the multiplexor or selector channel, certain steps must be accomplished. (Refer to Figure 6.)

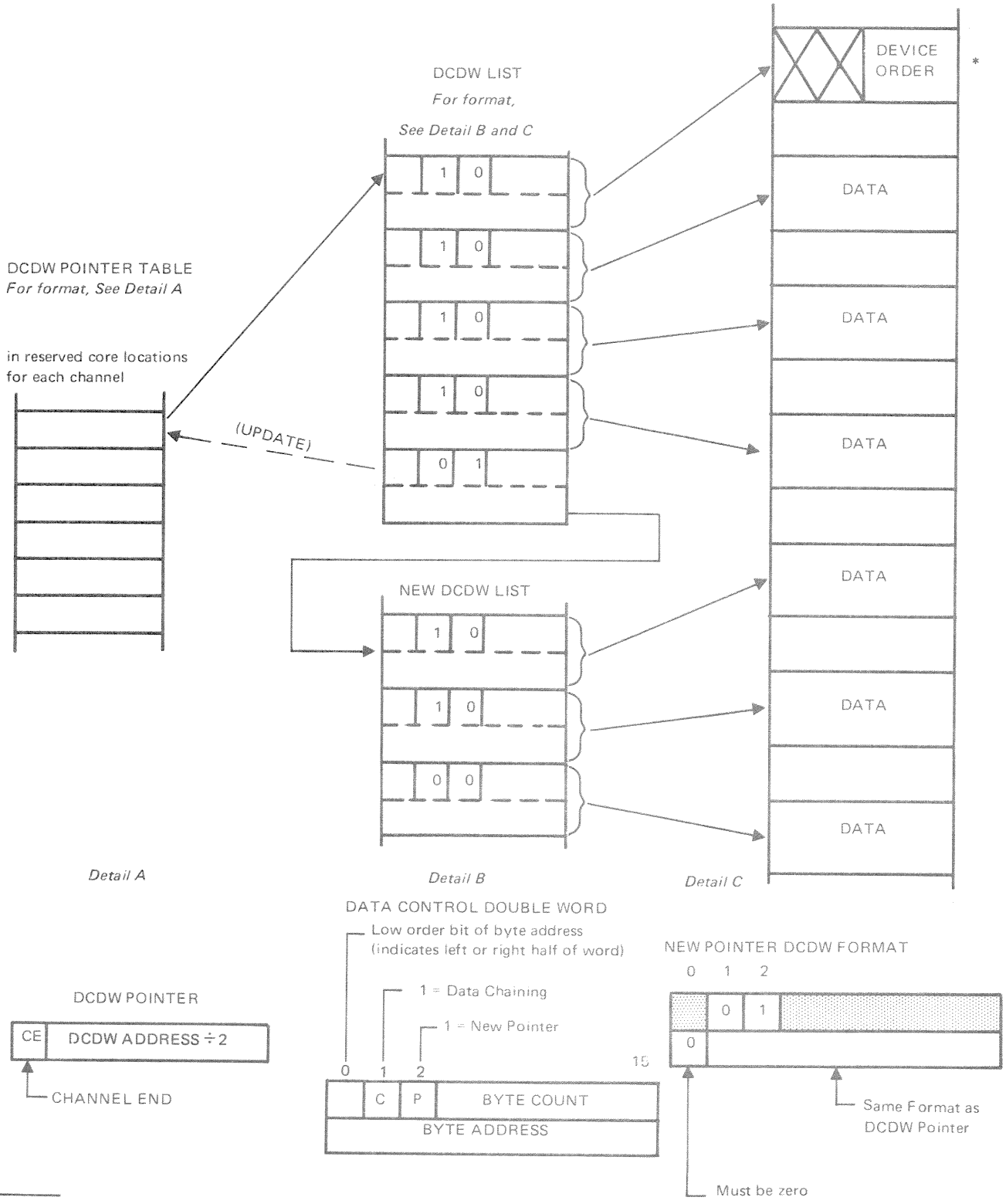
1. An input/output data area is reserved in memory.
2. A DCDW area is reserved in memory.
3. A pointer to the Data Control Double Word (DCDW) controlling this transfer is placed at the proper location in memory. The location for the DCDW pointer is calculated by: the channel number and device number (juxtapositioned as in the channel command) plus 28_{10}.

   The format of the DCDW Pointer is:



   The entry for the DCDW pointer is obtained by performing a logical right shift of one place on the real core address of the first DCDW.

   The channel end bit of the pointer is set (=1) by the channel when transfer is complete. If the program logic uses this bit to determine completion of transfer, it must be reset before the same pointer is referenced again.



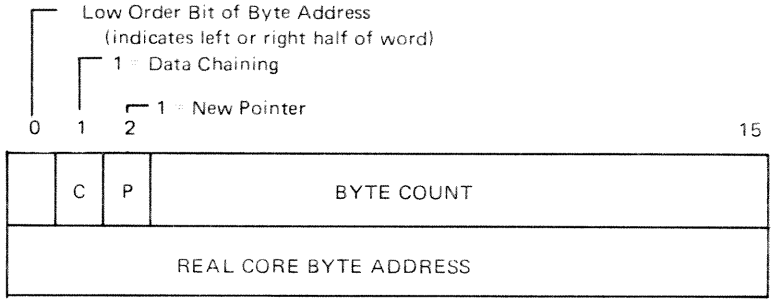
* If the device order is not placed with the data as shown above, separate DCDW's are used. If the device order occupies the first byte in the data field, only one DCDW is necessary.

Figure 6. Typical Block Transfer Operation

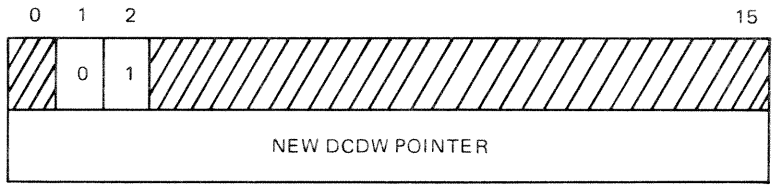
4. DCDW's are entered into memory, starting at the location specified by the DCDW pointer. The real core word address in the pointer is doubled to determine the word location of the DCDW; therefore, it is necessary that DCDW's always be located on even word boundaries.

Note: The channels use only real core addresses; no virtual addresses are allowed.

The Data Control Double Word contains the byte count and byte address of the data area(s) to/from which data is to be transferred. The format of the DCDW is:



Normal DCDW Format

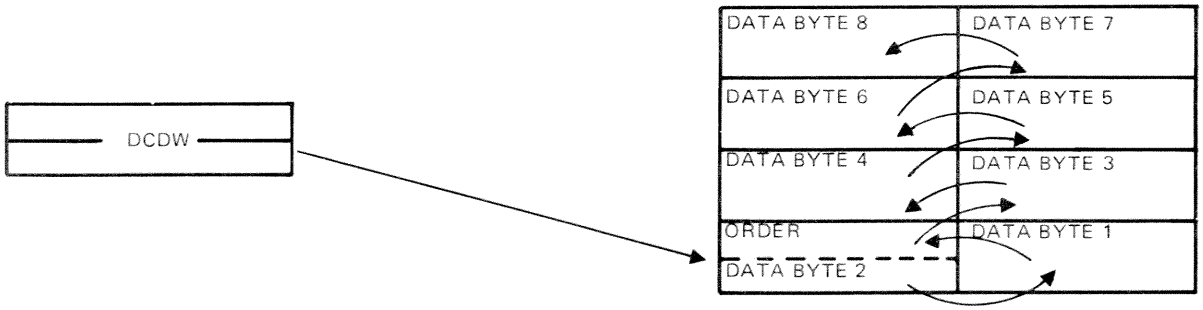


New Pointer DCDW Format

If bit 0 (the low order bit of the byte address) = 1, transfer starts with the right half-word (least significant half).

If bit 0 = 0, transfer starts with the left half word.

Note: If the device order is placed with the data to be transferred and the order causes the device to switch to a read backward mode, the following operation occurs: The DCDW specifies the location of the device order; after transferring the order, the byte address is incremented. On the next transfer, the machine is directed to this new location. After transferring the first data byte, the DCDW byte address is decremented and the machine is directed back to the byte containing the device order, overwriting the order byte with the next data byte. This condition is illustrated below.



If neither the data chaining bit (bit 1) or the new pointer bit (bit 2) are set, transfer ends at the completion of the DCDW. The channel end bit of the pointer is then set and the device terminates.

If the data chaining bit is set (=1), the channel, upon completion of the current DCDW, proceeds to the next DCDW in the list (which must be contiguous).

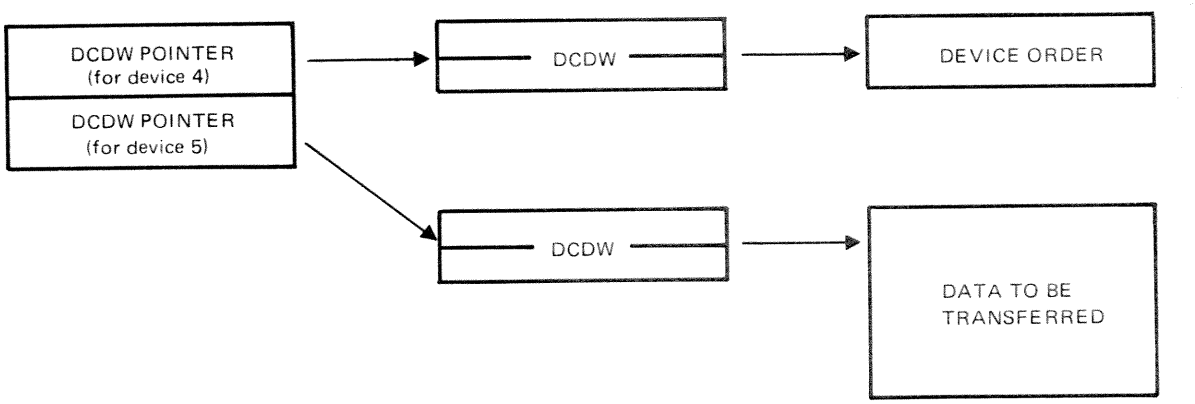
If, when proceeding to the next DCDW, the new pointer bit is set (=1), the second word of the DCDW becomes a new DCDW pointer. In the multiplexor channel this new DCDW pointer replaces the old DCDW pointer in memory. The selector channel does not modify memory, but maintains the new pointer in the channel hardware. (The count field of the new pointer DCDW is ignored.) Thus, the new pointer DCDW allows noncontiguous lists of DCDW's to be chained together.

The byte count field of the DCDW specifies the total number of bytes to be transferred under control of the current DCDW. It may never be specified as zero (except as a new pointer) and must not be large enough to allow the byte address to cross a block boundary in a mapped system (counts greater than 1024 are not recommended for general applications).

The second word of the DCDW contains a word address which, when combined with the low order bit in the first word, provides the byte address of the first byte to be transferred under control of the DCDW.

Some devices may reverse the direction of data transfer (input versus output) during certain operations (e.g., multibyte orders). On the selector channel this may occur only when chaining between DCDW's; however, operation will proceed normally using the new DCDW from the same list. On the multiplexor channel a change in direction may occur at any time (indicated by a change in device numbers) and will cause a different DCDW list to be executed. Thus, two DCDW lists must be prepared based on both the input and output device numbers of the device.

Example: Before performing the transfer operation, the multiplexor channel requests device identification. The device replys with a device number of 4, which indicates an input operation. The DCDW operation to transfer the device order on device number 4 (illustrated below) is performed. Upon completion of this DCDW operation, the channel again requests identification from the same device. This time the device will reply with a device number of 5, which indicates an output operation. As a result, the DCDW pointer will specify a new DCDW list to be used in transferring the proper data.



NOTE: The first byte of an order is always output regardless of the device number; however, additional bytes of a multibyte order must use an odd device number.

5. A device order is then written to activate the device controller. The order may be located separately from the data with its own DCDW or placed with the data to be transferred, but it must be the first byte to be output.

   For a multibyte order on the selector channel, a separate DCDW must be specified for the data.

6. After the above information has been prepared and assembled in the proper sequence and format and after verifying the device available, the desired data transfer is performed by executing an IOC (Input/Output Control) instruction followed by an SIO (Start I/O) channel command in the block mode.

   Example: IOC
   SIO 0E05

   On execution of the SIO command, the selector channel locates the data to be transferred by means of the DCDW pointer and DCDW's and performs the transfer asynchronously and independently of the CPU.

   For block transfers on the multiplexor channel, the CPU performs the necessary update to the DCDW's and DCDW pointer in memory and transfers each byte directly from memory, as each device becomes ready, asynchronously of the program. The programmable registers are not affected. The programmer may determine how far the multiplexor channel has progressed with the operation by testing the DCDW pointer and the DCDW's.

7. The transfer may be repeated on the selector channel by resetting the channel end bit (if desired) and executing another SIO.

   On the multiplexor channel the DCDW pointer and the DCDW's must all be reinitialized (new pointer DCDW's are not changed) before executing another SIO.

SIO (Start Input/Output) Channel Command Operation in Block Mode

The SIO command performs the following operations when transferring data in the block mode on the multiplexor channel.

1. An IOC instruction is executed by the CPU and the associated command is transferred by the channel to the device controller. If the command was an SIO in the block mode, the DCDW pointer location is determined by adding 28_{10} to the channel number and device number at it appears in the SIO command.

2. The DCDW location is calculated by doubling the contents of the DCDW pointer location.

3. If the byte count of the DCDW is zero, the operation branches to step 12. If it is not zero, the count is decremented, and the byte specified by the byte address of the DCDW is transferred to the device with a hardware generated EOA command.

4. The DCDW byte address is incremented.

5. If the resultant byte count is zero and the chaining bit is also zero, the channel end bit in the DCDW pointer is set and a TWC command is sent to the device. The TWC stops the device from requesting more data transmission and causes the device to terminate when complete. Upon termination the device generates an interrupt if armed.

6. If a channel or memory error occurred during the preceding operations, the error bit (IUS bit 7) in the device is set by transmitting an OUS command to the device.

7. The CPU then proceeds to execute the next instruction, while the channel waits for a transfer request from the device. (A transfer request is a signal from a device indicating it is ready for a data transmission in the block mode.)

8. When the channel receives a transfer request, it performs an IDN command. (This is a special command used by the channel to determine the highest priority device signaling a transfer request.) 28_{10} is then added to the channel and device number response received to the IDN command to determine the DCDW pointer location.

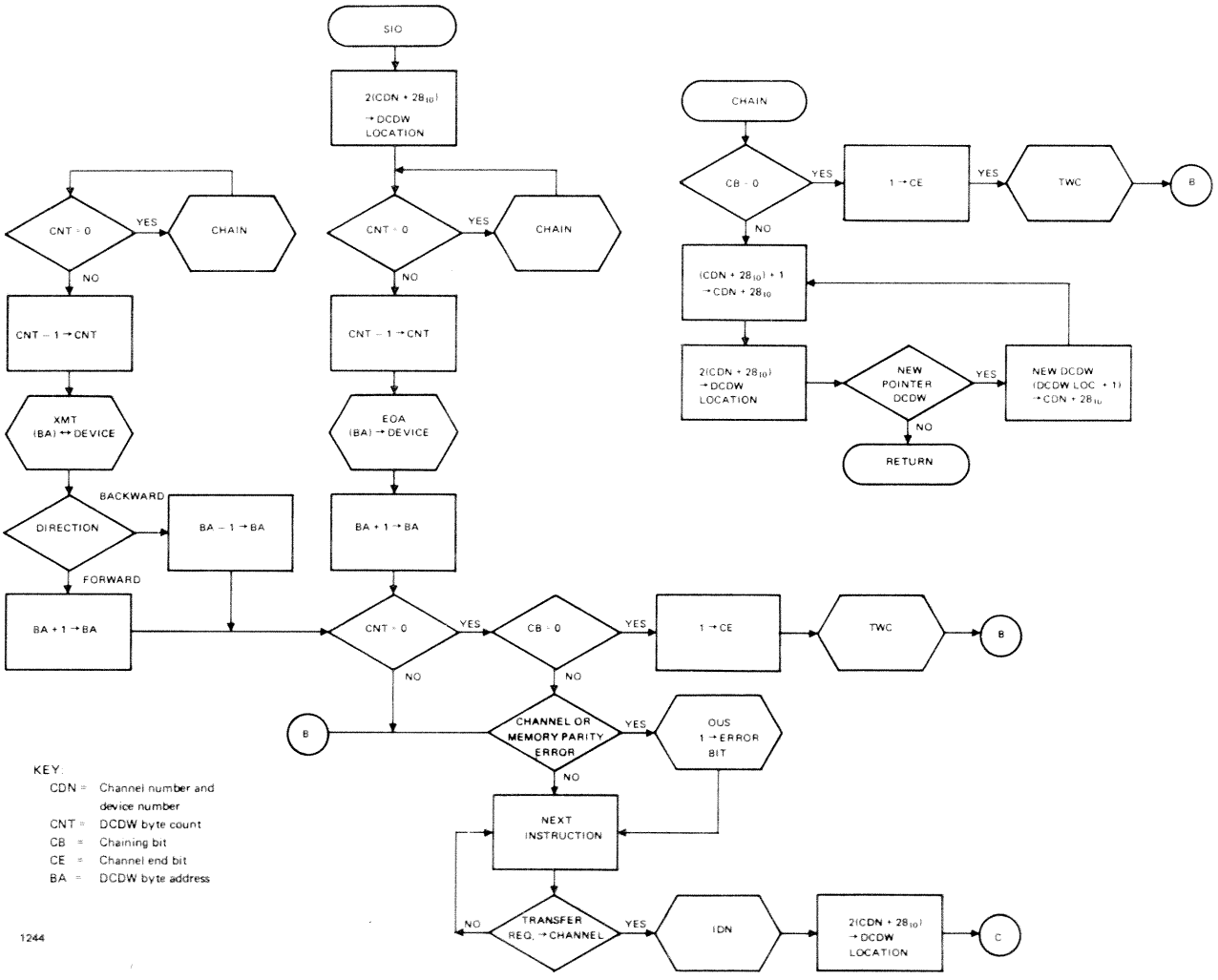
9. The DCDW location is calculated by doubling the contents of the DCDW pointer location.

10. If the byte count of the DCDW is zero, the operation branches to step 12. If it is not zero, the count is decremented, and the byte specified by the byte address of the DCDW is transferred to/from the device with a hardware generated XMT command. The direction of data transfer is determined by the least significant bit of the device number.

11. The byte address is then incremented (decremented if a read backward indication was received from the device). The operation returns to step 5.

12. If the chaining bit in the DCDW is not set, an error condition has been detected. The channel end bit is set, a TWC command is sent to the device and the operation proceeds to step 6. If the chaining bit is a one, the DCDW pointer is incremented to the next consecutive DCDW and the operation proceeds using this new DCDW.

13. If the new pointer bit is not set, the operation returns to and repeats the step that branched to step 12. If the new pointer bit is set, the second word of the DCDW replaces the DCDW pointer and this step is repeated using the new DCDW specified by the updated DCDW pointer.



SECTION V

INTERRUPT SYSTEM

GENERAL DESCRIPTION

The SCC 4700 interrupt capability allows the normal execution of a program to be interrupted in order to process a program of higher priority. Situations which may interrupt program execution are referred to as interrupt conditions.

Upon receiving notification of an interrupt condition, the SCC 4700 interrupt system instructs the central processor to process a program of high priority. The central processor stops execution of the current program and honors the request for an interrupt at the first available time. When the interrupt request is honored, the interrupt system supplies a memory location to the central processor. The central processor then generates in hardware and executes a System Call (SYCL) on the reserved location assigned for the interrupt (unless the interrupt is for power on). Contained in the reserved location is the effective address of the interrupt service subroutine. Hence, execution of SYCL on the interrupt location transfers control to the interrupt service routine. (This subroutine can be interrupted by the occurrence of an interrupt of higher priority, but not a lower until the current interrupt is cleared.)

On completion of the interrupt service subroutine (which resets the source of the interrupt), a Clear Interrupt (CLI) and System Return (SRT) instruction are executed so that lower priority interrupts may be serviced. The interrupt service subroutine then returns control to the interrupted program and execution of the interrupted program continues. An example of this concept is illustrated in Figure 7 below.

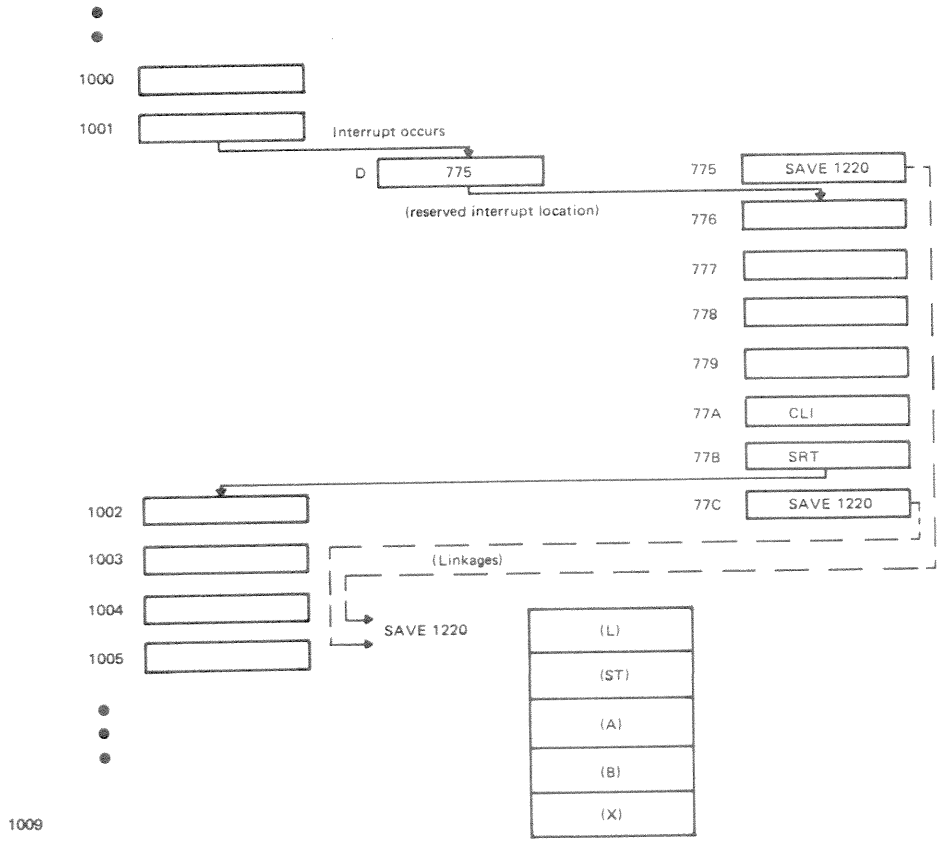


Figure 7. Interrupt Sequence

THEORY OF OPERATION

Enabling the Interrupt System

The interrupt system, except power up, power down, and system trap interrupts, is activated by the Enable Interrupts (ENA) instruction and deactivated by the Disable Interrupts (DIS) instruction. The ENA and DIS instructions affect all of the following interrupts as a group:

Interrupts for Channels 1 through 4

External Interrupts

Real Time Clock

Console Interrupt

Power up, power down, and system trap are enabled (i.e., ready to be activated) at all times and cannot be disarmed.

Arming Interrupts

Arming/disarming of the various interrupts in the basic SCC 4700 system is as follows:

External interrupts are always armed (selective arming/disarming is available as an option).

Individual device interrupts are armed/disarmed by an SIO command.

Channel interrupts, console interrupt, and real time clock interrupt are armed/disarmed by the ARM instruction.

A Master Clear or power on disarms and disables *all* interrupts.

Servicing Interrupts

An interrupt request from the interrupt system to the central processor is honored following completion of the instruction under execution unless the instruction is a non-interruptable instruction (ARM, ENA, DIS, CLI, SYCL, or ACT). If one of these instructions is being executed when the interrupt request occurs, the interrupt request is not granted until completion of an instruction which allows interrupts.

Interrupts and traps which are honored immediately (i.e., the instruction under execution is not completed) consist of the following:

Power On

Unimplemented Instruction } System Traps

Protection Key Violation } System Traps

Privileged Instruction Violation } System Traps

Memory Parity

The program interrupts contained in channels 1 through 4 and the external interrupt subsystem follow a two-state sequence when an interrupt condition occurs. If an interrupt is armed when an interrupt condition occurs, it immediately enters the waiting state. The waiting state is maintained until no higher priority interrupt is waiting or being serviced. An interrupt request and an identifying address to the CPU is then generated. When the CPU honors this interrupt request, the above interrupt switches from the waiting state to the active state. The interrupt remains in the active state until a CLI (Clear Interrupt) instruction, which clears the highest active state, is executed.

An interrupt in the active state may be interrupted by a higher priority interrupt. If several interrupts enter the waiting state, the CPU will honor each interrupt request individually according to priority.

Interrupt requests in the waiting state are honored by the CPU only when the interrupts have been group enabled by the ENA instruction.

PRIORITY STRUCTURE

The promptness with which an interrupt request is serviced is dependent on its priority; i.e., the highest priority interrupt is always serviced first, followed by the next highest, etc. After an interrupt has been serviced and cleared by the CLI instruction, the interrupt system prepares to receive the next highest priority interrupt waiting for service. Any lower priority interrupts which occur are remembered and serviced after each of the higher priority interrupts have been cleared.

The priority structure of the SCC 4700 interrupt system performs the following functions:

1. Resolves contention problems arising from simultaneous occurrence of interrupt conditions.
2. Permits an interrupt program to be interrupted by a request to service an interrupt of higher priority.
3. Permits lower priority interrupts which occur to be remembered and serviced at a later time.

Table 1 provides the priority level, core assignment, and timing of the various traps and interrupts which the SCC 4700 may service.

Table 1. SCC 4700 Traps and Interrupts

RESERVED CORE ASSIGNMENTS			PRIORITY LEVELS		TIMING	
DECIMALS	HEXADECIMAL	DESCRIPTION	SEQUENCE	REMARKS	μsec	MULTIPLE FUNCTIONS
0	0	Power Up Interrupt *	1		5.62	
1	1	Power Down Interrupt *	5		7.36	
2	2	Real Time Clock (location being decremented)	27		2.59	Decrement Location**
3	3	Real Time Clock Trap Location			8.73	Decrement, Test and Trap
4	4	Console Interrupt	28		7.36	
5	5	System Trap Indicator	4		7.76	Unimplemented Instruction Floating Point Over/Underflow * (1)
6	6	System Trap Location	2	System Protection	8.26	Protection Key Violation * Privileged Instruction * Memory Parity * (2)
7	7	Unassigned				
8	8	Channel 1 Interrupt	7-26	Service Request (lowest location has highest priority)	7.36	Service Request (I/O or external interrupts)
9	9	Channel 2 Interrupt *				
10	A	Channel 3 Interrupt *				
11	B	Channel 4 Interrupt *				
12–27	C–1B	External Interrupts *				
28–283	1C–11B	DCDW Pointers (64/channel)	6	Transfer** Request	7.94 ***	Normal Transfer (block mode)
					10.33	Transfer with data chaining
					12.47	Transfer with chaining and new pointer.
284–347	11C–15B	System Calls (64 monitor entry points)			7.64	Storage of register plus transfer

* Optional.

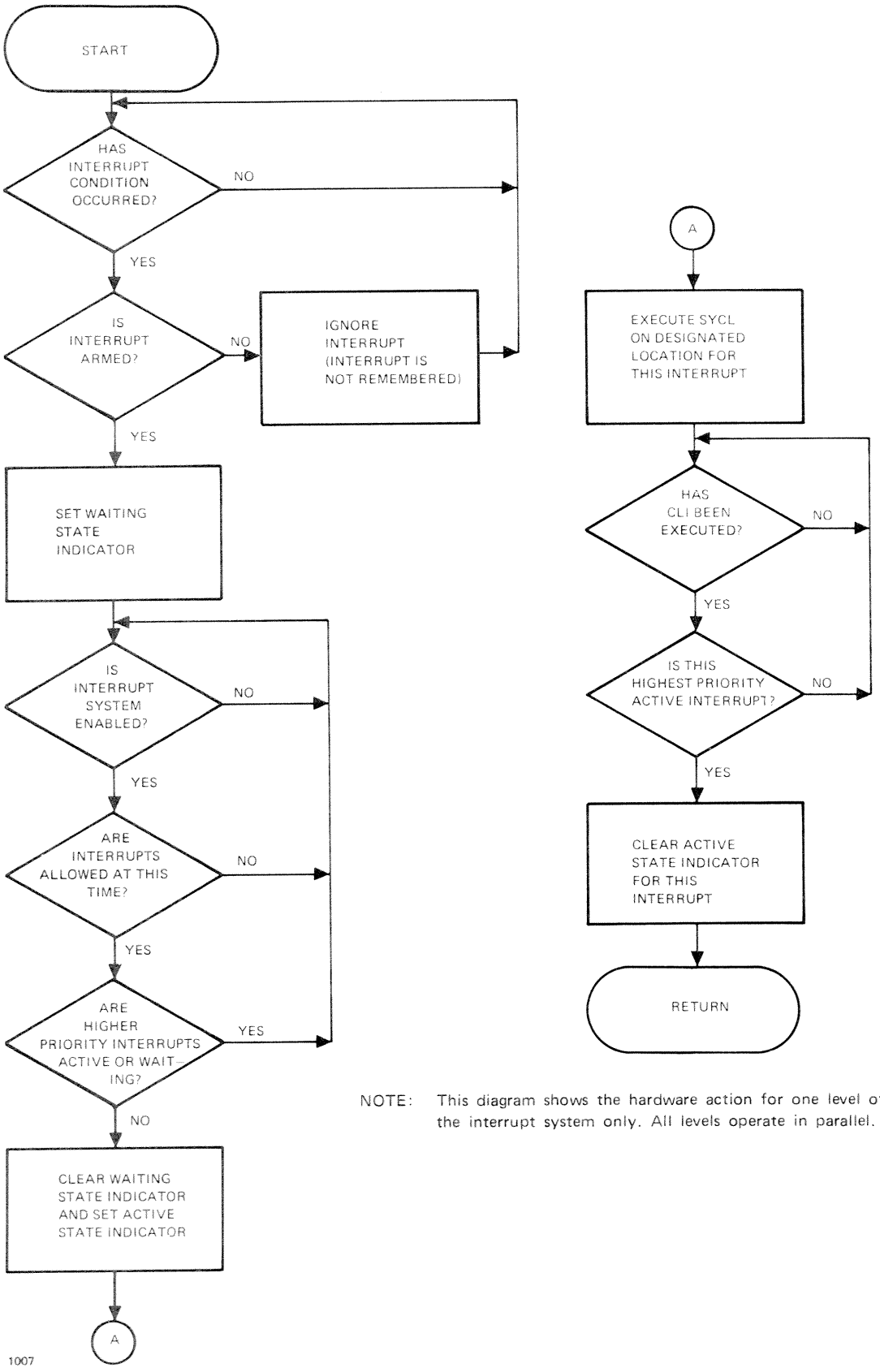
** Microinterrupts which do not affect the program.

***Minimum value, actual value depends upon device response time.

NOTES:

(1) In basic machine, only the unimplemented instruction function of the system trap is operative.

(2) Another microinterrupt is implemented with memory mapping. Called "Map No-match" it is 3rd in priority sequence (inhibits Protection Key Violation). **



NOTE: This diagram shows the hardware action for one level of the interrupt system only. All levels operate in parallel.

Figure 8. Interrupt System Flow Diagram

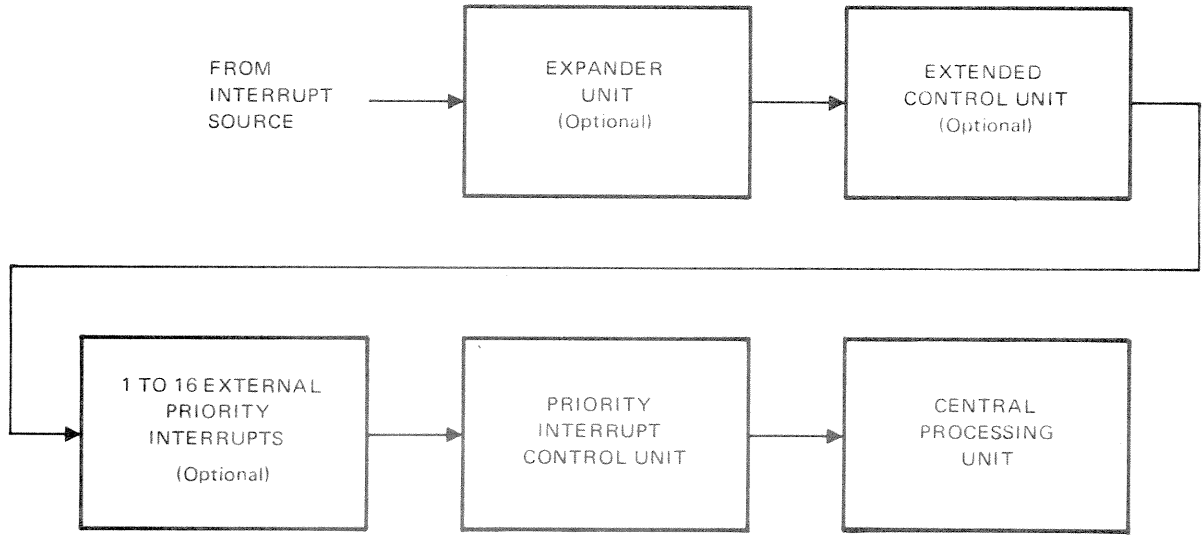
CHANNEL INTERRUPTS

One interrupt level is provided for each of the four channels (multiplexor and selector). There is no individual enabling of the channel interrupts; however, they may be group enabled or disabled by the ENA or DIS instruction. Individual arming or disarming of the channel interrupts is accomplished through the use of the ARM instruction.

If an interrupt occurs on a channel, the program may identify the interrupting device by executing an IIU command. Each device is individually armed or disarmed with an SIO command. A system clear disarms device and channel interrupts. (Refer to sections "Channel Commands", Section IV, and "Control Instructions", Section III). The device must be serviced with the appropriate command to clear its interrupt condition before executing a CLI. This is accomplished by XMT in the single byte mode, IUS, HIO, and master clear unless otherwise specified by the device.

EXTERNAL INTERRUPT SUBSYSTEM

The external interrupt subsystem for the SCC 4700 may include the units shown in Figure 9.

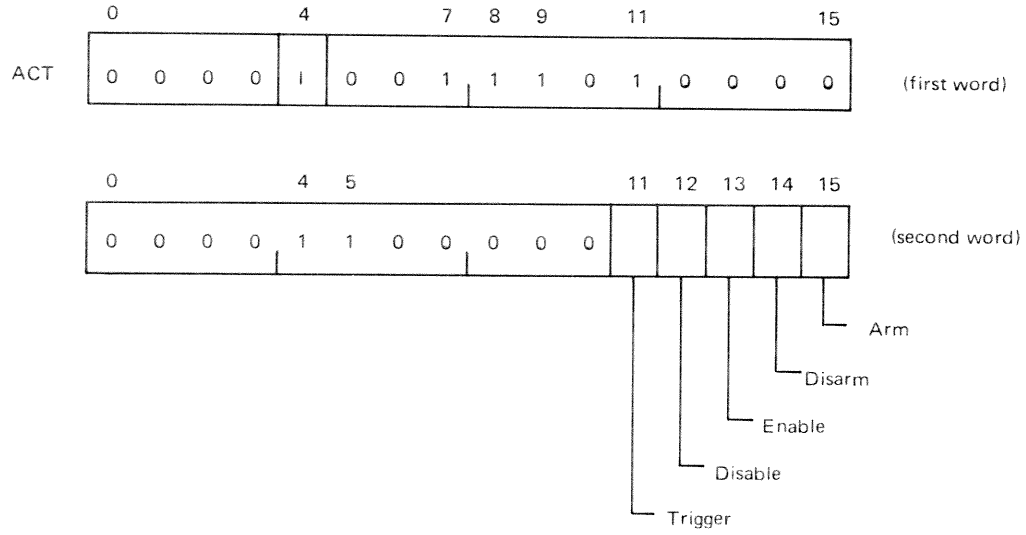


1240

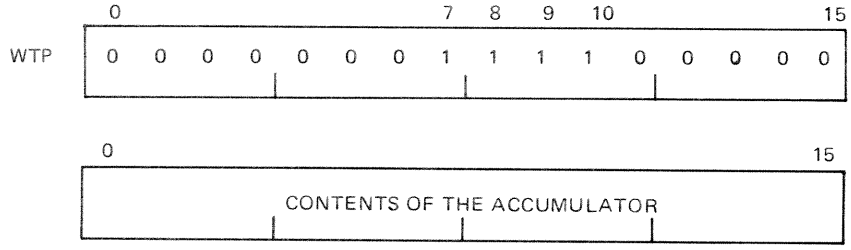
Figure 9. External Interrupt Subsystem

The basic external interrupt subsystem consists of the Priority Interrupt Control Unit which provides an interface to the CPU and the 1 to 16 external priority interrupts. Each interrupt provides one priority interrupt level. Individual arming/disarming, enabling/disabling, and triggering of the external interrupts is available as an option.

The optional Extended Control Unit interfaces with the parallel I/O and adds the capability to individually arm/disarm, enable/disable, and trigger each of the 16 external priority interrupts. Through the use of an Activate (ACT) instruction followed by a Write Parallel (WTP) instruction, the desired interrupt function may be specified. The format for ACT and WTP is as follows.



By setting (=1) the proper bit of the second word of ACT, the desired interrupt functions (to a maximum of three) may be specified.

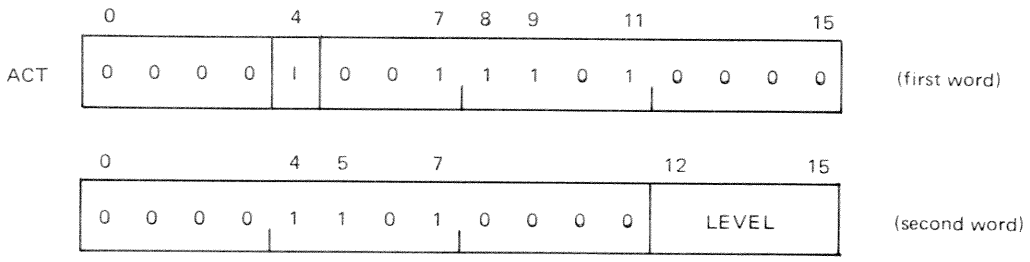


Each of the 16 bits transferred from the accumulator by the WTP instruction specifies a level to be armed, disarmed, enabled, disabled, or triggered.

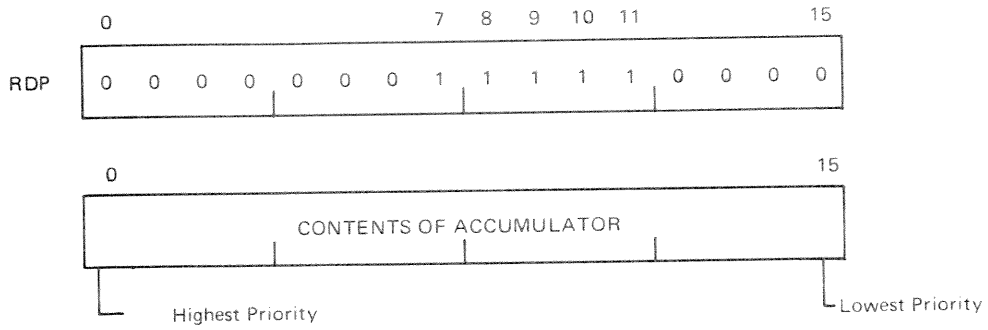
1 = arms, disarms, enables, disables, or triggers (as indicated by ACT).

0 = no action.

The Expander Unit is an optional unit which also interfaces with the parallel I/O. The addition of this unit adds the capability of each interrupt level to service up to 16 devices rather than one. When an interrupt occurs, the programmer identifies the device causing the interrupt by using an ACT and RDP (Read Parallel I/O) pair of instructions. The format for ACT and RDP is as follows.



Bits 12 through 15 of the second word of ACT designate (in binary) the interrupt level to be read.



The indicated interrupt is specified by setting the proper bit of the accumulator.

The LLO (Locate Leading One) instruction may be used to determine in priority order which interrupts on a given level are active.

REAL TIME CLOCK

The Real Time Clock (RTC) decrements location 2 at the local power line frequency if the interrupt system is enabled and the RTC is armed. When location 2 reaches zero, a trap will be made to location 3.

The Real Time Clock may be disarmed or disabled for one-half cycle of the power line frequency without losing clock counts. Disarming the RTC will inhibit the update of displays on the control console. (Refer to Section VII, "Control Contole.")

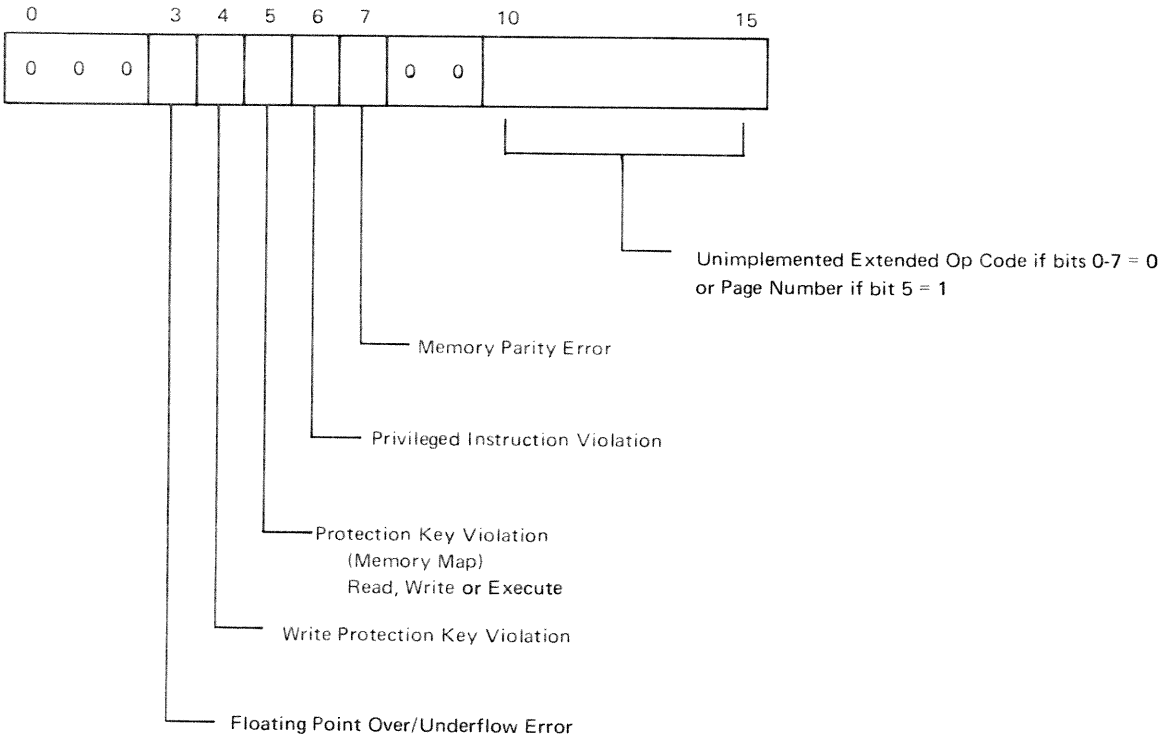
POWER ON INTERRUPT

The Power On interrupt operates differently from other interrupts so as to restore the registers. Instead of simulating the System Call, it generates in hardware and executes System Return (SRT). Interrupt location zero corresponds to the second word of an SRT instruction and should contain the address of the registers.

Without the Power Fail Safe option the machine will halt following the interrupt.

SYSTEM TRAP INDICATOR

Location 5 in core memory is reserved for the system trap indicator. If a system trap occurs, a system call will be executed on location 6. At that time the system trap indicator will be set to show what condition caused the trap.



If bits 3, 4, 5, 6, or 7 are set (=1), it will indicate the condition shown on the diagram. If more than one error condition occurs simultaneously, all of the corresponding bits in the trap indicator will be set.

If the system trap occurs and bits 3, 4, 5, 6, and 7 are all equal to zero, this will indicate that an attempt was made to execute an unimplemented instruction.

Since all 16 possible combinations are utilized in the standard machine for basic operation codes, any unimplemented instruction must be of the extended non-privileged op code set.

The extended non-privileged operation code is contained in bits 7 through 12 of the instruction. When the trap occurs, bits 0 through 7 of the system trap indicator are reset to zero and bits 7 through 12 of the instruction which caused the trap are stored in bits 10 through 15 of the indicator word.

In the basic machine (without memory parity, memory mapping unit, or optional instruction sets) only the unimplemented instruction trap is operative.

The location counter will always contain the address of the instruction causing the system trap, except for floating point over/underflow. In this case, the location counter will contain the address of the next instruction. If a protection key violation occurs (bit 5 = 1), bits 10 through 15 will contain the page number causing the violation.

Memory parity errors may cause loss of data in the programmable registers when the error occurs. Floating point under/overflow causes the contents of the A, B, and E registers to be undefined. Protection key violation, privileged instruction violation, or unimplemented instruction system traps cause loss of data.

SECTION VI

MEMORY MAPPING

INTRODUCTION

Memory mapping is a technique which facilitates operation in a multiprogrammed environment. The concept of virtual and real memory is essential to understanding memory mapping.

> *Virtual* memory for the SCC 4700 is an imaginary memory space of 32K, contiguous, 16-bit words. Virtual memory allows the programmer to write a program on the assumption that he may utilize the entire core memory.
>
> *Real* memory is the actual memory space of the program being executed. Real memory may be larger or smaller than 32K.

Although virtual memory is an abstraction, it is an important concept in the creation of a multiprogramming system since it allows a program to be divided into segments. A segment of virtual memory is called a *page* and contains 512 words or locations. The 32K of virtual memory is divided into 64 pages of 512 words.

Real memory is also divided into 512 word segments which are called *blocks*. Since the maximum size addressable by the SCC 4700 is 64K, there may be 128 blocks in real memory.

By segmenting a program into pages of virtual memory and loading it into blocks of real memory, virtual memory is *mapped* onto the real core memory area. This concept is illustrated in Figure 10.

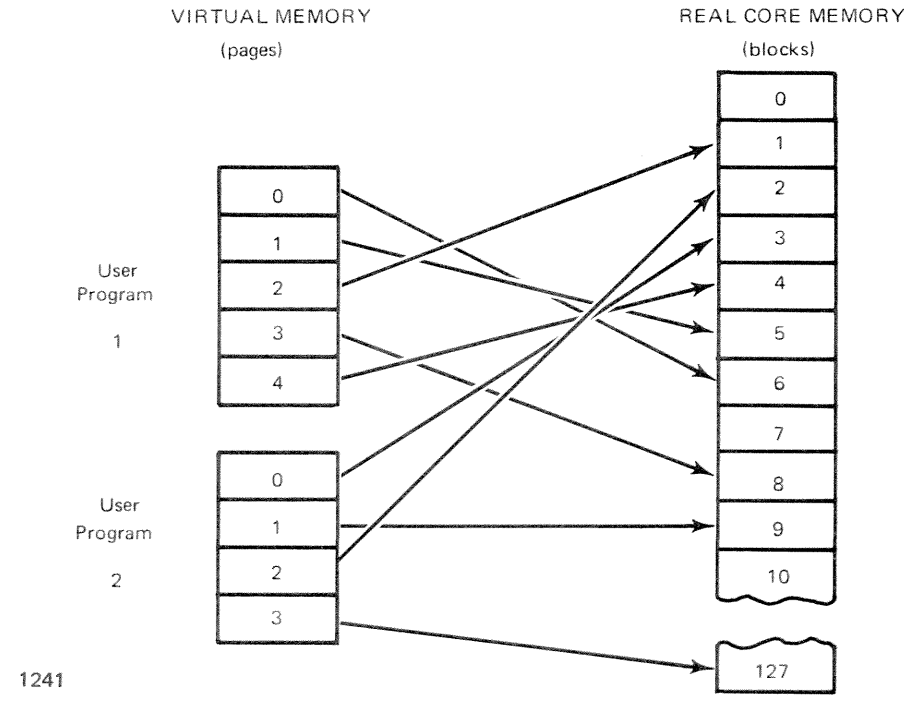


Figure 10. Memory Mapping.

In a multiprogrammed system there will be many programs in operation. Some of these will be totally resident in real memory, others partially resident, and many totally non-resident at any given time. This residency is a dynamically changing variable which is controlled by the system program called the Executive or Real Time Monitor. This program may execute all instructions and uses one or more system memory maps. The various applications programs are known as user programs, each with its own user memory map which is generated by the system program. User programs are restricted to using only those instructions which will not directly affect the system status, i.e., non-privileged instructions to prevent it from interfering with other programs. User programs needing to perform operations which may affect the system status, such as I/O service, request the system program to perform the function by executing a System Call which returns the machine to the system mode. The system program then performs the function, after ensuring that no other program will be affected, and finally returns control to the user program by executing a System Return.

MEMORY MAP OPERATION

General Description

Through the use of associative registers in the memory map unit, the executive system determines which blocks of real memory are "vacant" and maps the pages into these blocks. Blocks of real memory, unlike pages of virtual memory, may not always be in consecutive order (as shown in Figure 10). It is also possible that pages for a program may be located outside the actual core area; i.e., pages may be placed on tape, disc, etc.

The executive system maintains a *page table* for the system and the user which identifies: (1) the block location, and (2) the protection assigned to each page for each program being executed. The protection for each page consists of three bits (R, W, and E) located in each page table entry. Any of these bits may be set (=1) to indicate the operations which may be performed on the page are read, write, or execute. (Bits 0-5 of the page table entry may be used by the system program as desired.)

There are multiple page tables stored in memory as are required; however, only one system and one user page table are used by the CPU at a given time. Page tables are distinguished through the use of *page table pointers* (Figure 11) which contain the starting address (α) of the list of pages for each program.

Each page table must contain 64 entries; however, pages which are not used by the program (or are non-resident) are indicated by setting the protection bits to zero to prevent any access by the program.

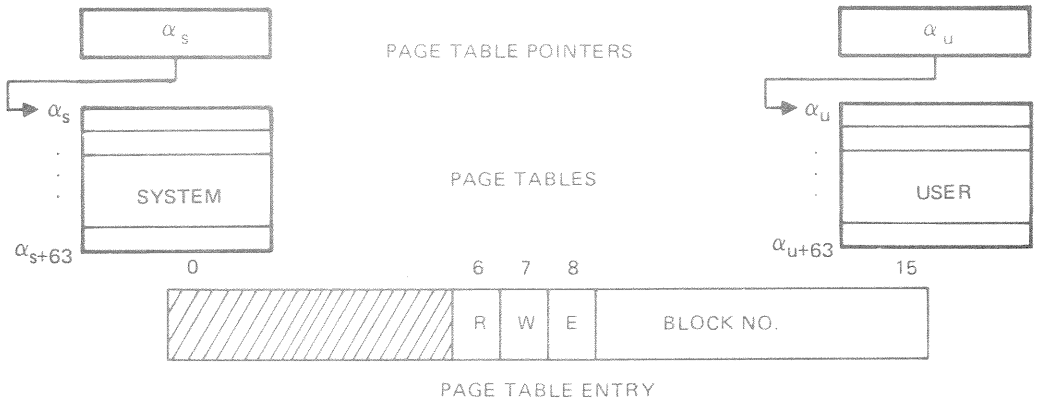


Figure 11. Page Tables and Pointers

Address Transformation

When a program instruction is executed, a 15-bit address is formed and placed in the S register of the CPU. When memory mapping is implemented, every address formed by memory reference instructions must be transformed into a real address before execution (except system page zero, which is always block zero). This transformation is accomplished through the use of associative registers. (Refer to Figure 12.)

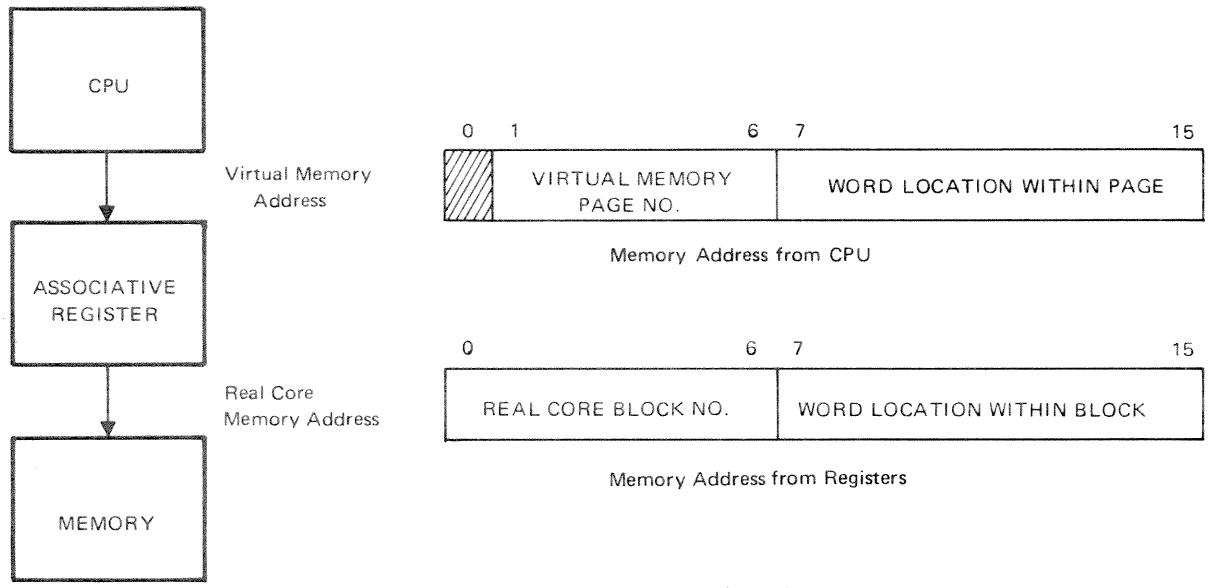


Figure 12. Address Transformation

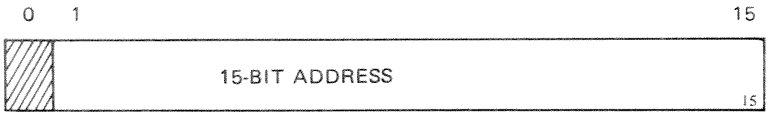
1243

The associative registers are loaded by the memory map unit with (1) an entry in the 64-word page table contained in memory, (2) the page number, and (3) a system or user mode indicator. The associative register, whose page number and mode indicator match the memory request, is selected and its block number is combined with the word location within the page to form the real core address. If no match is found, the memory map unit selects a new associative register entry from the page table.

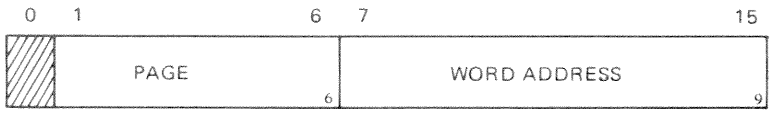
Detailed Memory Map Operation

Memory access is accomplished in the following manner.

1. The CPU generates a memory request to the memory map unit. Also transmitted are indicators specifying type of request (read, write, or execute) and mode (system or user).



2. The memory map unit receives the request and divides the 15-bit address into two parts.



3. The map unit compares the mode and page number of the request simultaneously with the mode and page number in each of the associative registers.

   If a match is not found, the following occurs.

   a. Execution of the instruction is aborted.

   b. The page number of the request is added to the starting address contained in the System Map Table Pointer or User Map Table Pointer register (depending on the setting of the mode bit). This forms the real core address of the map table entry which corresponds to the current page.

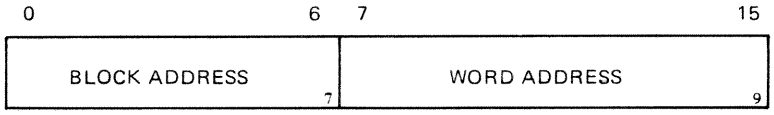
   c. The contents of this address is used to load the next available associative register with the memory protection bits and the block number for the page in question. The page number and the current setting of the mode bit from the page number requested is recorded.

   d. The instruction execution is restarted from the beginning.

4. After a match for the page number is found, the map unit compares the type of request (read, write, or execute) against the memory protection bits (read, write, or execute) of the associative registers. (System page zero does not have protection; i.e., protection bits are not tested.)

   If a match is not found, an interrupt signal is sent to the CPU, which generates a system trap. Since system page zero is not mapped, all interrupts and traps are routed to block zero. The memory request and the instruction execution are aborted.

5. After a match for the request type is accomplished, the block number in the associative register is linked with the 9-bit word address to form a 16-bit memory address.



   Memory is then accessed with this address.

SECTION VII

CONTROL CONSOLE UNIT

GENERAL DESCRIPTION

The control console unit with display panel is 8 inches high, 22-5/16 inches wide, and 5-1/4 inches deep. The console is mounted at a 10-degree slant above a 16-inch by 26-inch writing shelf on the front of the main cabinet of the SCC 4700. If desired, the control console may be mounted as a remote unit. Connected to the Central Processing Unit by up to 30 feet of cable, this optional remote console may also contain a Selectric Typewriter for operator messages and commands. (See Figure 13.)

The operator interface station contains 31 pushbuttons, 2 rotary switches, and a display panel to control the computer system. (Refer to Figure 14.) To aid the operator in identifying the various register and status indicators on the display panel, the following color scheme is implemented.

Yellow	L (register)
White	Reg (select)
Orange	SW (register)
	Carryout (CRO)
	Overflow (OVF)
	Interrupts Disabled (INT)
	User Mode (MODE)
	Indirect User Mode (IMODE)
Red	Halt
	Memory Parity Error (MPE)
Green	Power

This display panel contains 64 lamps (28 volts, 40 milliamperes) which are replaceable from the rear without removing the display panel from the cabinet.

DISPLAY PANEL

The function of the display panel is as follows.

1. STATUS (of CPU)

   The status indicators show the state of certain internal conditions. Each indicator is marked to indicate the corresponding function. When lighted, their color calls the attention of the operator to the internal state of the CPU.

2. L (L register)

   This register display shows the contents of the location counter. It will contain the

Figure 13. Remote Console with Typewriter Unit

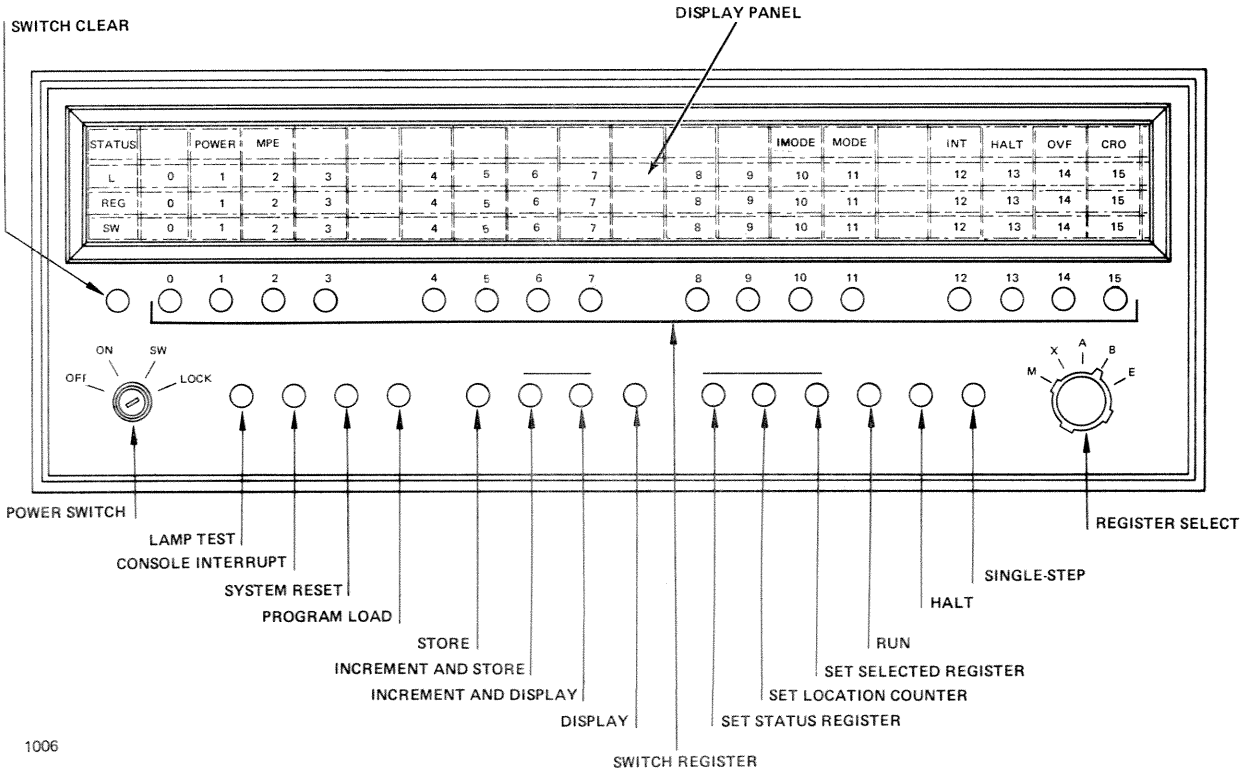


Figure 14. SCC 4700 Control Console

address of the next instruction to be executed. This is the real core address, except when memory mapping is implemented.*

3. REG (Register)

   The desired register is selected by use of the REG SELECT switch. When the DISPLAY button is depressed, the contents of the register indicated by the REG SELECT knob (M, S, A, B, or E) are displayed in this row of indicator lights (ON=1).

   The M register is the path to and from memory. On execution of a halt, the M register contains the next instruction.

   If the M register is selected and the DISPLAY button is pressed, the contents of the address specified by the location counter is displayed in the REG indicators.

4. SW (Switch Register)

   Data to be entered in a register or memory location from the console is placed in the switch register by the operator. These indicators show the current contents of the switch register (ON=1).

   The switch register is an external holding register. Data entered in this register by use of the 16 console switches is retained in the switch register until changed by the operator. The register may be changed or cleared as necessary without affecting either memory or the registers of the CPU.

The contents of the display registers are normally updated at the line voltage rate (50 to 60 cycles) while the CPU is in the run mode. It is possible to inhibit the update of the displays either manually by a switch on the maintenance panel inside the cabinet, or under program control by disarming the Real Time Clock or by disabling interrupts.

If the display update is inhibited, the register displays will be updated only on execution of a halt and when data is entered or displayed by use of the control console.

If display update is not inhibited, the M register will contain the current reading of the Real Time Clock. (This is equal to the contents of Location 2.)

If memory mapping is not implemented, all addresses displayed will be real core addresses.

If the memory mapping option is implemented, than addresses displayed will normally be virtual memory addresses (or addresses before mapping is performed); however, provision has been made to use real core address when desired*.

---

*A switch on the maintenance panel inside the machine may be used to inhibit mapping.

CONTROL CONSOLE

The following pushbuttons are operative while the CPU is in the run mode and the power switch is in ON position:

LAMP TEST	REG SELECT
CONSOLE INTERRUPT	POWER
SYSTEM RESET	RUN
HALT	The 16 switches of the Switch Register.

The operation and function of each control console button (from left to right) is as follows.

1. Switch Register

   a. SW CLEAR (Switch Register Clear)

      Resets (=0) entire switch register and all switch register indicators on display panel (OFF=0).

   b. Depressing one of these 16 buttons will set (=1) the corresponding bit of the switch register and illuminate the indicator in the SW row of indicators (ON=1) for that binary position. Contents of the Switch Register may be entered into one of the other registers or a memory location from the control console, or into the accumulator under program control. (See instruction description of Load Accumulator from Switches.)

2. POWER (Rotary Key Switch)

   a. OFF – all power supplies off.

   b. ON – all power supplies on; master clear of all internal and external conditions when bringing power up occurs automatically. All console switches are operative.

   c. SW – only the following switches are operative in either run or halt mode:

      SW CLEAR

      Switch Register

      LAMP TEST

      CONSOLE INTERRUPT.

      Data may be entered into Switch Register for program modification and control, but registers and memory locations cannot be altered or examined from the control console.

   d. LOCK – Power is on and all console switches except lamp test are inoperative. Program execution cannot be interrupted or data entered from the console.

3. LAMP TEST

Supplies power to all indicator lamps on display panel at same time regardless of state. No data is altered nor functions affected. This switch is always operative as long as machine power is on.

4. CONSOLE INTERRUPT

Causes a trap to location 4 to occur. The console interrupt will be serviced by the software routine written to handle this condition for the system being executed. If the console interrupt occurs while the CPU is in the halt mode, the CPU will transfer to the run mode.

5. SYSTEM RESET

Operative only when power switch is in ON position. This causes all the indicators and switch settings to be reset. It is the master clear signal which will initialize the central processor and all channel interrupts.

The interrupt system will be disabled and all interrupts disarmed. The status register will be reset, except for Halt and Interrupts Disabled, which will be set.

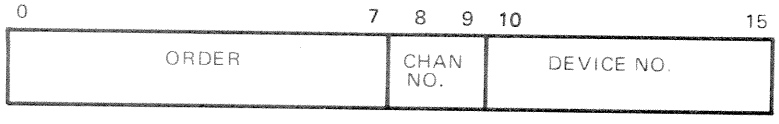
Data in core memory and the programmable registers (A, B, X, E and L) will not be changed unless system reset is activated while in the run mode.

6. PROGRAM LOAD

Operative only when power switch is in ON position; disabled in the run mode.

Programs may be loaded from any device connected to a Multiplexor or Selector channel which provides initial program loading. This switch automatically stores the necessary control information in memory and initiates the channel to load a 70-word "bootstrap" loader from the selected device. When the 70 words are loaded, control is transferred to the "bootstrap" routine to complete the load operation under program control.

Prior to using this switch, the operator should place the loader program on the device, perform a System Reset, set the starting address in which the "bootstrap" is to be loaded in the L register, and enter the device order, channel number, and device number in the switch register as shown.



OPERATOR ENTRY IN SWITCH REGISTER

When specifing a loading address less than 284_{10} in the L register, the locations to be loaded must not conflict with the control information loaded in memory. The memory locations affected are shown in Figure 15.

When load control is passed to the "bootstrap," the system is in an interrupt state. The "bootstrap" should clear this interrupt by transmitting an IUS command to the device followed by a CLI instruction. The status byte may be checked for device errors.

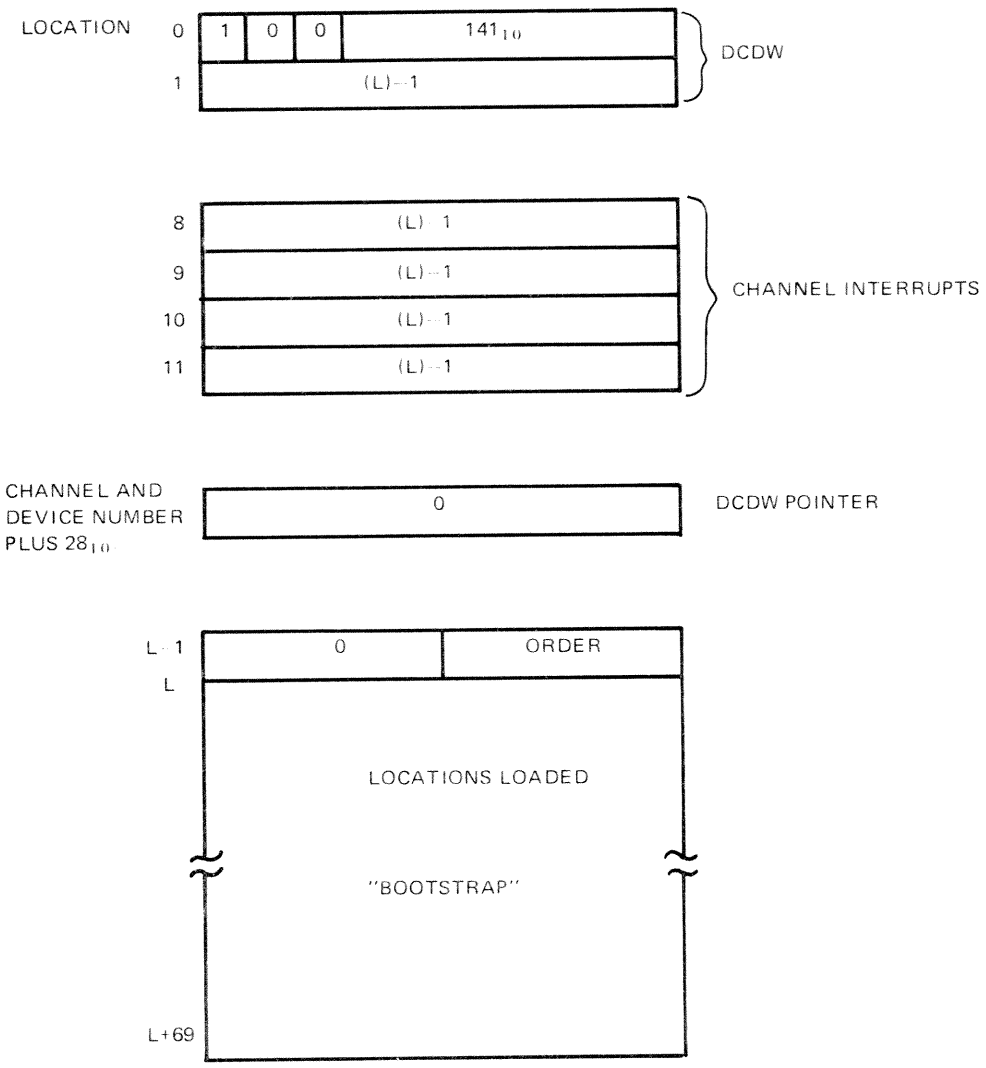


Figure 15 – Initial Program Load

7. STORE

Operative only when power switch is in ON position; disabled in the run mode. Used to store data from the switch register into the address specified by the L register.

Since data stored must pass through the M register, the data last transferred to memory can be displayed in the REG row of indicators by selecting the M register.

8. INCREMENT and STORE

Operative only when power switch is in ON position; disabled in run mode. The L register is incremented by one (L+1), and the contents of the switch register are stored into the address specified by the new value of the location counter.

9. INCREMENT and DISPLAY

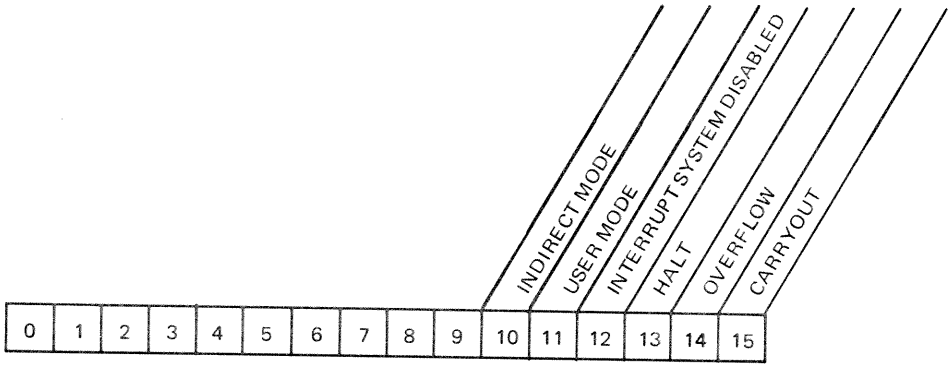
Operative only when power switch is in ON position; disabled in the run mode. The L register is incremented by one (L+1), and the contents of the memory location specified by the new value of the location counter are brought into the M register. This data may be displayed in the REG row of indicators by selecting the M register with the REG SELECT knob.

10. DISPLAY

Operative only when the power switch is in ON position; disabled in the run mode. This switch brings the contents of the memory address specified in the L register into the M register. This data may be displayed in the REG row of indicators by selecting the M register with the REG SELECT knob.

11. SET STATUS (Register)

Operative only when the power switch is in ON position; disabled in the run mode. This switch transfers the contents of the switch registers to the corresponding indicators in the status register. The format of the status register is:



The Halt indicator cannot be set (=1) or reset from the switch register. It can be set only by the HALT button, Halt instruction and SYSTEM RESET button, and reset only by the RUN button or an interrupt.

12. SET L (register)

Operative only when the power switch is in ON position; disabled in the run mode. This switch transfers the contents of the switch register to the location counter.

13. SET REG (indicated by REG SELECT)

Operative only when the power switch is in ON position; disabled in the run mode. This switch transfers the contents of the switch register to the register indicated by the setting of the REG SELECT knob. They are:

M Memory register

X Index

A Accumulator

B Extended Accumulator

E Exponent register (optional)

14. RUN

Operative only when the power switch is in ON position. Resets halt indicator and puts machine in run mode to execute the program, beginning with the instruction in the location addressed by the L register.

15. HALT

Operative only when the power switch in is ON position. This switch stops program execution, sets (=1) the halt indicator and updates the display register.

Only the CPU is halted; it will not affect input/output devices which are running. The CPU is in a "wait" state and may be interrupted from a halt by the following interrupts:

Power off

Power on

Console interrupt

Channel interrupts

External interrupts

Real Time Clock (traps on zero)

These interrupts, except Power On without Power Fail Safe, cause the machine to be placed in the run mode. The RTC update does not cause the machine to run, unless the trap location goes to zero.

16. STEP

Operative only when the power switch is in ON position; disabled in the run mode. The STEP button causes the machine to execute only the instruction contained in the address specified by the L register and then halt.

17. REG SELECT

By using this switch, the operator may select which register (M, X, A, B or E) is to be displayed in the REG row of indicators, or the register into which data will be entered by the SET REG button.

APPENDIX A
PERIPHERAL DEVICE CODES

(700 Series)

INPUT			OUTPUT			REMARKS
OCTAL	HEXA-DECIMAL	FUNCTION	OCTAL	HEXA-DECIMAL	FUNCTION	STANDARD DEVICES
00	00		01	01		
02	02	Read Storage No. 1	03	03	Write Storage No. 1	Drum or Disk
04	04	Keyboard No. 1	05	05	Printer No. 1	Teletype
06	06	Tape Reader No. 1	07	07	Tape Punch No. 1	ASR 33
10	08	Byte Input No. 1	11	09	Byte Output No. 1	
12	0A	Read Storage No. 2	13	0B	Write Storage No. 2	Drum or Disk
14	0C	Keyboard No. 2	15	0D	Printer No. 2	Teletype
16	0E	Tape Reader No. 2	17	0F	Tape Punch No. 2	ASR 35
20	10		21	11	Line Printer	
22	12		23	13	X-Y Plotter No. 1	
24	14	Keyboard No. 3	25	15	Printer No. 3	IBM Selectric
26	16	Card Reader	27	17	Card Punch	
30	18	Byte Input No. 2	31	19	Byte Output No. 2	
32	1A		33	1B	X-Y Plotter No. 2	
34	1C		35	1D		
36	1E		37	1F		
40	20	Incremental Tape Read A	41	21	Incremental Tape Write A	
42	22	Analog - Digital Converter (in)	43	23	Digital - Analog Converter (out)	
44	24	System in A No. 1	45	25	System Out A No. 1	Unique Device
46	26	System in B No. 1	47	27	System Out B No. 1	(Controller Oriented)
50	28	Byte Input No. 3	51	29	Byte Output No. 3	
52	2A		53	2B		
54	2C	System in A No. 2	55	2D	System Out A No. 2	Unique Device
56	2E	System in B No. 2	57	2F	System Out B No. 2	(Controller Oriented)
60	30	Magnetic Tape A No. 1 Read	61	31	Magnetic Tape A No. 1 Write	
62	32	Magnetic Tape B No. 1 Read	63	33	Magnetic Tape B No. 1 Write	
64	34	Magnetic Tape C No. 1 Read	65	35	Magnetic Tape C No. 1 Write	
66	36	Magnetic Tape D No. 1 Read	67	37	Magnetic Tape D No. 1 Write	
70	38	Magnetic Tape A No. 2 Read	71	39	Magnetic Tape A No. 2 Write	
72	3A	Magnetic Tape B No. 2 Read	73	3B	Magnetic Tape B No. 2 Write	
74	3C	Magnetic Tape C No. 2 Read	75	3D	Magnetic Tape C No. 2 Write	
76	3E	Magnetic Tape D No. 2 Read	77	3F	Magnetic Tape D No. 2 Write	

APPENDIX B
HEXADECIMAL ARITHMETIC

ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

MULTIPLICATION TABLE

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

APPENDIX C
TABLE OF POWERS OF 16_{10}
EXPRESSED IN DECIMAL

16^n	n	16^{-n}
1	0	$0.10000\ 00000\ 00000\ 00000 \times 10$
16	1	$0.62500\ 00000\ 00000\ 00000 \times 10^{-1}$
256	2	$0.39062\ 50000\ 00000\ 00000 \times 10^{-2}$
4 096	3	$0.24414\ 06250\ 00000\ 00000 \times 10^{-3}$
65 536	4	$0.15258\ 78906\ 25000\ 00000 \times 10^{-4}$
1 048 576	5	$0.95367\ 43164\ 06250\ 00000 \times 10^{-6}$
16 777 216	6	$0.59604\ 64477\ 53906\ 25000 \times 10^{-7}$
268 435 456	7	$0.37252\ 90298\ 46191\ 40625 \times 10^{-8}$
4 294 967 296	8	$0.23283\ 06436\ 53869\ 62891 \times 10^{-9}$
68 719 476 736	9	$0.14551\ 91522\ 83668\ 51807 \times 10^{-10}$
1 099 511 627 776	10	$0.90949\ 47017\ 72928\ 23792 \times 10^{-12}$
17 592 186 044 416	11	$0.56843\ 41886\ 08080\ 14870 \times 10^{-13}$
281 474 976 710 656	12	$0.35527\ 13678\ 80050\ 09294 \times 10^{-14}$
4 503 599 627 370 496	13	$0.22204\ 46049\ 25031\ 30808 \times 10^{-15}$
72 057 594 037 927 936	14	$0.13877\ 78780\ 78144\ 56755 \times 10^{-16}$
1 152 921 504 606 846 976	15	$0.86736\ 17379\ 88403\ 54721 \times 10^{-18}$

APPENDIX D
TABLE OF POWERS OF 10_{10}
EXPRESSED IN HEXADECIMAL

10^n	n	10^{-n}					
1	0	1.0000	0000	0000	0000		
A	1	0.1999	9999	9999	999A		
64	2	0.28F5	C28F	5C28	F5C3	$\times$	16^{-1}
3E8	3	0.4189	374B	C6A7	EF9E	$\times$	16^{-2}
2710	4	0.68DB	8BAC	710C	B296	$\times$	16^{-3}
1 86A0	5	0.A7C5	AC47	1B47	8423	$\times$	16^{-4}
F 4240	6	0.10C6	F7A0	B5ED	8D37	$\times$	16^{-4}
98 9680	7	0.1AD7	F29A	BCAF	4858	$\times$	16^{-5}
5F5 E100	8	0.2AF3	1DC4	6118	73BF	$\times$	16^{-6}
3B9A CA00	9	0.44B8	2FA0	9B5A	52CC	$\times$	16^{-7}
2 540B E400	10	0.6DF3	7F67	5EF6	EADF	$\times$	16^{-8}
17 4876 E800	11	0.AFEB	FF0B	CB24	AAFF	$\times$	16^{-9}
E8 D4A5 1000	12	0.1197	9981	2DEA	1119	$\times$	16^{-9}
918 4E72 A000	13	0.1C25	C268	4976	81C2	$\times$	16^{-10}
5AF3 107A 4000	14	0.2D09	370D	4257	3604	$\times$	16^{-11}
3 8D7E A4C6 8000	15	0.4801	BE7B	9D58	566D	$\times$	16^{-12}
23 8652 6FC1 0000	16	0.734A	CA51	6226	F0AE	$\times$	16^{-13}
163 4578 5D8A 0000	17	0.B877	AA32	36A4	B449	$\times$	16^{-14}
DE0 B6B3 A764 0000	18	0.1272	5DD1	D243	ABA1	$\times$	16^{-14}
8AC7 2304 89E8 0000	19	0.1D83	C94F	B6D2	AC35	$\times$	16^{-15}

APPENDIX E

HEXADECIMAL-DECIMAL CONVERSION TABLE

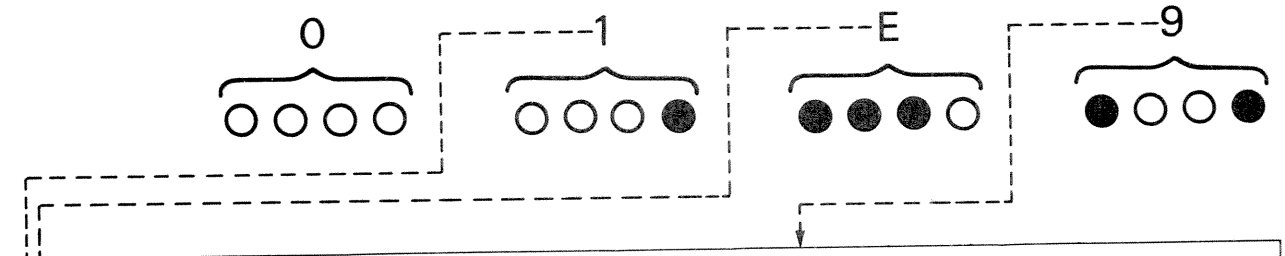
The table in this appendix provides for direct conversion of decimal and hexadecimal numbers in these ranges:

HEXADECIMAL	DECIMAL
000 to FFF	0000 to 4095

For numbers outside the range of the table, add the following values to the table figures:

HEXADECIMAL	DECIMAL
1000	4096
2000	8192
3000	12288
4000	16384
5000	20484
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	53248
E000	57344
F000	61440

HEXADECIMAL-DECIMAL CONVERSION TABLE (cont.)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

HEXADECIMAL-DECIMAL CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

HEXADECIMAL-DECIMAL CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

HEXADECIMAL-DECIMAL CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E00	3584	3585	3586	3587	3583	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

APPENDIX F
HEXADECIMAL–DECIMAL FRACTION CONVERSION TABLE

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.40 00 00 00	.25000 00000	.80 00 00 00	.50000 00000	.C0 00 00 00	.75000 00000
.01 00 00 00	.00390 62500	.41 00 00 00	.25390 62500	.81 00 00 00	.50390 62500	.C1 00 00 00	.75390 62500
.02 00 00 00	.00781 25000	.42 00 00 00	.25781 25000	.82 00 00 00	.50781 25000	.C2 00 00 00	.75781 25000
.03 00 00 00	.01171 87500	.43 00 00 00	.26171 87500	.83 00 00 00	.51171 87500	.C3 00 00 00	.76171 87500
.04 00 00 00	.01562 50000	.44 00 00 00	.26562 50000	.84 00 00 00	.51562 50000	.C4 00 00 00	.76562 50000
.05 00 00 00	.01953 12500	.45 00 00 00	.26953 12500	.85 00 00 00	.51953 12500	.C5 00 00 00	.76953 12500
.06 00 00 00	.02343 75000	.46 00 00 00	.27343 75000	.86 00 00 00	.52343 75000	.C6 00 00 00	.77343 75000
.07 00 00 00	.02734 37500	.47 00 00 00	.27734 37500	.87 00 00 00	.52734 37500	.C7 00 00 00	.77734 37500
.08 00 00 00	.03125 00000	.48 00 00 00	.28125 00000	.88 00 00 00	.53125 00000	.C8 00 00 00	.78125 00000
.09 00 00 00	.03515 62500	.49 00 00 00	.28515 62500	.89 00 00 00	.53515 62500	.C9 00 00 00	.78515 62500
.0A 00 00 00	.03906 25000	.4A 00 00 00	.28906 25000	.8A 00 00 00	.53906 25000	.CA 00 00 00	.78906 25000
.0B 00 00 00	.04296 87500	.4B 00 00 00	.29296 87500	.8B 00 00 00	.54296 87500	.CB 00 00 00	.79296 87500
.0C 00 00 00	.04687 50000	.4C 00 00 00	.29687 50000	.8C 00 00 00	.54687 50000	.CC 00 00 00	.79687 50000
.0D 00 00 00	.05078 12500	.4D 00 00 00	.30078 12500	.8D 00 00 00	.55078 12500	.CD 00 00 00	.80078 12500
.0E 00 00 00	.05468 75000	.4E 00 00 00	.30468 75000	.8E 00 00 00	.55468 75000	.CE 00 00 00	.80468 75000
.0F 00 00 00	.05859 37500	.4F 00 00 00	.30859 37500	.8F 00 00 00	.55859 37500	.CF 00 00 00	.80859 37500
.10 00 00 00	.06250 00000	.50 00 00 00	.31250 00000	.90 00 00 00	.56250 00000	.D0 00 00 00	.81250 00000
.11 00 00 00	.06640 62500	.51 00 00 00	.31640 62500	.91 00 00 00	.56640 62500	.D1 00 00 00	.81640 62500
.12 00 00 00	.07031 25000	.52 00 00 00	.32031 25000	.92 00 00 00	.57031 25000	.D2 00 00 00	.82031 25000
.13 00 00 00	.07421 87500	.53 00 00 00	.32421 87500	.93 00 00 00	.57421 87500	.D3 00 00 00	.82421 87500
.14 00 00 00	.07812 50000	.54 00 00 00	.32812 50000	.94 00 00 00	.57812 50000	.D4 00 00 00	.82812 50000
.15 00 00 00	.08203 12500	.55 00 00 00	.33203 12500	.95 00 00 00	.58203 12500	.D5 00 00 00	.83203 12500
.16 00 00 00	.08593 75000	.56 00 00 00	.33593 75000	.96 00 00 00	.58593 75000	.D6 00 00 00	.83593 75000
.17 00 00 00	.08984 37500	.57 00 00 00	.33984 37500	.97 00 00 00	.58984 37500	.D7 00 00 00	.83984 37500
.18 00 00 00	.09375 00000	.58 00 00 00	.34375 00000	.98 00 00 00	.59375 00000	.D8 00 00 00	.84375 00000
.19 00 00 00	.09765 62500	.59 00 00 00	.34765 62500	.99 00 00 00	.59765 62500	.D9 00 00 00	.84765 62500
.1A 00 00 00	.10156 25000	.5A 00 00 00	.35156 25000	.9A 00 00 00	.60156 25000	.DA 00 00 00	.85156 25000
.1B 00 00 00	.10546 87500	.5B 00 00 00	.35546 87500	.9B 00 00 00	.60546 87500	.DB 00 00 00	.85546 87500
.1C 00 00 00	.10937 50000	.5C 00 00 00	.35937 50000	.9C 00 00 00	.60937 50000	.DC 00 00 00	.85937 50000
.1D 00 00 00	.11328 12500	.5D 00 00 00	.36328 12500	.9D 00 00 00	.61328 12500	.DD 00 00 00	.86328 12500
.1E 00 00 00	.11718 75000	.5E 00 00 00	.36718 75000	.9E 00 00 00	.61718 75000	.DE 00 00 00	.86718 75000
.1F 00 00 00	.12109 37500	.5F 00 00 00	.37109 37500	.9F 00 00 00	.62109 37500	.DF 00 00 00	.87109 37500
.20 00 00 00	.12500 00000	.60 00 00 00	.37500 00000	.A0 00 00 00	.62500 00000	.E0 00 00 00	.87500 00000
.21 00 00 00	.12890 62500	.61 00 00 00	.37890 62500	.A1 00 00 00	.62890 62500	.E1 00 00 00	.87890 62500
.22 00 00 00	.13281 25000	.62 00 00 00	.38281 25000	.A2 00 00 00	.63281 25000	.E2 00 00 00	.88281 25000
.23 00 00 00	.13671 87500	.63 00 00 00	.38671 87500	.A3 00 00 00	.63671 87500	.E3 00 00 00	.88671 87500
.24 00 00 00	.14062 50000	.64 00 00 00	.39062 50000	.A4 00 00 00	.64062 50000	.E4 00 00 00	.89062 50000
.25 00 00 00	.14453 12500	.65 00 00 00	.39453 12500	.A5 00 00 00	.64453 12500	.E5 00 00 00	.89453 12500
.26 00 00 00	.14843 75000	.66 00 00 00	.39843 75000	.A6 00 00 00	.64843 75000	.E6 00 00 00	.89843 75000
.27 00 00 00	.15234 37500	.67 00 00 00	.40234 37500	.A7 00 00 00	.65234 37500	.E7 00 00 00	.90234 37500
.28 00 00 00	.15625 00000	.68 00 00 00	.40625 00000	.A8 00 00 00	.65625 00000	.E8 00 00 00	.90625 00000
.29 00 00 00	.16015 62500	.69 00 00 00	.41015 62500	.A9 00 00 00	.66015 62500	.E9 00 00 00	.91015 62500
.2A 00 00 00	.16406 25000	.6A 00 00 00	.41406 25000	.AA 00 00 00	.66406 25000	.EA 00 00 00	.91406 25000
.2B 00 00 00	.16796 87500	.6B 00 00 00	.41796 87500	.AB 00 00 00	.66796 87500	.EB 00 00 00	.91796 87500
.2C 00 00 00	.17187 50000	.6C 00 00 00	.42187 50000	.AC 00 00 00	.67187 50000	.EC 00 00 00	.92187 50000
.2D 00 00 00	.17578 12500	.6D 00 00 00	.42578 12500	.AD 00 00 00	.67578 12500	.ED 00 00 00	.92578 12500
.2E 00 00 00	.17968 75000	.6E 00 00 00	.42968 75000	.AE 00 00 00	.67968 75000	.EE 00 00 00	.92968 75000
.2F 00 00 00	.18359 37500	.6F 00 00 00	.43359 37500	.AF 00 00 00	.68359 37500	.EF 00 00 00	.93359 37500
.30 00 00 00	.18750 00000	.70 00 00 00	.43750 00000	.B0 00 00 00	.68750 00000	.F0 00 00 00	.93750 00000
.31 00 00 00	.19140 62500	.71 00 00 00	.44140 62500	.B1 00 00 00	.69140 62500	.F1 00 00 00	.94140 62500
.32 00 00 00	.19531 25000	.72 00 00 00	.44531 25000	.B2 00 00 00	.69531 25000	.F2 00 00 00	.94531 25000
.33 00 00 00	.19921 87500	.73 00 00 00	.44921 87500	.B3 00 00 00	.69921 87500	.F3 00 00 00	.94921 87500
.34 00 00 00	.20312 50000	.74 00 00 00	.45312 50000	.B4 00 00 00	.70312 50000	.F4 00 00 00	.95312 50000
.35 00 00 00	.20703 12500	.75 00 00 00	.45703 12500	.B5 00 00 00	.70703 12500	.F5 00 00 00	.95703 12500
.36 00 00 00	.21093 75000	.76 00 00 00	.46093 75000	.B6 00 00 00	.71093 75000	.F6 00 00 00	.96093 75000
.37 00 00 00	.21484 37500	.77 00 00 00	.46484 37500	.B7 00 00 00	.71484 37500	.F7 00 00 00	.96484 37500
.38 00 00 00	.21875 00000	.78 00 00 00	.46875 00000	.B8 00 00 00	.71875 00000	.F8 00 00 00	.96875 00000
.39 00 00 00	.22265 62500	.79 00 00 00	.47265 62500	.B9 00 00 00	.72265 62500	.F9 00 00 00	.97265 62500
.3A 00 00 00	.22656 25000	.7A 00 00 00	.47656 25000	.BA 00 00 00	.72656 25000	.FA 00 00 00	.97656 25000
.3B 00 00 00	.23046 87500	.7B 00 00 00	.48046 87500	.BB 00 00 00	.73046 87500	.FB 00 00 00	.98046 87500
.3C 00 00 00	.23437 50000	.7C 00 00 00	.48437 50000	.BC 00 00 00	.73437 50000	.FC 00 00 00	.98437 50000
.3D 00 00 00	.23828 12500	.7D 00 00 00	.48828 12500	.BD 00 00 00	.73828 12500	.FD 00 00 00	.98828 12500
.3E 00 00 00	.24218 75000	.7E 00 00 00	.49218 75000	.BE 00 00 00	.74218 75000	.FE 00 00 00	.99218 75000
.3F 00 00 00	.24609 37500	.7F 00 00 00	.49609 37500	.BF 00 00 00	.74609 37500	.FF 00 00 00	.99609 37500

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 40 00 00	.00097 65625	.00 80 00 00	.00195 31250	.00 C0 00 00	.00292 96875
.00 01 00 00	.00001 52587	.00 41 00 00	.00099 18212	.00 81 00 00	.00196 83837	.00 C1 00 00	.00294 49462
.00 02 00 00	.00003 05175	.00 42 00 00	.00100 70800	.00 82 00 00	.00198 36425	.00 C2 00 00	.00296 02050
.00 03 00 00	.00004 57763	.00 43 00 00	.00102 23388	.00 83 00 00	.00199 89013	.00 C3 00 00	.00297 54638
.00 04 00 00	.00006 10351	.00 44 00 00	.00103 75976	.00 84 00 00	.00201 41601	.00 C4 00 00	.00299 07226
.00 05 00 00	.00007 62939	.00 45 00 00	.00105 28564	.00 85 00 00	.00202 94189	.00 C5 00 00	.00300 59814
.00 06 00 00	.00009 15527	.00 46 00 00	.00106 81152	.00 86 00 00	.00204 46777	.00 C6 00 00	.00302 12402
.00 07 00 00	.00010 68115	.00 47 00 00	.00108 33740	.00 87 00 00	.00205 99365	.00 C7 00 00	.00303 64990
.00 08 00 00	.00012 20703	.00 48 00 00	.00109 86328	.00 88 00 00	.00207 51953	.00 C8 00 00	.00305 17578
.00 09 00 00	.00013 73291	.00 49 00 00	.00111 38916	.00 89 00 00	.00209 04541	.00 C9 00 00	.00306 70166
.00 0A 00 00	.00015 25878	.00 4A 00 00	.00112 91503	.00 8A 00 00	.00210 57128	.00 CA 00 00	.00308 22753
.00 0B 00 00	.00016 78466	.00 4B 00 00	.00114 44091	.00 8B 00 00	.00212 09716	.00 CB 00 00	.00309 75341
.00 0C 00 00	.00018 31054	.00 4C 00 00	.00115 96679	.00 8C 00 00	.00213 62304	.00 CC 00 00	.00311 27929
.00 0D 00 00	.00019 83642	.00 4D 00 00	.00117 49267	.00 8D 00 00	.00215 14892	.00 CD 00 00	.00312 80517
.00 0E 00 00	.00021 36230	.00 4E 00 00	.00119 01855	.00 8E 00 00	.00216 67480	.00 CE 00 00	.00314 33105
.00 0F 00 00	.00022 88818	.00 4F 00 00	.00120 54443	.00 8F 00 00	.00218 20068	.00 CF 00 00	.00315 85693
.00 10 00 00	.00024 41406	.00 50 00 00	.00122 07031	.00 90 00 00	.00219 72656	.00 D0 00 00	.00317 38281
.00 11 00 00	.00025 93994	.00 51 00 00	.00123 59619	.00 91 00 00	.00221 25244	.00 D1 00 00	.00318 90869
.00 12 00 00	.00027 46582	.00 52 00 00	.00125 12207	.00 92 00 00	.00222 77832	.00 D2 00 00	.00320 43457
.00 13 00 00	.00028 99169	.00 53 00 00	.00126 64794	.00 93 00 00	.00224 30419	.00 D3 00 00	.00321 96044
.00 14 00 00	.00030 51757	.00 54 00 00	.00128 17382	.00 94 00 00	.00225 83007	.00 D4 00 00	.00323 48632
.00 15 00 00	.00032 04345	.00 55 00 00	.00129 69970	.00 95 00 00	.00227 35595	.00 D5 00 00	.00325 01220
.00 16 00 00	.00033 56933	.00 56 00 00	.00131 22558	.00 96 00 00	.00228 88183	.00 D6 00 00	.00326 53808
.00 17 00 00	.00035 09521	.00 57 00 00	.00132 75146	.00 97 00 00	.00230 40771	.00 D7 00 00	.00328 06396
.00 18 00 00	.00036 62109	.00 58 00 00	.00134 27734	.00 98 00 00	.00231 93359	.00 D8 00 00	.00329 58984
.00 19 00 00	.00038 14697	.00 59 00 00	.00135 80322	.00 99 00 00	.00233 45947	.00 D9 00 00	.00331 11572
.00 1A 00 00	.00039 67285	.00 5A 00 00	.00137 32910	.00 9A 00 00	.00234 98535	.00 DA 00 00	.00332 64160
.00 1B 00 00	.00041 19873	.00 5B 00 00	.00138 85498	.00 9B 00 00	.00236 51123	.00 DB 00 00	.00334 16748
.00 1C 00 00	.00042 72460	.00 5C 00 00	.00140 38085	.00 9C 00 00	.00238 03710	.00 DC 00 00	.00335 69335
.00 1D 00 00	.00044 25048	.00 5D 00 00	.00141 90673	.00 9D 00 00	.00239 56298	.00 DD 00 00	.00337 21923
.00 1E 00 00	.00045 77636	.00 5E 00 00	.00143 43261	.00 9E 00 00	.00241 08886	.00 DE 00 00	.00338 74511
.00 1F 00 00	.00047 30224	.00 5F 00 00	.00144 95849	.00 9F 00 00	.00242 61474	.00 DF 00 00	.00340 27099
.00 20 00 00	.00048 82812	.00 60 00 00	.00146 48437	.00 A0 00 00	.00244 14062	.00 E0 00 00	.00341 79687
.00 21 00 00	.00050 35400	.00 61 00 00	.00148 01025	.00 A1 00 00	.00245 66650	.00 E1 00 00	.00343 32275
.00 22 00 00	.00051 87988	.00 62 00 00	.00149 53613	.00 A2 00 00	.00247 19238	.00 E2 00 00	.00344 84863
.00 23 00 00	.00053 40576	.00 63 00 00	.00151 06201	.00 A3 00 00	.00248 71826	.00 E3 00 00	.00346 37451
.00 24 00 00	.00054 93164	.00 64 00 00	.00152 58789	.00 A4 00 00	.00250 24414	.00 E4 00 00	.00347 90039
.00 25 00 00	.00056 45751	.00 65 00 00	.00154 11376	.00 A5 00 00	.00251 77001	.00 E5 00 00	.00349 42626
.00 26 00 00	.00057 98339	.00 66 00 00	.00155 63964	.00 A6 00 C0	.00253 29589	.00 E6 00 00	.00350 95214
.00 27 00 00	.00059 50927	.00 67 00 00	.00157 16552	.00 A7 00 00	.00254 82177	.00 E7 00 00	.00352 47802
.00 28 00 00	.00061 03515	.00 68 00 00	.00158 69140	.00 A8 00 00	.00256 34765	.00 E8 00 00	.00354 00390
.00 29 00 00	.00062 56103	.00 69 00 00	.00160 21728	.00 A9 00 00	.00257 87353	.00 E9 00 00	.00355 52978
.00 2A 00 00	.00064 08691	.00 6A 00 00	.00161 74316	.00 AA 00 00	.00259 39941	.00 EA 00 00	.00357 05566
.00 2B 00 00	.00065 61279	.00 6B 00 00	.00163 26904	.00 AB 00 00	.00260 92529	.00 EB 00 00	.00358 58154
.00 2C 00 00	.00067 13867	.00 6C 00 00	.00164 79492	.00 AC 00 00	.00262 45117	.00 EC 00 00	.00360 10742
.00 2D 00 00	.00068 66455	.00 6D 00 00	.00166 32080	.00 AD 00 00	.00263 97705	.00 ED 00 00	.00361 63330
.00 2E 00 00	.00070 19042	.00 6E 00 00	.00167 84667	.00 AE 00 00	.00265 50292	.00 EE 00 00	.00363 15917
.00 2F 00 00	.00071 71630	.00 6F 00 00	.00169 37255	.00 AF 00 00	.00267 02880	.00 EF 00 00	.00364 68505
.00 30 00 00	.00073 24218	.00 70 00 00	.00170 89843	.00 B0 00 00	.00268 55468	.00 F0 00 00	.00366 21093
.00 31 00 00	.00074 76806	.00 71 00 00	.00172 42431	.00 B1 00 00	.00270 08056	.00 F1 00 00	.00367 73681
.00 32 00 00	.00076 29394	.00 72. 00 00	.00173 95019	.00 B2 00 00	.00271 60644	.00 F2 00 00	.00369 26269
.00 33 00 00	.00077 81982	.00 73 00 00	.00175 47607	.00 B3 00 00	.00273 13232	.00 F3 00 00	.00370 78857
.00 34 00 00	.00079 34570	.00 74 00 00	.00177 00195	.00 B4 00 00	.00274 65820	.00 F4 00 00	.00372 31445
.00 35 00 00	.00080 87158	.00 75 00 00	.00178 52783	.00 B5 00 00	.00276 18408	.00 F5 00 00	.00373 84033
.00 36 00 00	.00082 39746	.00 76 00 00	.00180 05371	.00 B6 00 00	.00277 70996	.00 F6 00 00	.00375 36621
.00 37 00 00	.00083 92333	.00 77 00 00	.00181 57958	.00 B7 00 00	.00279 23583	.00 F7 00 00	.00376 89208
.00 38 00 00	.00085 44921	.00 78 00 00	.00183 10546	.00 B8 00 00	.00280 76171	.00 F8 00 00	.00378 41796
.00 39 00 00	.00086 97509	.00 79 00 00	.00184 63134	.00 B9 00 00	.00282 28759	.00 F9 00 00	.00379 94384
.00 3A 00 00	.00088 50097	.00 7A 00 00	.00186 15722	.00 BA 00 00	.00283 81347	.00 FA 00 00	.00381 46972
.00 3B 00 00	.00090 02685	.00 7B 00 00	.00187 68310	.00 BB 00 00	.00285 33935	.00 FB 00 00	.00382 99560
.00 3C 00 00	.00091 55273	.00 7C 00 00	.00189 20898	.00 BC 00 00	.00286 86523	.00 FC 00 00	.00384 52148
.00 3D 00 00	.00093 07861	.00 7D 00 00	.00190 73486	.00 BD 00 00	.00288 39111	.00 FD 00 00	.00386 04736
.00 3E 00 00	.00094 60449	.00 7E 00 00	.00192 26074	.00 BE 00 00	.00289 91699	.00 FE 00 00	.00387 57324
.00 3F 00 00	.00096 13037	.00 7F 00 00	.00193 78662	.00 BF 00 00	.00291 44287	.00 FF 00 00	.00389 09912

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 40 00	.00000 38146	.00 00 80 00	.00000 76293	.00 00 C0 00	.00001 14440
.00 00 01 00	.00000 00596	.00 00 41 00	.00000 38743	.00 00 81 00	.00000 76889	.00 00 C1 00	.00001 15036
.00 00 02 00	.00000 01192	.00 00 42 00	.00000 39339	.00 00 82 00	.00000 77486	.00 00 C2 00	.00001 15633
.00 00 03 00	.00000 01788	.00 00 43 00	.00000 39935	.00 00 83 00	.00000 78082	.00 00 C3 00	.00001 16229
.00 00 04 00	.00000 02384	.00 00 44 00	.00000 40531	.00 00 84 00	.00000 78678	.00 00 C4 00	.00001 16825
.00 00 05 00	.00000 02980	.00 00 45 00	.00000 41127	.00 00 85 00	.00000 79274	.00 00 C5 00	.00001 17421
.00 00 06 00	.00000 03576	.00 00 46 00	.00000 41723	.00 00 86 00	.00000 79870	.00 00 C6 00	.00001 18017
.00 00 07 00	.00000 04172	.00 00 47 00	.00000 42319	.00 00 87 00	.00000 80466	.00 00 C7 00	.00001 18613
.00 00 08 00	.00000 04768	.00 00 48 00	.00000 42915	.00 00 88 00	.00000 81062	.00 00 C8 00	.00001 19209
.00 00 09 00	.00000 05364	.00 00 49 00	.00000 43511	.00 00 89 00	.00000 81658	.00 00 C9 00	.00001 19805
.00 00 0A 00	.00000 05960	.00 00 4A 00	.00000 44107	.00 00 8A 00	.00000 82254	.00 00 CA 00	.00001 20401
.00 00 0B 00	.00000 06556	.00 00 4B 00	.00000 44703	.00 00 8B 00	.00000 82850	.00 00 CB 00	.00001 20997
.00 00 0C 00	.00000 07152	.00 00 4C 00	.00000 45299	.00 00 8C 00	.00000 83446	.00 00 CC 00	.00001 21593
.00 00 0D 00	.00000 07748	.00 00 4D 00	.00000 45895	.00 00 8D 00	.00000 84042	.00 00 CD 00	.00001 22189
.00 00 0E 00	.00000 08344	.00 00 4E 00	.00000 46491	.00 00 8E 00	.00000 84638	.00 00 CE 00	.00001 22785
.00 00 0F 00	.00000 08940	.00 00 4F 00	.00000 47087	.00 00 8F 00	.00000 85234	.00 00 CF 00	.00001 23381
.00 00 10 00	.00000 09536	.00 00 50 00	.00000 47683	.00 00 90 00	.00000 85830	.00 00 D0 00	.00001 23977
.00 00 11 00	.00000 10132	.00 00 51 00	.00000 48279	.00 00 91 00	.00000 86426	.00 00 D1 00	.00001 24573
.00 00 12 00	.00000 10728	.00 00 52 00	.00000 48875	.00 00 92 00	.00000 87022	.00 00 D2 00	.00001 25169
.00 00 13 00	.00000 11324	.00 00 53 00	.00000 49471	.00 00 93 00	.00000 87618	.00 00 D3 00	.00001 25765
.00 00 14 00	.00000 11920	.00 00 54 00	.00000 50067	.00 00 94 00	.00000 88214	.00 00 D4 00	.00001 26361
.00 00 15 00	.00000 12516	.00 00 55 00	.00000 50663	.00 00 95 00	.00000 88810	.00 00 D5 00	.00001 26957
.00 00 16 00	.00000 13113	.00 00 56 00	.00000 51259	.00 00 96 00	.00000 89406	.00 00 D6 00	.00001 27553
.00 00 17 00	.00000 13709	.00 00 57 00	.00000 51856	.00 00 97 00	.00000 90003	.00 00 D7 00	.00001 28149
.00 00 18 00	.00000 14305	.00 00 58 00	.00000 52452	.00 00 98 00	.00000 90599	.00 00 D8 00	.00001 28746
.00 00 19 00	.00000 14901	.00 00 59 00	.00000 53048	.00 00 99 00	.00000 91195	.00 00 D9 00	.00001 29342
.00 00 1A 00	.00000 15497	.00 00 5A 00	.00000 53644	.00 00 9A 00	.00000 91791	.00 00 DA 00	.00001 29938
.00 00 1B 00	.00000 16093	.00 00 5B 00	.00000 54240	.00 00 9B 00	.00000 92387	.00 00 DB 00	.00001 30534
.00 00 1C 00	.00000 16689	.00 00 5C 00	.00000 54836	.00 00 9C 00	.00000 92983	.00 00 DC 00	.00001 31130
.00 00 1D 00	.00000 17285	.00 00 5D 00	.00000 55432	.00 00 9D 00	.00000 93579	.00 00 DD 00	.00001 31726
.00 00 1E 00	.00000 17881	.00 00 5E 00	.00000 56028	.00 00 9E 00	.00000 94175	.00 00 DE 00	.00001 32322
.00 00 1F 00	.00000 18477	.00 00 5F 00	.00000 56624	.00 00 9F 00	.00000 94771	.00 00 DF 00	.00001 32918
.00 00 20 00	.00000 19073	.00 00 60 00	.00000 57220	.00 00 A0 00	.00000 95367	.00 00 E0 00	.00001 33514
.00 00 21 00	.00000 19669	.00 00 61 00	.00000 57816	.00 00 A1 00	.00000 95963	.00 00 E1 00	.00001 34110
.00 00 22 00	.00000 20265	.00 00 62 00	.00000 58412	.00 00 A2 00	.00000 96559	.00 00 E2 00	.00001 34706
.00 00 23 00	.00000 20861	.00 00 63 00	.00000 59008	.00 00 A3 00	.00000 97155	.00 00 E3 00	.00001 35302
.00 00 24 00	.00000 21457	.00 00 64 00	.00000 59604	.00 00 A4 00	.00000 97751	.00 00 E4 00	.00001 35898
.00 00 25 00	.00000 22053	.00 00 65 00	.00000 60200	.00 00 A5 00	.00000 98347	.00 00 E5 00	.00001 36494
.00 00 26 00	.00000 22649	.00 00 66 00	.00000 60796	.00 00 A6 00	.00000 98943	.00 00 E6 00	.00001 37090
.00 00 27 00	.00000 23245	.00 00 67 00	.00000 61392	.00 00 A7 00	.00000 99539	.00 00 E7 00	.00001 37686
.00 00 28 00	.00000 23841	.00 00 68 00	.00000 61988	.00 00 A8 00	.00001 00135	.00 00 E8 00	.00001 38282
.00 00 29 00	.00000 24437	.00 00 69 00	.00000 62584	.00 00 A9 00	.00001 00731	.00 00 E9 00	.00001 38878
.00 00 2A 00	.00000 25033	.00 00 6A 00	.00000 63180	.00 00 AA 00	.00001 01327	.00 00 EA 00	.00001 39474
.00 00 2B 00	.00000 25629	.00 00 6B 00	.00000 63776	.00 00 AB 00	.00001 01923	.00 00 EB 00	.00001 40070
.00 00 2C 00	.00000 26226	.00 00 6C 00	.00000 64373	.00 00 AC 00	.00001 02519	.00 00 EC 00	.00001 40666
.00 00 2D 00	.00000 26822	.00 00 6D 00	.00000 64969	.00 00 AD 00	.00001 03116	.00 00 ED 00	.00001 41263
.00 00 2E 00	.00000 27418	.00 00 6E 00	.00000 65565	.00 00 AE 00	.00001 03712	.00 00 EE 00	.00001 41859
.00 00 2F 00	.00000 28014	.00 00 6F 00	.00000 66161	.00 00 AF 00	.00001 04308	.00 00 EF 00	.00001 42455
.00 00 30 00	.00000 28610	.00 00 70 00	.00000 66757	.00 00 B0 00	.00001 04904	.00 00 F0 00	.00001 43051
.00 00 31 00	.00000 29206	.00 00 71 00	.00000 67353	.00 00 B1 00	.00001 05500	.00 00 F1 00	.00001 43647
.00 00 32 00	.00000 29802	.00 00 72 00	.00000 67949	.00 00 B2 00	.00001 06096	.00 00 F2 00	.00001 44243
.00 00 33 00	.00000 30398	.00 00 73 00	.00000 68545	.00 00 B3 00	.00001 06692	.00 00 F3 00	.00001 44839
.00 00 34 00	.00000 30994	.00 00 74 00	.00000 69141	.00 00 B4 00	.00001 07288	.00 00 F4 00	.00001 45435
.00 00 35 00	.00000 31590	.00 00 75 00	.00000 69737	.00 00 B5 00	.00001 07884	.00 00 F5 00	.00001 46031
.00 00 36 00	.00000 32186	.00 00 76 00	.00000 70333	.00 00 B6 00	.00001 08480	.00 00 F6 00	.00001 46627
.00 00 37 00	.00000 32782	.00 00 77 00	.00000 70929	.00 00 B7 00	.00001 09076	.00 00 F7 00	.00001 47223
.00 00 38 00	.00000 33378	.00 00 78 00	.00000 71525	.00 00 B8 00	.00001 09672	.00 00 F8 00	.00001 47819
.00 00 39 00	.00000 33974	.00 00 79 00	.00000 72121	.00 00 B9 00	.00001 10268	.00 00 F9 00	.00001 48415
.00 00 3A 00	.00000 34570	.00 00 7A 00	.00000 72717	.00 00 BA 00	.00001 10864	.00 00 FA 00	.00001 49011
.00 00 3B 00	.00000 35166	.00 00 7B 00	.00000 73313	.00 00 BB 00	.00001 11460	.00 00 FB 00	.00001 49607
.00 00 3C 00	.00000 35762	.00 00 7C 00	.00000 73909	.00 00 BC 00	.00001 12056	.00 00 FC 00	.00001 50203
.00 00 3D 00	.00000 36358	.00 00 7D 00	.00000 74505	.00 00 BD 00	.00001 12652	.00 00 FD 00	.00001 50799
.00 00 3E 00	.00000 36954	.00 00 7E 00	.00000 75101	.00 00 BE 00	.00001 13248	.00 00 FE 00	.00001 51395
.00 00 3F 00	.00000 37550	.00 00 7F 00	.00000 75697	.00 00 BF 00	.00001 13844	.00 00 FF 00	.00001 51991

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 00 40	.00000 00149	.00 00 00 80	.00000 00298	.00 00 00 C0	.00000 00447
.00 00 00 01	.00000 00002	.00 00 00 41	.00000 00151	.00 00 00 81	.00000 00300	.00 00 00 C1	.00000 00449
.00 00 00 02	.00000 00004	.00 00 00 42	.00000 00153	.00 00 00 82	.00000 00302	.00 00 00 C2	.00000 00451
.00 00 00 03	.00000 00006	.00 00 00 43	.00000 00155	.00 00 00 83	.00000 00305	.00 00 00 C3	.00000 00454
.00 00 00 04	.00000 00009	.00 00 00 44	.00000 00158	.00 00 00 84	.00000 00307	.00 00 00 C4	.00000 00456
.00 00 00 05	.00000 00011	.00 00 00 45	.00000 00160	.00 00 00 85	.00000 00309	.00 00 00 C5	.00000 00458
.00 00 00 06	.00000 00013	.00 00 00 46	.00000 00162	.00 00 00 86	.00000 00311	.00 00 00 C6	.00000 00461
.00 00 00 07	.00000 00016	.00 00 00 47	.00000 00165	.00 00 00 87	.00000 00314	.00 00 00 C7	.00000 00463
.00 00 00 08	.00000 00018	.00 00 00 48	.00000 00167	.00 00 00 88	.00000 00316	.00 00 00 C8	.00000 00465
.00 00 00 09	.00000 00020	.00 00 00 49	.00000 00169	.00 00 00 89	.00000 00318	.00 00 00 C9	.00000 00467
.00 00 00 0A	.00000 00023	.00 00 00 4A	.00000 00172	.00 00 00 8A	.00000 00321	.00 00 00 CA	.00000 00470
.00 00 00 0B	.00000 00025	.00 00 00 4B	.00000 00174	.00 00 00 8B	.00000 00323	.00 00 00 CB	.00000 00472
.00 00 00 0C	.00000 00027	.00 00 00 4C	.00000 00176	.00 00 00 8C	.00000 00325	.00 00 00 CC	.00000 00474
.00 00 00 0D	.00000 00030	.00 00 00 4D	.00000 00179	.00 00 00 8D	.00000 00328	.00 00 00 CD	.00000 00477
.00 00 00 0E	.00000 00032	.00 00 00 4E	.00000 00181	.00 00 00 8E	.00000 00330	.00 00 00 CE	.00000 00479
.00 00 00 0F	.00000 00034	.00 00 00 4F	.00000 00183	.00 00 00 8F	.00000 00332	.00 00 00 CF	.00000 00481
.00 00 00 10	.00000 00037	.00 00 00 50	.00000 00186	.00 00 00 90	.00000 00335	.00 00 00 D0	.00000 00484
.00 00 00 11	.00000 00039	.00 00 00 51	.00000 00188	.00 00 00 91	.00000 00337	.00 00 00 D1	.00000 00486
.00 00 00 12	.00000 00041	.00 00 00 52	.00000 00190	.00 00 00 92	.00000 00339	.00 00 00 D2	.00000 00488
.00 00 00 13	.00000 00044	.00 00 00 53	.00000 00193	.00 00 00 93	.00000 00342	.00 00 00 D3	.00000 00491
.00 00 00 14	.00000 00046	.00 00 00 54	.00000 00195	.00 00 00 94	.00000 00344	.00 00 00 D4	.00000 00493
.00 00 00 15	.00000 00048	.00 00 00 55	.00000 00197	.00 00 00 95	.00000 00346	.00 00 00 D5	.00000 00495
.00 00 00 16	.00000 00051	.00 00 00 56	.00000 00200	.00 00 00 96	.00000 00349	.00 00 00 D6	.00000 00498
.00 00 00 17	.00000 00053	.00 00 00 57	.00000 00202	.00 00 00 97	.00000 00351	.00 00 00 D7	.00000 00500
.00 00 00 18	.00000 00055	.00 00 00 58	.00000 00204	.00 00 00 98	.00000 00353	.00 00 00 D8	.00000 00502
.00 00 00 19	.00000 00058	.00 00 00 59	.00000 00207	.00 00 00 99	.00000 00356	.00 00 00 D9	.00000 00505
.00 00 00 1A	.00000 00060	.00 00 00 5A	.00000 00209	.00 00 00 9A	.00000 00358	.00 00 00 DA	.00000 00507
.00 00 00 1B	.00000 00062	.00 00 00 5B	.00000 00211	.00 00 00 9B	.00000 00360	.00 00 00 DB	.00000 00509
.00 00 00 1C	.00000 00065	.00 00 00 5C	.00000 00214	.00 00 00 9C	.00000 00363	.00 00 00 DC	.00000 00512
.00 00 00 1D	.00000 00067	.00 00 00 5D	.00000 00216	.00 00 00 9D	.00000 00365	.00 00 00 DD	.00000 00514
.00 00 00 1E	.00000 00069	.00 00 00 5E	.00000 00218	.00 00 00 9E	.00000 00367	.00 00 00 DE	.00000 00516
.00 00 00 1F	.00000 00072	.00 00 00 5F	.00000 00221	.00 00 00 9F	.00000 00370	.00 00 00 DF	.00000 00519
.00 00 00 20	.00000 00074	.00 00 00 60	.00000 00223	.00 00 00 A0	.00000 00372	.00 00 00 E0	.00000 00521
.00 00 00 21	.00000 00076	.00 00 00 61	.00000 00225	.00 00 00 A1	.00000 00374	.00 00 00 E1	.00000 00523
.00 00 00 22	.00000 00079	.00 00 00 62	.00000 00228	.00 00 00 A2	.00000 00377	.00 00 00 E2	.00000 00526
.00 00 00 23	.00000 00081	.00 00 00 63	.00000 00230	.00 00 00 A3	.00000 00379	.00 00 00 E3	.00000 00528
.00 00 00 24	.00000 00083	.00 00 00 64	.00000 00232	.00 00 00 A4	.00000 00381	.00 00 00 E4	.00000 00530
.00 00 00 25	.00000 00086	.00 00 00 65	.00000 00235	.00 00 00 A5	.00000 00384	.00 00 00 E5	.00000 00533
.00 00 00 26	.00000 00088	.00 00 00 66	.00000 00237	.00 00 00 A6	.00000 00386	.00 00 00 E6	.00000 00535
.00 00 00 27	.00000 00090	.00 00 00 67	.00000 00239	.00 00 00 A7	.00000 00388	.00 00 00 E7	.00000 00537
.00 00 00 28	.00000 00093	.00 00 00 68	.00000 00242	.00 00 00 A8	.00000 00391	.00 00 00 E8	.00000 00540
.00 00 00 29	.00000 00095	.00 00 00 69	.00000 00244	.00 00 00 A9	.00000 00393	.00 00 00 E9	.00000 00542
.00 00 00 2A	.00000 00097	.00 00 00 6A	.00000 00246	.00 00 00 AA	.00000 00395	.00 00 00 EA	.00000 00544
.00 00 00 2B	.00000 00100	.00 00 00 6B	.00000 00249	.00 00 00 AB	.00000 00398	.00 00 00 EB	.00000 00547
.00 00 00 2C	.00000 00102	.00 00 00 6C	.00000 00251	.00 00 00 AC	.00000 00400	.00 00 00 EC	.00000 00549
.00 00 00 2D	.00000 00104	.00 00 00 6D	.00000 00253	.00 00 00 AD	.00000 00402	.00 00 00 ED	.00000 00551
.00 00 00 2E	.00000 00107	.00 00 00 6E	.00000 00256	.00 00 00 AE	.00000 00405	.00 00 00 EE	.00000 00554
.00 00 00 2F	.00000 00109	.00 00 00 6F	.00000 00258	.00 00 00 AF	.00000 00407	.00 00 00 EF	.00000 00556
.00 00 00 30	.00000 00111	.00 00 00 70	.00000 00260	.00 00 00 B0	.00000 00409	.00 00 00 F0	.00000 00558
.00 00 00 31	.00000 00114	.00 00 00 71	.00000 00263	.00 00 00 B1	.00000 00412	.00 00 00 F1	.00000 00561
.00 00 00 32	.00000 00116	.00 00 00 72	.00000 00265	.00 00 00 B2	.00000 00414	.00 00 00 F2	.00000 00563
.00 00 00 33	.00000 00118	.00 00 00 73	.00000 00267	.00 00 00 B3	.00000 00416	.00 00 00 F3	.00000 00565
.00 00 00 34	.00000 00121	.00 00 00 74	.00000 00270	.00 00 00 B4	.00000 00419	.00 00 00 F4	.00000 00568
.00 00 00 35	.00000 00123	.00 00 00 75	.00000 00272	.00 00 00 B5	.00000 00421	.00 00 00 F5	.00000 00570
.00 00 00 36	.00000 00125	.00 00 00 76	.00000 00274	.00 00 00 B6	.00000 00423	.00 00 00 F6	.00000 00572
.00 00 00 37	.00000 00128	.00 00 00 77	.00000 00277	.00 00 00 B7	.00000 00426	.00 00 00 F7	.00000 00575
.00 00 00 38	.00000 00130	.00 00 00 78	.00000 00279	.00 00 00 B8	.00000 00428	.00 00 00 F8	.00000 00577
.00 00 00 39	.00000 00132	.00 00 00 79	.00000 00281	.00 00 00 B9	.00000 00430	.00 00 00 F9	.00000 00579
.00 00 00 3A	.00000 00135	.00 00 00 7A	.00000 00284	.00 00 00 BA	.00000 00433	.00 00 00 FA	.00000 00582
.00 00 00 3B	.00000 00137	.00 00 00 7B	.00000 00286	.00 00 00 BB	.00000 00435	.00 00 00 FB	.00000 00584
.00 00 00 3C	.00000 00139	.00 00 00 7C	.00000 00288	.00 00 00 BC	.00000 00437	.00 00 00 FC	.00000 00586
.00 00 00 3D	.00000 00142	.00 00 00 7D	.00000 00291	.00 00 00 BD	.00000 00440	.00 00 00 FD	00000 00589
.00 00 00 3E	.00000 00144	.00 00 00 7E	.00000 00293	.00 00 00 BE	.00000 00442	.00 00 00 FE	.00000 00591
.00 00 00 3F	.00000 00146	.00 00 00 7F	.00000 00295	.00 00 00 BF	.00000 00444	.00 00 00 FF	.00000 00593

APPENDIX G
MATHEMATICAL CONSTANTS

Constant	Decimal Value	Hexadecimal Value
π	3.14159 26535 89793	3.243F 6A89
π^{-1}	0.31830 98861 83790	0.517C C1B7
$\sqrt{\pi}$	1.77245 38509 05516	1.C5BF 891C
$\ln \pi$	1.14472 98858 49400	1.250D 048F
e	2.71828 18284 59045	2.B7E1 5163
e^{-1}	0.36787 94411 71442	0.5E2D 58D9
$\sqrt{e}$	1.64872 12707 00128	1.A612 98E2
$\log_{10} e$	0.43429 44819 03252	0.6F2D EC55
$\log_2 e$	1.44269 50408 88963	1.7154 7653
γ	0.57721 56649 01533	0.93C4 67E4
$\ln \gamma$	-0.54953 93129 81645	-0.8CAE 9BC1
$\sqrt{2}$	1.41421 35623 73095	1.6A09 E668
ln2	0.69314 71805 59945	0.B172 17F8
$\log_{10} 2$	0.30102 99956 63981	0.4D10 4D42
$\sqrt{10}$	3.16227 76601 68379	3.298B 075C
ln10	2.30258 50929 94046	2.4D76 3777

APPENDIX H

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS

LOAD AND STORE INSTRUCTIONS

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
LAS	0640	Load Accumulator From Switches	(Switches) → A	E	1.00	ST
LDA	Cxxx	Load Accumulator	(EA) → A	B	1.84	ST
LDB	2xxx	Load B Register	(EA) → B	B	1.84	ST
LDD	0x80	Load Double	(EA) → A,(EA+1) → B	D	3.68	DP
LDF	0x88	Load Floating	(EA) → A,(EA+1) → B,(EA+2) → E	D	4.60	DFP
LDH	1xxx	Load Halfword	$0 \rightarrow A_{0-7}$; (EA) → A_{8-15}	B	2.59 or 3.01***	ST
LDL	6xxx	Load A, Literal	$(INS)_7 \rightarrow A_{0-7}$, $(INS)_{8-15} \rightarrow A_{8-15}$	L	1.05	ST
LDLB	6xxx	Load B, Literal	$(INS)_7 \rightarrow B_{0-7}$, $(INS)_{8-15} \rightarrow B_{8-15}$	L	1.05	ST
LDS	0648	Load Status	(ST) → A	E	1.00	ST
LDX	0x78	Load Index	(EA) → X	D	2.81	ST
LSMP	01E1	Load System Map Pointer	α_s → SMTP, 0 → Assoc. Reg.	E	1.95	ST
LUMP	01E2	Load User Map Pointer	α_u → UMTP, 0 → Assoc. Reg.	E	1.95	ST
STA	7xxx	Store Accumulator	(A) → EA	B	1.84	ST
STB	3xxx	Store B Register	(B) → EA	B	1.84	ST
STD	0x90	Store Double	(A) → EA, (B) → EA + 1	D	3.68	DP
STF	0x98	Store Floating	(A) → EA, (B) → EA + 1, (E) → EA + 2	D	4.60	DFP
STH	5xxx	Store Halfword	$(A)_{8-15}$ → EA	B	2.89 or 3.31***	ST
STX	0x38	Store Index	(X) → EA	D	2.81	ST

ARITHMETIC INSTRUCTIONS

ADC	0608	Add Carry	(A) + (CRO) → A	E	1.25	ST
ADD	9xxx	Add to Accumulator	(A) + (EA) → A	B	1.84	ST
ADL	6xxx	Add to A, Literal	$(INS)_7 \rightarrow INS_{0-6}$; (INS)+(A) → A	L	1.15	ST
ADLB	6xxx	Add to B, Literal	$(INS)_7 \rightarrow INS_{0-6}$; (INS)+(B) → B	L	1.15	ST
DAD	0xA0	Double Add	(A,B) + (EA,EA+1) → A, B	D	3.96	DP
DDV	0xB8	Double Divide	(A,B) ÷ (EA,EA+1) → A, B	D	32.81 or 44.81	DP

*** Indirect Address Time.

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS (CONT'D)

ARITHMETIC INSTRUCTIONS (CONT'D)

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
DFA	0xC8	Double Floating Add	(A,B,E) + (EA,EA+1,EA+2) → A,B,E + denotes floating add	D	7.88 to 9.03**	DFP
DFD	0xF8	Double Floating Divide	(A,B,E) ÷ (EA,EA+1,EA+2) → A,B,E ÷ denotes floating divide	D	34.68 to 39.58**	DFP
DFM	0xE8	Double Floating Multiply	(A,B,E) X (EA,EA+1,EA+2) → A,B,E X denotes floating multiply	D	26.28 to 27.63**	DFP
DFS	0xD8	Double Floating Subtract	(A,B,E) - (EA,EA+1,EA+2) → A,B,E - denotes floating subtract	D	7.88 to 9.03**	DFP
DIV	0x08	Divide	(B, A) / (EA) → A; remainder → B	D	8.14 to 8.99	MD
DMP	0xB0	Double Multiply	(A,B) X (EA,EA+1) → A,B	D	25.30	DP
DSB	0xA8	Double Subtract	(A,B) - (EA,EA+1) → A,B	D	3.91	DP
FAD	0xC0	Floating Point Add	(A,B) + (EA,EA+1) → A,B + denotes floating add	D	9.11 to 12.01‡¤	FP
FDV	0xF0	Floating Point Divide	(A,B) ÷ (EA,EA+1) → A,B ÷ denotes floating divide	D	37.16 to 44.41	FP
FMP	0xE0	Floating Point Multiply	(A,B) X (EA,EA+1) → A,B X denotes floating multiply	D	31.99 to 33.49	FP
FSB	0xD0	Floating Point Subtract	(A,B) - (EA, EA+1) → A,B - denotes floating subtract	D	9.81 to 13.01‡¤	FP

MIN	8xxx	Memory Increment Skip on Zero	(EA) + 1 → EA
			If (EA) = 0, (L)+2 → L or if PA = L+1,(L)+3 → L
			If (EA) ≠ 0, (L) +2 → L or if PA = L+1,(L)+2 → L
				B	2.14	ST

MPX	0728	Multiply Index	(X) * 3 → X	E	1.20	ST
MPY	0x00	Multiply	(A) X (EA) → B, A	D	8.44	MD
SUB	Dxxx	Subtract from Accumulator	(A) − (EA) → A	B	1.84	ST

LOGICAL INSTRUCTIONS

AAB	0600	AND Accumulator with B Register	(A) ∩ (B) → A	E	1.15	ST
AND	Fxxx	AND with Accumulator	(A) ∩ (EA) → A	B	1.84	ST

‡ +.25N for Alignment.

¤ +.55N for Normalize.

** +.21N for Alignment.

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS (CONT'D)

LOGICAL INSTRUCTIONS (CONT'D)

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
ANL	6xxx	AND the Accumulator, Literal	$(INS)_7 \rightarrow INS_{0-6}$;(INS) $\cap$ (A) $\rightarrow$ A	L	1.15	ST
ANLB	6xxx	AND the B Register, Literal	$(INS)_7 \rightarrow INS_{0-6}$;(INS) $\cap$ (B) $\rightarrow$ B	L	1.15	ST
AOB	0610	OR Accumulator with B Register	(A) $\cup$ (B) $\rightarrow$ A	E	1.15	ST
XOL	6xxx	Exclusive OR the Accumulator, Literal	$(INS)_7 \rightarrow INS_{0-6}$;(INS) $\oplus$ (A) $\rightarrow$ A	L	1.15	ST
XOLB	6xxx	Exclusive OR the B Register, Literal	$(INS)_7 \rightarrow INS_{0-6}$;(INS) $\oplus$ (B) $\rightarrow$ B	L	1.15	ST
XOR	Bxxx	Exclusive OR with Accumulator	(A) $\oplus$ (EA) $\rightarrow$ A	B	1.84	ST

REGISTER MANIPULATION

ESA	0620	Extend Sign of Accumulator	$(A)_0 \rightarrow B_{0-15}$	E	1.05	ST
XAB	0720	Exchange Accumulator and B Register	(A) $\rightarrow$ B, (B) $\rightarrow$ A	E	1.15	ST
XAX	0618	Exchange Accumulator and Index	(A) $\rightarrow$ X, (X) $\rightarrow$ A	E	1.15	ST
XBX	0670	Exchange B Register and Index	(B) $\rightarrow$ X, (X) $\rightarrow$ B	E	1.15	ST
XHA	0308	Exchange Halves of Accumulator	$(A)_{0-7} \rightarrow A_{8-15}$, $(A)_{8-15} \rightarrow A_{0-7}$	E	3.25	ST

OPERATE GROUP

NOP	0400	No Operation	(A) $\rightarrow$ A	E	1.25	ST
RADD	04xx	Register Add	S + D $\rightarrow$ D	E	1.45	ST
RCMP	05xx	Register Complement	$\overline{S} \rightarrow$ D	E	1.25	ST
RCPY	04xx	Register Copy	S $\rightarrow$ D	E	1.25	ST
RDEC	05xx	Register Decrement	S − 1 $\rightarrow$ D	E	1.25	ST
RINC	04xx	Register Increment	S + 1 $\rightarrow$ D	E	1.25	ST
RNEG	05xx	Register Negate	$\overline{S}$ + 1 $\rightarrow$ D	E	1.25	ST
RSUB	05xx	Register Subtract	S − D $\rightarrow$ D	E	1.45	ST
RXOR	04xx	Register Exclusive OR	S $\oplus$ D $\rightarrow$ D	E	1.45	ST

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS (CONT'D)

SHIFT INSTRUCTIONS

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
LAL	0xxx	Long Arithmetic, Left Shift	$(A)_i \rightarrow A_{i-1}, (B)_i \rightarrow B_{i-1}$, $(B)_0 \rightarrow A_{15}, 0 \rightarrow B_{15}$	E	1.25 + .25N	ST
LAR	0xxx	Long Arithmetic, Right Shift	$(A)_0 \rightarrow A_0, (A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}, (A)_{15} \rightarrow B_0$	E	1.25 + .25N	ST
LCL	0xxx	Long Circulate, Left	$(A)_i \rightarrow A_{i-1}, (B)_i \rightarrow B_{i-1}$, $(CRO) \rightarrow B_{15}, (B)_0 \rightarrow A_{15}$ $(A)_0 \rightarrow CRO$	E	1.25 + .25N	ST
LCR	0xxx	Long Circulate, Right	$(CRO) \rightarrow A_0, (A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}, (A)_{15} \rightarrow B_0$, $(B)_0 \rightarrow CRO$	E	1.25 + .25N	ST
LLL	0xxx	Long Logical, Left Shift	$(A)_i \rightarrow A_{i-1}, (B)_i \rightarrow B_{i-1}$, $(B)_0 \rightarrow A_{15}, 0 \rightarrow B_{15}$	E	1.25 + .25N	ST
LLO	0628	Locate Leading Ones	If (A) = 0, (L)+1 → L. If (A) ≠ 0, $(A)_i \rightarrow A_{i-1}$ and (X)+1 → X until $A_0 = 1$. When $A_0 = 1$, (L)+2 → L and $0 \rightarrow A_0$.		If A = 0, 1.25 If A ≠ 0, 1.85 + .20N	ST
LLR	0xxx	Long Logical, Right Shift	$0 \rightarrow A_0, (A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}, (A)_{15} \rightarrow (B)_0$	E	1.25 + .25N	ST
LRL	0xxx	Long Rotate Left	$(A)_i \rightarrow A_{i-1}, (B)_i \rightarrow B_{i-1}$, $(B)_0 \rightarrow A_{15}, (A)_0 \rightarrow B_{15}$	E	1.25 + .25N	ST
LRR	0xxx	Long Rotate Right	$(B)_{15} \rightarrow A_0, (A)_i \rightarrow A_{i+1}$, $(B)_i \rightarrow B_{i+1}, (A)_{15} \rightarrow B_0$	E	1.25 + .25N	ST
NDX	0710	Normalize and Decrement X	If (A) and (B) = 0, 0 → X. If (A) or (B) ≠ 0, $(A)_i \rightarrow A_{i-1}$, $(B)_i \rightarrow B_{i-1}, (B)_0 \rightarrow A_{15}$, and (X)-1 → X until $(A)_0 \neq (A)_1$	E	If (A) and (B) = 0, 1.45 If (A) or (B) ≠ 0, 1.65 + .4N	ST
SAL	0xxx	Short Arithmetic Left Shift	$0 \rightarrow A_{15}, (A)_i \rightarrow A_{i-1}$	E	1.25 + .25N	ST
SAR	0xxx	Short Arithmetic Right Shift	$(A)_0 \rightarrow A_0, (A)_i \rightarrow A_{i+1}$	E	1.25 + .25N	ST
SCL	0xxx	Short Circulate, Left	$(A)_i \rightarrow A_{i-1}, (A)_0 \rightarrow CRO$, $(CRO) \rightarrow A_{15}$	E	1.25 + .25N	ST
SCR	0xxx	Short Circulate, Right	$(CRO) \rightarrow A_0, (A)_i \rightarrow A_{i+1}$, $(A)_{15} \rightarrow CRO$	E	1.25 + .25N	ST
SLL	0xxx	Short Logical, Left Shift	$(A)_i \rightarrow A_{i-1}, 0 \rightarrow A_{15}$	E	1.25 + .25N	ST
SLR	0xxx	Short Logical, Right Shift	$0 \rightarrow A_0, (A)_i \rightarrow A_{i+1}$	E	1.25 + .25N	ST

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS (CONT'D)

SHIFT INSTRUCTIONS (CONT'D)

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
SRL	0xxx	Short Rotate Left	$(A)_i \rightarrow A_{i-1}, (A)_0 \rightarrow A_{15}$	E	1.25 + .25N	ST
SRR	0xxx	Short Rotate Right	$(A)_{15} \rightarrow A_0, (A)_i \rightarrow A_{i+1}$	E	1.25 + .25N	ST
			$i = 0$ to CNT			
			$0 \leqslant CNT < 32$			

JUMP AND SKIP INSTRUCTIONS

COT	0650	Carry Out Test	If CRO = 1, (L) + 1 → L 0 → CRO. If CRO = 0, (L) + 2 → L.	E	1.25	ST
JMP	4xxx	Jump	EA → L	B	0.92	ST
JRT	0x30	Jump Return	(EA) → L	D	2.81	ST
JSL	Axxx	Jump and Store Location	(L) + 1 → (EA), EA + 1 → L	B	2.76	ST
OFT	0658	Overflow Test	If OVF = 1, (L) + 1 → L and 0 → OVF If OVF = 0, (L) + 2 → L.	E	1.25	ST
SKN	Exxx	Skip if A ≠ Memory	(L) + 1 → L, If (A) = (EA) (L) + 2 → L, If (A) ≠ (EA)	B	1.84	ST

CONTROL INSTRUCTIONS

ARM	0x80	Arm Interrupts	Arm selected interrupts	D	1.92	ST
CLI	0100	Clear Interrupt	Clear Current Priority Interrupt	E	0.92	ST
DIS	00C0	Disable Interrupts	Disable Interrupt System	E	0.92	ST
ENA	0080	Enable Interrupts	Enable Interrupt System	E	0.92	ST
HLT	0000	Halt	Halt	E		ST
SCO	0660	Set Carry Out	1 → CRO	E	1.00	ST
SISM	0040	Set Indirect System Mode	$0 \rightarrow ST_{10}$	E	0.95	ST
SIUM	0050	Set Indirect User Mode	$1 \rightarrow ST_{10}$	E	0.92	ST
SOF	0668	Set Overflow	1 → OVF	E	1.00	ST

FUNCTIONAL MNEMONIC LIST OF INSTRUCTIONS (CONT'D)

SYSTEM INSTRUCTIONS

MNEMONIC	OP-CODE	INSTRUCTION NAME	LOGICAL FUNCTION	FORMAT TYPE	TIME MICRO-SECONDS	INSTRUCTION SET
SRT	0x40	System Return	(EA) → L (EA + 1) → ST (EA + 2) → X (EA + 3) → A (EA + 4) → B	D	6.44	ST
SYCL	07xx	System Call	(L) → ((BA + INS_{10-15})) (ST) → ((BA + INS_{10-15})) + 1 (X) → ((BA + INS_{10-15})) + 2 (A) → ((BA + INS_{10-15})) + 3 (B) → ((BA + INS_{10-15})) + 4 0 → CRO, OVF ((BA) + INS_{10-15}) + 1 → L	SCL	7.64	ST

TSL	0x30	Test and Set Lock	If (EA) = 0, (L) + 1 → L
			If (EA) ≠ 0, (L) + 2 → L
			0 → EA
				D	3.12	ST

I/O INSTRUCTIONS

ACT	0xD0	Activate Parallel I/O		D	1.84	ST
IOC	0xC0	Input/Output Control		D	2.87 to 10.00	ST
RDP	01F0	Read Parallel	(DEVICE) → A, (L) + 2 → L if device ready.	E	1.95	ST
WTP	01E0	Write Parallel	(A) → (DEVICE), (L) + 2 → L if device ready.	E	1.95	ST

CHANNEL COMMANDS

EOA	x8xx	Execute Order in A		CC		ST
HIO	x0xx	Halt I/O		CC		ST
IDN	x000	Input Device Number		CC		ST
IIU	x800	Input Interrupt Unit		CC		ST
ISB1	x8xx	Input Status Byte 1		CC		ST
ISB2	x0xx	Input Status Byte 2		CC		ST
ISB3	x8xx	Input Status Byte 3		CC		ST
IUS	x0xx	Input Unit Status		CC		ST
OUS	x8xx	Output Unit Status		CC		ST
SDA	x0xx	Skip if Device Available		CC		ST
SDR	x8xx	Skip if Device Ready		CC		ST
SIO	xxxx	Start I/O		CC		ST
TWC	x8xx	Terminate When Complete		CC		ST
XMT	x0xx	Transmit Characters		CC		ST

APPENDIX I
NUMERIC LIST OF INSTRUCTIONS

(Basic Op Codes Without Modifiers)

Hexadecimal Code	Mnemonic
0000	HLT
0030	TSL
0040	SISM
0050	SIUM
0080	ENA
00C0	DIS
00xx	HIO
0100	CLI
0140	SRT
0180	ARM
01C0	IOC
01D0	ACT
01E0	WTP
01E1	LSMP
01E2	LUMP
01F0	RDP
0200	SAR
0220	LAR
0240	SAL
0260	LAL
0280	SLR
02A0	LLR
02C0	SRL
02E0	LLL
0300	SRR
0308	XHA
0320	LRR
0340	SRL
0360	LRL
0380	SCR
03A0	LCR
03C0	SCL
03E0	LCL
0400	NOP
04xx	RADD
04xx	RCPY
04xx	RINC
04xx	RXOR
05xx	RCMP
05xx	RDEC
05xx	RNEG
05xx	RSUB
0600	AAB
0608	ADC
0610	AOB
0618	XAX
0620	ESA
0628	LLO
0630	JRT
0638	STX
0640	LAS
0648	LDS
0650	COT
0658	OFT
0660	SCO
0668	SOF
0670	XBX
0678	LDX
0680	LDD
0688	LDF
0690	STD
0698	STF
06A0	DAD
06A8	DSB
06B0	DMP
06B8	DDV
06C0	FAD
06C8	DFA
06D0	FSB
06D8	DFS
06E0	FMP
06F8	DFD
0700	MPY
0708	DIV
0710	NDX
0720	XAB
0728	MPX
0740	SYCL
0xxx	SIO
10xx	XMT
18xx	EOA
1xxx	LDH
2000	IDN
28xx	OUS
2xxx	LDB
38xx	TWC
3xxx	STB
48xx	SDR
4xxx	JMP
50xx	SDA
5800	IIU
5xxx	STH
60xx	IUS
60xx	ANL
62xx	XOL
64xx	LDL
66xx	ADL
68xx	ANLB
68xx	ISB1
6Axx	XOLB
6Cxx	LDLB
6Exx	ADLB
70xx	ISB2
78xx	ISB3
7xxx	STA
8xxx	MIN
9xxx	ADD
Axxx	JSL
Bxxx	XOR
Cxxx	LDA
Dxxx	SUB
Exxx	SKN
Fxxx	AND

APPENDIX J

ALPHABETIC INSTRUCTION LIST

MNEMONIC	INSTRUCTION NAME	OPERATION CODE HEX.	OPERATION CODE BINARY	TIME MICRO-SECONDS	PAGE
AAB	AND Accumulator and B Register	0600	0000 0110 0000 0000	1.15	3-26
ACT	Activate Channel	0xD0	0000 x001 1101 0000	1.84	4-2
ADC	Add Carry	0608	0000 0110 0000 1000	1.25	3-20
ADD	Add to Accumulator	9xxx	1001 xxxx xxxx xxxx	1.84	3-18
ADL	Add to A, Literal	6xxx	0110 011x xxxx xxxx	1.15	3-19
ADLB	Add to B, Literal	6xxx	0110 111x xxxx xxxx	1.15	3-19
AND	AND with Accumulator	Fxxx	1111 xxxx xxxx xxxx	1.84	3-26
ANL	AND with Accumulator, Literal	6xxx	0110 000x xxxx xxxx	1.15	3-27
ANLB	AND with B Register, Literal	6xxx	0110 100x xxxx xxxx	1.15	3-27
AOB	OR Accumulator with B Register	0610	0000 0110 0001 0000	1.15	3-26
ARM	Arm Interrupts	0x80	0000 x001 1000 0000	1.92	3-44
CLI	Clear Interrupt	0100	0000 0001 0000 0000	0.92	3-44
COT	Carryout Test	0650	0000 0110 0101 0000	1.25	3-42
DAD	Double Add	0xA0	0000 x110 1010 0000	3.96	3-21
DDV	Double Divide	0xB8	0000 x110 1011 1000	32.81 to 44.81	3-22
DFA	Double Floating Add	0xC8	0000 x110 1100 1000	7.88 to 9.03**	3-24
DFD	Double Floating Divide	0xF8	0000 x110 1111 1000	34.68 to 39.58**	3-25
DFM	Double Floating Multiply	0xE8	0000 x110 1110 1000	26.28 to 27.63**	3-25
DFS	Double Floating Subtract	0xD8	0000 x110 1101 1000	7.88 to 9.03**	3-24
DIS	Disable Interrupts	00C0	0000 0000 1100 0000	0.92	3-43
DIV	Divide	0x08	0000 x111 0000 1000	8.14 to 8.99	3-21
DMP	Double Multiply	0xB0	0000 x110 1011 0000	25.30	3-22
DSB	Double Subtract	0xA8	0000 x110 1010 1000	3.91	3-22
ENA	Enable Interrupts	0080	0000 0000 1000 0000	0.92	3-43
EOA	Execute Order in A	x8xx	x001 1000 xxxx xxxx	*	4-5
ESA	Extend Sign of Accumulator	0620	0000 0110 0010 0000	1.05	3-28
FAD	Floating Point Add	0xC0	0000 x110 1100 0000	9.11 to 12.01‡¤	3-23
FDV	Floating Point Divide	0xF0	0000 x110 1111 0000	37.16 to 44.41	3-24
FMP	Floating Point Multiply	0xE0	0000 x110 1110 0000	31.99 to 33.49	3-23
FSB	Floating Point Subtract	0xD0	0000 x110 1101 0000	9.81 to 13.01‡¤	3-23

* I/O Commands.

** +.21N for Alignment.

‡ +.25N for Alignment.

¤ +.55N for Normalize.

ALPHABETIC INSTRUCTION LIST (CONT'D)

MNEMONIC	INSTRUCTION NAME	OPERATION CODE HEX.	OPERATION CODE BINARY	TIME MICRO-SECONDS	PAGE
HIO	Halt I/O	x0xx	x000 0000 xxxx xxxx	*	4-4
HLT	Halt	0000	0000 0000 0000 0000		3-43
IDN	Input Device Number	x000	x010 0000 0000 0000	*	4-6
IIU	Input Interrupting Unit	x800	x101 1000 0000 0000	*	4-7
IOC	I/O Control	0xC0	0000 x001 1100 xxxx	2.87 to 10.00	4-3
ISB1	Input Status Byte 1	x8xx	x110 1000 xxxx xxxx	*	4-7
ISB2	Input Status Byte 2	x0xx	x111 0000 xxxx xxxx	*	4-7
ISB3	Input Status Byte 3	x8xx	x111 1000 xxxx xxxx	*	4-8
IUS	Input Unit Status	x0xx	x110 0000 xxxx xxxx	*	4-7
JMP	Jump	4xxx	0100 xxxx xxxx xxxx	0.92	3-41
JRT	Jump Return	0x30	0000 x110 0011 0000	2.81	3-41
JSL	Jump and Store Location	Axxx	1010 xxxx xxxx xxxx	2.76	3-41
LAL	Long Arithmetic Left Shift	0xxx	0000 x010 011x xxxx	1.25 + .25N	3-33
LAR	Long Arithmetic Right Shift	0xxx	0000 x010 010x xxxx	1.25 + .25N	3-33
LAS	Load Accumulator from Switches	0640	0000 0110 0100 0000	1.00	3-16
LCL	Long Circulate Left	0xxx	0000 x011 111x xxxx	1.25 + .25N	3-39
LCR	Long Circulate Right	0xxx	0000 x011 101x xxxx	1.25 + .25N	3-39
LDA	Load Accumulator	Cxxx	1100 xxxx xxxx xxxx	1.84	3-14
LDB	Load B Register	2xxx	0010 xxxx xxxx xxxx	1.84	3-14
LDD	Load Double	0x80	0000 x110 1000 0000	3.68	3-17
LDF	Load Floating	0x88	0000 x110 1000 1000	4.60	3-17
LDH	Load Halfword	1xxx	0001 xxxx xxxx xxxx	2.59 or 3.01***	3-15
LDL	Load A, Literal	6xxx	0110 010x xxxx xxxx	1.05	3-16
LDLB	Load B, Literal	6xxx	0110 110x xxxx xxxx	1.05	3-16
LDS	Load Status A	0648	0000 0110 0100 1000	1.00	3-16
LDX	Load Index	0x78	0000 x110 0111 1000	2.81	3-15
LLL	Long Logical Left Shift	0xxx	0000 x010 111x xxxx	1.25 + .25N	3-35
LLO	Locate Leading One	0628	0000 0110 0010 1000	1.25 to 1.85 + .20N	3-40
LLR	Long Logical Right Shift	0xxx	0000 x010 101x xxxx	1.25 + .25N	3-35
LRL	Long Rotate Left	0xxx	0000 x011 011x xxxx	1.25 + .25N	3-37
LRR	Long Rotate Right	0xxx	0000 x011 001x xxxx	1.25 + .25N	3-37
LSMP	Load System Map Pointer	01E1	0000 0001 1110 0001	1.95	3-18
LUMP	Load User Map Pointer	01E2	0000 0001 1110 0010	1.95	3-18

* I/O Commands.

*** Indirect Address Time.

ALPHABETIC INSTRUCTION LIST (CONT'D)

MNEMONIC	INSTRUCTION NAME	HEX.	OPERATION CODE BINARY	TIME MICRO-SECONDS	PAGE
MIN	Memory Increment, Skip on Zero	8xxx	1000 xxxx xxxx xxxx	2.14	3-20
MPX	Multiply Index	0728	0000 1000 0010 0111	1.20	3-21
MPY	Multiply	0x00	0000 0000 0000 x111	8.44	3-20
NDX	Normalize & Decrement Index	0710	0000 0000 0001 0111	1.45 to 1.65 + .4N	3-40
NOP	No Operation	0400	0000 0000 0000 0100	1.25	3-31
OFT	Overflow Test	0658	0000 1000 0101 0110	1.25	3-42
OUS	Output Unit Status	x800	x010 xxxx xxxx 1000	*	4-6
RADD	Register Add	04xx	0000 xxxx 10xx 0100	1.45	3-30
RCMP	Register Complement	03xx	0000 xxxx 00xx 0101	1.25	3-30
RCPY	Register Copy	04xx	0000 xxxx 00xx 0100	1.25	3-30
RDEC	Register Decrement	05xx	0000 xxxx 10xx 0101	1.25	3-31
RDP	Read Parallel	01F0	0000 0000 1111 0001	1.95	4-3
RINC	Register Increment	04xx	0000 xxxx 01xx 0100	1.25	3-31
RNEG	Register Negate	05xx	0000 xxxx 01xx 0101	1.25	3-30
RSUB	Register Subtract	05xx	0000 xxxx 11xx 0101	1.45	3-30
RXOR	Register Exclusive OR	04xx	0000 xxxx 11xx 0100	1.45	3-31
SAL	Short Arithmetic Left Shift	0xxx	0000 xxxx 010x x010	1.25 + .25N	3-32
SAR	Short Arithmetic Right Shift	0xxx	0000 xxxx 000x x010	1.25 + .25N	3-32
SCL	Short Circulate Left	0xxx	0000 xxxx 110x x011	1.25 + .25N	3-38
SCO	Set Carryout	0660	0000 0000 0110 0110	1.00	3-45
SCR	Short Circulate Right	0xxx	0000 xxxx 100x x011	1.25 + .25N	3-38
SDA	Skip if Device Available	x0xx	x101 xxxx xxxx 0000	*	4-7
SDR	Skip if Device Ready	x8xx	x100 xxxx xxxx 1000	*	4-6
SIO	Start I/O	xxxx	x000 xxxx xxxx 1xx0	*	4-5
SISM	Set Indirect System Mode	0040	0000 0000 0100 0000	0.92	3-45
SIUM	Set Indirect User Mode	0050	0000 0000 0101 0000	0.92	3-45
SKN	Skip if A ≠ Memory	Exxx	1110 xxxx xxxx xxxx	1.84	3-42
SLL	Short Logical Left Shift	0xxx	0000 xxxx 110x x010	1.25 + .25N	3-34
SLR	Short Logical Right Shift	0xxx	0000 xxxx 100x x010	1.25 + .25N	3-34
SOF	Set Overflow	0668	0000 1000 0110 0110	1.00	3-45
SRL	Short Rotate Left	0xxx	0000 xxxx 010x x011	1.25 + .25N	3-36
SRR	Short Rotate Right	0xxx	0000 xxxx 000x x011	1.25 + .25N	3-36
SRT	System Return	0x40	0000 0000 0100 x001	6.44	3-46
STA	Store Accumulator	7xxx	0111 xxxx xxxx xxxx	1.84	3-14

* I/O Commands.

ALPHABETIC INSTRUCTION LIST (CONT'D)

MNEMONIC	INSTRUCTION NAME	OPERATION CODE HEX.	OPERATION CODE BINARY	TIME MICRO-SECONDS	PAGE
STB	Store B Register	3xxx	0011 xxxx xxxx xxxx	1.84	3-15
STD	Store Double	0x90	0000 0110 1001 0000	3.68	3-17
STF	Store Floating	0x98	0000 x110 1001 1000	4.60	3-17
STH	Store Halfword	5xxx	0101 xxxx xxxx xxxx	2.89 or 3.31***	3-16
STX	Store Index	0x38	0000 x110 0011 1000	2.81	3-15
SUB	Subtract from Accumulator	Dxxx	1101 xxxx xxxx xxxx	1.84	3-19
SYCL	System Call	07xx	0000 0111 01xx xxxx	7.64	3-46
TSL	Test and Set Lock	0x30	0000 x111 0011 0000	3.12	3-47
TWC	Terminate When Complete	x8xx	x011 1000 xxxx xxxx	*	4-6
WTP	Write Parallel	01E0	0000 0001 1110 0000	1.95	4-2
XAB	Exchange Accumulator & B Register	0720	0000 0111 0010 0000	1.15	3-28
XAX	Exchange Accumulator & Index	0618	0000 0110 0001 1000	1.15	3-27
XBX	Exchange B and Index Register	0670	0000 0110 0111 0000	1.15	3-28
XHA	Exchange Halves of Accumulator	0308	0000 0011 0000 1000	3.25	3-28
XMT	Transmit	x0xx	x001 0000 xxxx xxxx	*	4-5
XOL	Exclusive OR the Accumulator, Literal	6xxx	0110 001x xxxx xxxx	1.15	3-27
XOLB	Exclusive OR the B Register, Literal	6xxx	0110 101x xxxx xxxx	1.15	3-27
XOR	Exclusive OR with Accumulator	Bxxx	1011 xxxx xxxx xxxx	1.84	3-26

* I/O Commands.

*** Indirect Address Time.

All specifications subject to change without notice.

Scientific Control Corporation

Dallas, Texas 214–242-6555 TWX–910-860-5956

Order by part number from above address or call the nearest sales office for additional information.

EASTERN

REGIONAL OFFICE

Washington, D.C.
301–779-3330

DISTRICT OFFICES

College Park, Md.
301–779-2510

Orlando, Fla.
305–855-5833

New York, N.Y.
201–335-3001

Boston, Mass.
617–449-0880

CENTRAL

REGIONAL OFFICE

Dallas, Texas
214–358-1331

DISTRICT OFFICES

Dallas, Texas
214–358-1332

Houston, Texas
713–526-5721

Chicago, Ill.
312–297-2470

Denver, Colorado
303–322-0516

Detroit, Mich.
Branch Office

WESTERN

REGIONAL OFFICE

San Francisco, Calif.
415–328-8980

DISTRICT OFFICES

Palo Alto, Calif.
415–328-8981

Los Angeles, Calif.
213–443-0143