10077-00A0
APRIL 1973

NAKED MINI LSI/ALPHA LSI
PROGRAMMING REFERENCE MANUAL

COMPUTER AUTOMATION, INC.
the NAKED MINI company
18651 Von Karman, Irvine, Calif. 92664
tel. 714-833-8830   TWX 910-595-1767

TABLE OF CONTENTS

Section | Page

Section 1. GENERAL DESCRIPTION

TABLE OF CONTENTS (Cont'd)

Section Page

Section 2. CONSOLE

Section 3. INSTRUCTIONS AND DIRECTIVES

TABLE OF CONTENTS (Cont'd)

Section Page

Section 3. INSTRUCTIONS AND DIRECTIVES (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

Section Page

Section 4. INPUT/OUTPUT AND INTERRUPT OPERATION

Section 5. PROCESSOR OPTIONS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

Section 1
GENERAL DESCRIPTION

1.1 INTRODUCTION

The ALPHA 16 LSI and NAKED MINI 16 LSI (hereafter referred to as ALPHA LSI when discussed together) are general purpose, stored program digital computers. They are extensions of the successful and proven 16-bit computer family from Computer Automation.

1.1.1 Upward Compatibility

The ALPHA LSI is upward software and I/O compatible with earlier 16-bit computers from Computer Automation. Upward software compatibility means that virtually all programs written for the earlier 16-bit computers will run without change on the ALPHA LSI. However, due to the expanded and improved instruction set of the ALPHA LSI, programs written for these computers may not run on the earlier computers.

1.1.2 General Features

The ALPHA LSI computer features a 16-bit word format and 162 basic instructions. The instruction set is divided into seven major classes which provide memory-to-register and register-to-register data movement as well as conditional jump, single and double-register shift, register change, machine control and Input/Output instructions. The computer utilizes eight addressing modes for effective and efficient management of memory resources.

The ALPHA LSI computer has a fully buffered I/O structure coupled with five levels of interrupts and five I/O modes which permit high speed, low speed, synchronous and asynchronous data transfers to take place.

The ALPHA LSI may readily accommodate additional memory and I/O by adding expansion chassis to the basic system. An optional Memory Banking feature permits the user to extend the upper limit of memory from 32K words to 256K words.

1.2 THE NAKED MINI 16 LSI CONCEPT

The NAKED MINI 16 LSI computer consists of the Processor and first memory module on one printed circuit board. The NAKED MINI 16 LSI is a complete stand alone computer without a chassis, motherboard, power supply or operators console.

The NAKED MINI 16 LSI computer is designed to be used as a system component along with other system components. It depends on the system power supply for a power source, the system control panel for operational control signals and the system enclosure for structural and environmental support.

1.3 THE ALPHA 16 LSI

Take a NAKED MINI 16 LSI computer and add a power supply module, a motherboard, a chassis and an operator's console and you get the ALPHA 16 LSI computer. The Motherboard interconnects the NAKED MINI 16 LSI computer with additional I/O and memory modules, the power supply and the operator's console.

1.4 CHARACTERISTICS

The characteristics of the ALPHA LSI are explained in subsequent sections of this manual. The following is an overview of the characteristics of this computer.

1.4.1 Processor

Some of the significant characteristics of the computer are:

Parallel processing of full 16-bit words and 8-bit bytes

Seven 16-bit hardware registers, one 8-bit Status Register

Memory word size of 16 bits, with each word addressable as a full 16-bit word or as two separate 8-bit bytes.

Memory capacity is 1,024 words minimum, expandable to 32,768 words per bank maximum (Up to 262,144 words with optional memory banking.)

Computer cycle time is 1.6 microseconds

Direct Memory access (Standard) provides data transfer rates of 625,000 words per second in a single memory bank or 1,250,000 words per second with interleaved memory banks

Binary 2's complement arithmetic processing

Automatic memory scan (standard)

Hardware Multiply and Divide (standard)

1.4.2 Instruction Set

These computers have a very powerful instruction set consisting of 162 basic instructions divided into seven classes. The instruction classes are:

1. Memory Reference.

   Access memory in either full word or byte mode and perform logical and arithmetic operations involving data in memory and data in hardware registers. The hardware multiply, divide and normalize instructions are included in this class.

2. Byte Immediate

   Similar to memory reference in that they perform logical and arithmetic operations involving data in hardware registers. The memory data, however, is contained within the instruction word so that it is immediately available for processing without requiring an operand cycle to fetch it from memory.

3. Conditional Jump.

   Test conditions within the processor and perform conditional branches depending on the results of the tests performed. Jumps may be as much as ± 64 locations from the location of the conditional jump instruction.

4. Shift.

   Include single-register logical, arithmetic and rotate shifts; double-register logical and rotate shifts.

5. Register Change.

   Provide logical manipulation of data within hardware registers.

6. Control.

   Enable and disable interrupts; suppress status, control word or byte mode data processing and perform other general control functions.

7. Input/Output.

   Provide communications between the computer and external devices. They include conventional I/O instructions plus Block Transfer and Automatic Input/Output instructions. I/O may be to/from register or directly to/from memory.

1.4.3 Memory Addressing

An important feature of these machines is the ability to access full 16-bit words and 8-bit bytes (half words) in memory. Memory may be as small as 1K x 16-bit words, and as large as 32K x 16-bit words. Since memory may contain 32K words, and since each word contains two bytes, provisions are made for addressing up to 64K bytes.

Instructions which access memory may operate in either word or byte mode. Memory reference instructions are sixteen bits in length (one-word instructions), with the eight least-significant bits plus three control bits dedicated to memory addressing. The eight least significant bits address 256 words or bytes. The ALPHA LSI computer uses the three control bits to specify several addressing modes. These addressing modes are discussed briefly below and are explained in detail in Section 3. The addressing modes used are Scratchpad, Relative Forward, Relative Backward, Indexed, and Indirect.

1. Scratchpad

   Scratchpad addressing accesses the first 256 words in memory in Word Mode, or the first 256 bytes in Byte Mode. The first 256 words in memory are referred to as "Scratchpad" memory, because these are common words which can be addressed directly by instructions located anywhere in memory.

2. Relative

   In Word Mode, relative addressing can address an area of memory extending from the instruction address forward 256 words (+256) or backward 255 words (-255). In Byte Mode, the range is forward 512 bytes. Bytes cannot be directly addressed relative backward.

3. Indexed

   The Index register (X register) can be added to the address field of memory reference instructions to form an effective memory word or byte address.

4. Indirect

   Indirect addressing uses scratchpad or relative addressing to access a word in memory which contains the address of a memory operand. The word that contains a memory address rather than an operand is called an address pointer. In Word Mode, multi-level indirect addressing is possible; i.e., one address pointer may contain the address of another address pointer rather than the address of an operand. In Byte Mode, only one level of indirect addressing is possible.

   Indirect addressing may also be used in conjunction with indexing. When indexed indirect addressing is specified, the indirect operation is performed first and then the contents of the X register are added to the contents of the address pointer. This process is called Post Indexing.

1.4.4 I/O Structure

The ALPHA LSI computer has a parallel I/O structure that provides both ease of interfacing and powerful peripheral control. Some special features of the I/O Structure are:

1. Vectored Interrupts.

   These machines feature vectored hardware priority interrupts. Wherein each peripheral controller supplies its own unique interrupt address to any location in memory. There are five standard interrupt levels (two internal and three external). The third external level with control lines can accommodate a virtually unlimited number of vectored interrupts.

2. Direct Memory Channels.

   Direct memory channels (DMC) provide data transfers between the computer and peripheral components without affecting the operating registers of the computer. DMC's are a standard feature of these computers. The maximum data transfer rate using DMC's under interrupt control is 26,738 words/sec.

3. Block Input/Output.

   The Block I/O feature of these computers dedicates the computer to I/O data transfer at the maximum possible transfer rate. The maximum transfer rate using Block I/O is 131,579 words/sec. Block I/O is a standard feature of these computers.

4. Parallel Busses

   Separate busses providing device address selection, data transfer, and control signals are used for ease of interfacing. Busses are not time shared for I/O functions. This feature alone simplifies interface design considerably.

1.4.5 Processor Options

There are four general options that are offered with the ALPHA LSI computer. They are: Power Fail/Restart; the Teletype/CRT Interface; Real-time Clock, and Autoload.

The Power Fail/Restart option mounts directly on the NAKED MINI 16 LSI computer printed circuit board. The other three options mount on a special option board which plugs into a special connector (in piggyback fashion) on the NAKED MINI 16 LSI computer printed circuit board. None of these options interface directly with the motherboard.

1. Teletype/CRT Interface

   Interfaces a modified ASR-33 or ASR-35 Teletype or CRT terminal to the computer. This is a fully-buffered interface that includes remote Teletype motor on/off control. In addition to the standard TTY baud rate (110 baud), nine user selectable baud rates, ranging from 75 to 9600 bauds, are provided for driving a CRT terminal.

2. Power Fail Restart

   This option includes the hardware necessary to detect low input power conditions and bring the computer to an orderly halt until normal input power is restored. When normal power is restored, this option will generate an orderly restart. The Power Fail Restart option allows completely unattended operation of the computer at locations where power conditions are unreliable.

3. Real Time Clock

   The Real Time Clock option features a crystal controlled internal clock which may be wired to produce clock rates of 100 microseconds, 1 millisecond, 10 milliseconds, or twice the input AC line frequency (8.33 or 10 milliseconds -60Hz and 50Hz, respectively). The 10 millisecond (crystal derived) rate is standard. An external clock source may also be used. The Real Time Clock provides time-of-day information to the computer and may be used to time periodic events that must be controlled by the computer.

4. Multi-Device Autoload

   The Multi-Device Autoload option consists of a Read-Only Memory (ROM) programmed with a complete binary loader which is capable of loading binary programs from any one of several input devices. The Autoload hardware consists of the ROM when the LOAD switch is activated.

1.4.6 Plug-In Options

Locations are provided within the ALPHA 16 LSI computer chassis for the installation of processor options, peripheral interfaces, and memory modules. The options are mounted on printed circuit boards which plug into the locations within the computer chassis. Some of the available plug-in processor options are:

1. DTL I/O buffers, up to 64 bits

2. Relay I/O buffers, up to 32 isolated relays

3. Modem interfaces: non-synchronous and synchronous

4. Memory Banking Controller, extends upper limit of memory to 262,144 words.

5. Read Only Memory (ROM)

1.4.7 Peripheral Equipment

The following is a partial list of the various types of peripheral equipment for which interfaces to the ALPHA LSI have been developed. This list does not imply that these are the only devices for which interfaces can be developed. The interface structure of these computers is such that virtually any peripheral device can be interfaced to the computer.

1. ASR-33 and ASR-35 Teletypewriters
2. High speed paper tape readers and punches
3. Line printers
4. Card readers
5. Open reel and cassette magnetic tape units
6. Magnetic disks
7. A/D and D/A converters
8. CRT terminals
9. Plotters

1.5 DATA HANDLING CHARACTERISTICS

1.5.1 Data Word Format

Processor registers and memory word locations are capable of storing data words consisting of 16 binary digits or "bits". A word may be handled as a single 16-bit field or as two 8-bit bytes. The following paragraphs describe the word format of the computer. Byte format is described later in this section..

1.5.1.1 Bit Identification

A data word may contain a single number, or it may contain a string of individual binary bits, with each bit having a unique meaning. For purposes of explanation and identification, each bit within a word is uniquely identified. The identification is accomplished by numbering each bit within a word from right to left. The bit on the extreme right of the word is bit 0, and the bit on the extreme left is bit 15. Figure 1-1 illustrates the format of a 16-bit data word with the bit number shown above the bit position.

1.5.1.2 Bit Values

The ALPHA LSI is a binary computer, therefore numeric information stored in the computer and processed by the computer must be in binary format. Figure 1-1 illustrates the binary value of a one-bit in each bit position of the 16-bit data word. These values are expressed as powers of two. For example, a one-bit in bit position 3 has the value of 2^3 , or 8. The single exception to this rule is bit position 15 which is the sign bit.

1.5.1.3 Signed Numbers

The ALPHA LSI is capable of performing arithmetic operations with signed numbers. Binary two's complement notation is used to represent and process numeric information. Bit 15 of a data word indicates the algebraic sign of the number contained within that word.

1.5.1.4 Positive Numbers

A positive number is identified by a 0 in bit 15, and the binary equivalent of the magnitude of the positive number is stored in bits 0 to 14. The largest positive signed number which can be stored in a 16-bit word is $+32,767_{10}$.

1.5.1.5 Negative Numbers

A negative number is identified by a 1 in bit 15 of the data word. A negative number is represented by the binary two's complement of the equivalent positive number. A negative number must follow the mathematical rule where:

$$0-(+n) = -n$$

For example:

$$0 - (+5) = -5$$

Negative numbers must also be constructed such that:

$$(+n) + (-n) = 0$$

The binary two's complement of some numeric value may be constructed by subtracting the binary representation of the absolute magnitude of that value from 0.

Note that the formation of a binary two's complement negative number from the equivalent positive number automatically sets the sign bit to a one. The largest negative number that can be stored in a 16-bit word is $-32,768_{10}$.

1.5.2 Data Byte Format

A 16-bit data word is capable of storing two 8-bit bytes. Since most data transfers between mini computers and peripheral devices are in the form of bytes rather than words, the ALPHA LSI computer provides the capability of addressing individual bytes as well as full data words. Figure 1-2 illustrates the storage of two bytes within one computer word.

Bit positions within bytes are identified much the same as in 16-bit words. Figure 1-2 also illustrates the numbering of data bits within a byte. The bits are numbered 0 through 7, where bit 0 is the least-significant bit (LSB), and bit 7 is the most-significant bit (MSB) of the byte.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 1-1. Data Word Bit Identification

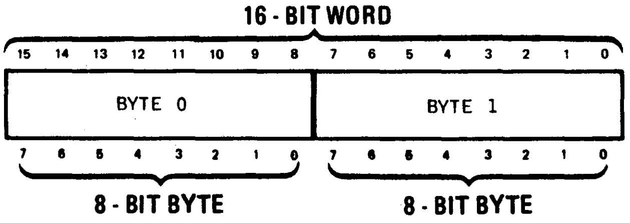


Figure 1-2. Byte Storage, Two Bytes Per Word

1.5.2.1 Byte Mode Processing

There are two control instructions in the computer which control Word Mode processing and Byte Mode processing. One of the instructions causes the computer to enter Byte Mode processing, and the other causes the computer to enter Word Mode.

In Word Mode, all memory reference instructions access full words in memory. In Byte Mode all memory reference instructions (except IMS, MPY, DIV, NRM, JMP, and JST) access one byte within a word. The method of addressing individual bytes is discussed in a subsequent part of this Section. The present discussion is concerned with computer operations while in Byte Mode as contrasted with computer operations in Word Mode.

Byte Mode affects the operand cycle of the computer only. All other computer functions operate the same as in Word Mode. In Byte Mode the computer operand cycle reads a single byte from memory instead of a full word. The following paragraphs illustrate Byte Mode operations for memory reference instructions.

1.5.2.2 Register Load

In Word Mode, the full word is loaded into the selected register. In Byte Mode, the selected byte is loaded into the lower eight bits of the selected register and the upper eight bits are set to zero. Note that the location of the byte within the memory word does not determine the location the byte will occupy in the register being loaded.

1.5.2.3 Arithmetic Operations

For arithmetic purposes, bytes are handled as positive numbers only. The reason is that a byte occupies the lower eight bits of a register, or a data bus, and the upper eight bits contain zeros.

1.5.2.4 Data Packing

One of the most useful features of byte mode processing is in the packing and unpacking of data in memory. Since most of the peripheral devices used with mini computers are byte oriented, high-speed data transfers between the computer and the peripheral device generally require data to be packed one byte per word. Such an arrangement is illustrated in Figure 1-3. In this illustration, the upper eight bits of each data word to be transmitted to a peripheral device contain zeros. A full 16-bit word is transmitted to the device, but the device discards the upper eight bits and accepts only the lower eight bits. Data received from a byte oriented peripheral device during high-speed data transfers is packed in memory one byte per word in the format shown in Figure 1-3. If a software subroutine were required to pack the data two bytes per word, in the format illustrated in Figure 1-4, it would waste memory and time in performing the formatting required for high-speed data transfers.

The capability of the ALPHA LSI computer to address individual bytes in memory allows high speed data transfers using the memory format shown in Figure 1-4 for both transmission and reception of data. Bytes may be addressed sequentially and transmitted or received sequentially, just as words are transmitted or received sequentially in conventional unpacked data transfers. This arrangement saves memory space since none of the memory word is wasted, and it saves time since no software routines are required to pack and unpack data for internal processing.

1.5.3 Memory Address Formats

Maximum memory capacity in the ALPHA LSI computer is 32,768 words which means a byte capacity of 65,536 bytes. A fifteen bit address is required to address 32,768 words, and a sixteen bit address is required to address 65,536 bytes. The following paragraphs discuss the formats of the addresses that must be presented to memory for addressing both words and bytes. This discussion is concerned only with address formats. Section 3 of this manual discusses the memory address modes which form these addresses.

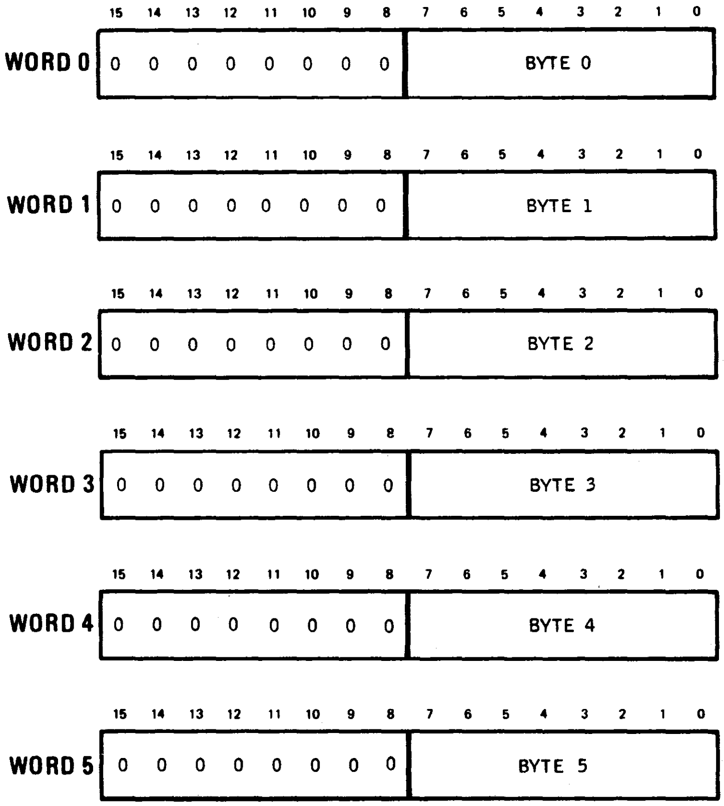


Figure 1-3. Data in Memory, One Byte Per Word

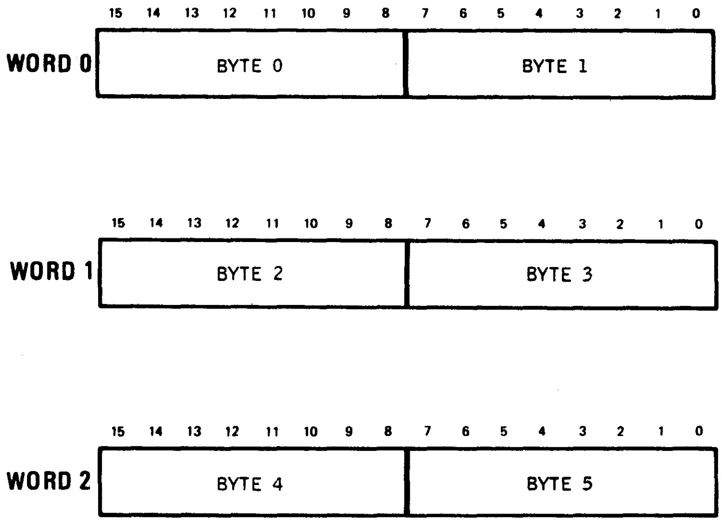


Figure 1-4. Data in Memory, Two Bytes Per Word

1.5.3.1 Word Addressing

Figure 1-5 illustrates the format of an address presented to memory to address a full word. This is the format that is used to address instructions or full data words. The address is contained in bits 0 - 14, and bit 15 contains a zero.

1.5.3.2 Byte Addressing

Figure 1-6 illustrates the format used to address a byte within a data word. Bits 1-15 contain the address of the memory word, and bit 0 specifies which byte within the word is to be addressed.

Bit 0 = 0 specifies Byte 0 (Most Significant Byte).

Bit 0 = 1 specifies Byte 1 (Least Significant Byte).

If the computer is set for Byte Mode, all operand addresses presented to memory are assumed to be byte addresses. The computer assumes that the address is in the format shown in Figure 1-6. If the computer is set for word mode processing, all addresses presented to memory are assumed to be word addresses in the format shown in Figure 1-5. These assumptions apply to operand cycles only. They do not apply to instruction cycles or indirect addressing cycles.

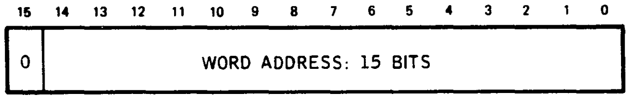


Figure 1-5. Basic Word Address Format

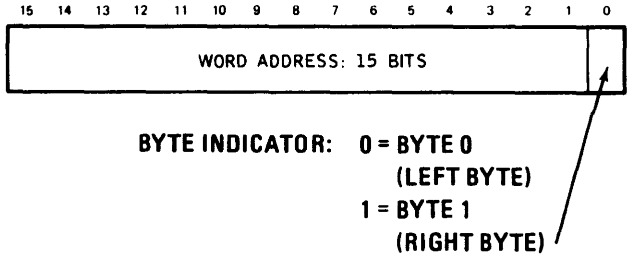


Figure 1-6. Byte Address Format

1.5.3.3 Indirect Addressing

The ALPHA LSI computer is capable of performing single level indirect addressing for addressing bytes, and multi-level indirect addressing for addressing words. Indirect addressing uses direct addressing to read a word in memory, called an address pointer, which contains the address of another word. In Byte Mode the address pointer contains the address of the byte to be addressed. The format of the address in the address pointer is the same as that shown in Figure 1-6.

In Word Mode the format of the address in the address pointer is that shown in Figure 1-7. Bits 0 - 14 contain the address of another word in memory. Bit 15 is a multilevel indicator. If bit 15 contains a 0 the address in bits 0 - 14 is the address of another indirect address pointer. The number of levels of indirect addressing which may be used is limited only by the size of memory.

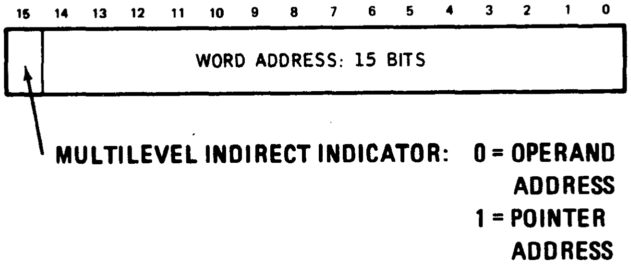


Figure 1-7. Indirect Address Pointer Format

Section 2
CONSOLE

2.1 INTRODUCTION

The ALPHA 16 LSI Console provides the switches and indicators required to operate, display and control the computer. This section describes the controls and indicators on the Console, provides operating procedures and defines machine modes.

2.2 SWITCHES AND INDICATORS

For the convenience of the user, the switches and indicators have been grouped into the following sections:

1. Status
2. Control
3. Entry and Display

Figure 2-1 illustrates the ALPHA 16 LSI Console. All switches and indicators are listed and explained in table 2-1.

NOTE

All console switches are momentary contact touch switches and all indicators are light-emitting diodes (LED's).

Table 2-1. Console Switches and Indicators

SWITCH OR INDICATOR	PURPOSE
System Status Section	
ON Indicator	On when power is applied, off when power is removed. The main power switch is located on the rear of the computer.
ENABLE Slide Switch and Indicator	The console enable/disable slide switch is located in a recess on the edge of the console. When the switch is on, the ENABLE indicator is on. Likewise, when the switch is off the indicator is off. When in the ENABLE state all switches and indicators are enabled. When in the disabled state the only functions that are effective are:

Table 2-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
	1. The SENSE switch and indicator 2. The Console Sense register, Register Display indicator and hex entry keyboard when the SENSE/DATA indicator is on.
BYTE Indicator	On when the Processor is in Byte Mode. Off when the Processor is in Word Mode.
OV Indicator	On when the Processor overflow flip-flop is on. Off when the overflow flip-flop is off.
SENSE Switch and Indicator	The SENSE Switch toggles the SENSE Indicator. The SENSE Indicator may be tested by program instructions. The Sense test will be true if the SENSE Indicator is on.
System Control Section	
STOP Switch and Indicator	The STOP Switch toggles the STOP Indicator. The Indicator is on when the Stop Mode is established. When the indicator is off the Run Enable Mode is established. When the Stop Mode is established and the console is enabled (ENABLE indicator on), data entry and display operations may be performed. In addition, the Processor will fetch and execute one program instruction each time the RUN switch is pressed. When in the Run Enable Mode, data entry and display operations may not be performed and the Run Mode is enabled but not entered until the RUN switch is pressed.
RESET Switch and Indicator	The indicator is on when the RESET switch is on and remains on only as long as the switch is pressed. The RESET switch generates a system reset signal which causes the Processor and all interfaces to be initialized. The RESET switch should not normally be used to stop the computer. If RESET is pressed while the computer is running, the instruction currently being executed may not complete. The STOP switch should normally be used to halt the computer. The only time that RESET should be used to halt the computer is in the case where the Processor is hung up in a non-escapable one instruction loop (e.g., multi-level indirect address instruction with closed address chain).

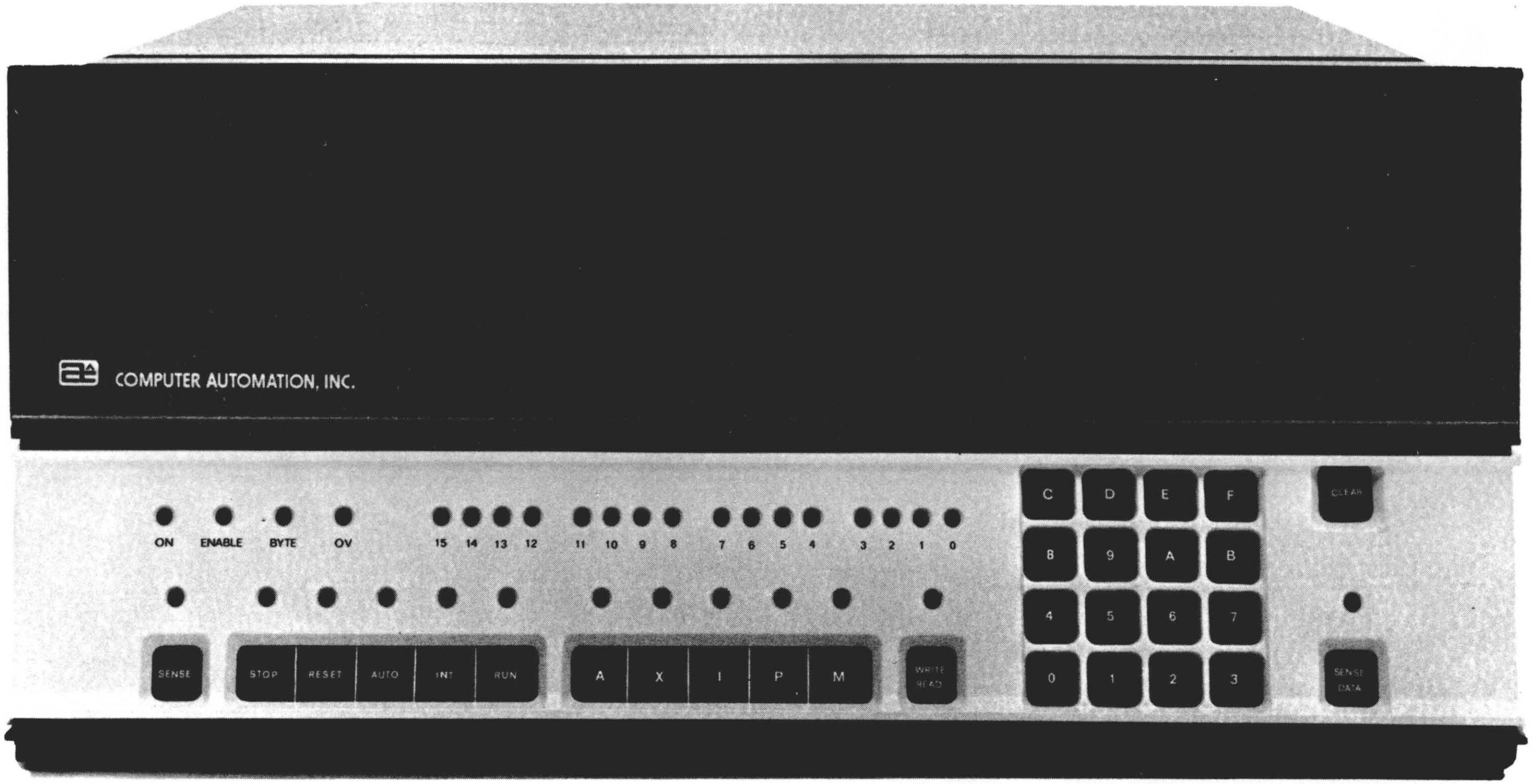


Figure 2-1. ALPHA 16 LSI Console

Table 2-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
AUTO Switch and Indicator	The AUTO switch is used to initiate an Autoload sequence if the Autoload option is installed. The AUTO switch is enabled only during the Run Enable Mode. Depressing the switch establishes the Run Mode and initiates the Autoload sequence. The indicator turns on when the switch is pressed and remains on until the Autoload sequence is completed.
INT Switch and Indicator	The INT switch is used to initiate a Console Interrupt. The switch is enabled only during the Run Mode. The indicator turns on when the switch is pressed and remains on until the Processor honors the Console Interrupt Request.
RUN Switch and Indicator	The RUN switch is used to establish the Run Mode when the STOP indicator is off. When the STOP indicator is on, the RUN switch causes one instruction to be fetched and executed when pressed. The WRITE indicator is turned off whenever RUN is pressed. The RUN indicator is turned on when in the Run Mode.
Entry/Display Section	
Register Display Indicators (0 thru 15)	The 16 Register Display Indicators display the contents of either the Console Data register or the Console Sense register depending on the state of the SENSE/DATA indicator. When the SENSE/DATA indicator is off, the contents of the Console Data register are displayed. The Console Data register contains either: 1) the most recent contents of the A, X, I or P register or Memory as requested by the Register Select switches; 2) the last Processor output to the Console Data register; or 3) the last keyboard entry to the Console Data register. When the SENSE/DATA indicator is on, the contents of the 4-bit Console Sense register are displayed on the Register Display Indicators. The Console Sense register contains either the last keyboard entry to the Sense register or the last Processor output via the Status Output Command. The upper 12 Register Display indicators are turned off when displaying the Console Sense Register.

Table 2-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
Register Select Switches and Indicators (A, X, I, P and M)	The five Register Select switches determine which one of four Processor registers or memory is to be involved in a read/write operation. Each switch has a corresponding indicator which turns on when a given switch is pressed. The indicators are interlocked such that only one indicator is on at a time. The A, X, I and P switches cause a transfer to occur between the target register and the Console Data register. The M switch causes a transfer between the memory and Console Data register to occur and also causes the P counter to increment after the transfer. This feature permits manual scanning or loading of sequential memory locations by repeated pressing of the M switch.
READ/WRITE Switch and Indicator	The READ/WRITE switch is used in conjunction with the Register Select switches. When the READ/WRITE indicator is on, the contents of the Console Data register will be written into the target register or memory when the appropriate Register Select switch is pressed. When the READ/WRITE indicator is off, the contents of the selected register or memory are copied into the Console Data register and displayed.
Hexadecimal Entry Keyboard (0 thru F)	The Hexadecimal Entry Keyboard consists of 16 switches which are used to enter data into either the 16-bit Console Data register or the 4-bit Console Sense register as determined by the SENSE/DATA switch and indicator. When the SENSE/DATA indicator is off, each depression of a key causes a corresponding 4-bit binary hex code to be entered into the four least-significant bits (LSB's) of the Console Data register with the previously entered data shifted four places to the left. The Console Data register will be statically displayed as long as the SENSE/DATA indicator is off and the computer program does not alter the contents of the Console Data register. When the SENSE/DATA indicator is turned on, each depression of a hex entry key causes the corresponding binary hex code to be entered into the four-bit Console Sense register. The Console Sense register is statically displayed in the four least significant Register Display indicators so long as SENSE/DATA is in the on state and the computer program does not modify the contents of the Console Sense register. The upper 12 Register Display indicators are extinguished.

Table 2-1. Console Switches and Indicators (Cont'd)

SWITCH OR INDICATOR	PURPOSE
SENSE/DATA Switch and Indicator	The SENSE/DATA switch toggles the SENSE/DATA indicator which determines whether the Console Data register or the Console Sense register is to be connected to the hex entry keyboard and the Register Display indicators. If the SENSE/DATA indicator is off, the hex entry keyboard is used to enter data into the Console Data register and the Register Display indicators are connected to the Console Data register. If the SENSE/DATA indicator is on, the keyboard and display are connected to the Console Sense register.
CLEAR Switch	The CLEAR switch, when pressed, clears data from the Console Data register. The switch does not affect the Console Sense register.

2.3 MACHINE MODES

There are four machine modes which are controlled from the Console. These modes are:

1. Stop Mode
2. Step Mode
3. Run Enable Mode
4. Run Mode

Mode selection is made by use of the RUN and STOP switches. The RUN and STOP indicators define the current machine mode as follows:

STOP	RUN	MODE
on	off	Stop
on	on	Step
off	off	Run Enable
off	on	Run

2.3.1 Stop Mode

The Stop Mode unconditionally halts program execution and enables the Entry and Display section of the Console. The Stop Mode is manually entered from either the Run Mode or the Run Enable Mode when the STOP switch is pressed. While in the Stop Mode, the Entry and Display section of the Console is enabled.

2.3.2 Step Mode

The Step Mode is a transient condition in which a single instruction is executed. The Stop Mode is re-entered upon completion of the instruction. A single instruction is executed each time the RUN switch is pressed while the STOP indicator is on.

2.3.3 Run Enable Mode

The Run Enable Mode is an intermediate mode between the Stop and Run Modes. Either the Run or Stop Mode may be entered from the Run Enable Mode. Conversely, the Run Enable Mode can be entered from the Run Mode by execution of a programmed halt. The Run Enable Mode can be entered from the Stop Mode by turning off the STOP indicator. While in the Run Enable Mode, the Entry and Display Section of the Console is disabled.

2.3.4 Run Mode

The Run Mode can be entered only from the Run Enable Mode. When entered, the Run Mode permits the user's program to execute. The Run Mode can be established manually from the Console; semi-automatically by means of the Autoload Option; or, automatically by means of the Power Fail/Restart Option.

The Run Mode is entered manually from the Run Enable Mode by pressing the Console RUN switch. If the Autoload and Power Fail/Restart options are installed, the Run Mode is entered from the Run Enable Mode when the AUTO switch is pressed. The Power Fail/Restart option automatically establishes the Run Mode upon application of adequate power regardless of Processor or Console status prior to the power failure.

2.4 CONSOLE OPERATION

The ALPHA 16/LSI Console is used for initial start-up, program debug, and troubleshooting. The primary functions executed at the console are register display and register change, and the display and entry of memory data. The following paragraphs discuss detailed procedures for performing these operations.

2.4.1 Console Preparation

There are several common steps that must be performed before any console operations may be attempted. These steps prepare the console and the computer for console operations. The initial steps are:

1. Power On — The main power switch for the computer is at the rear of the power supply module. Place the power switch in the up position (ON). The ON indicator on the Console will light and the chassis blowers will run.

2. Enable Console
   Enable the Console by moving the Console Enable slide switch (located in the recess on the side of the Console) to the enable position. The ENABLE indicator is on when the console is enabled.

3. Press STOP
   The computer may come up in the Run Mode because of a previously loaded program. Pressing STOP causes the computer to leave the Run mode.

NOTE

In some cases the RUN indicator may remain on after the STOP switch is pressed. This condition may exist when the computer is attempting to execute certain I/O instructions. This does not indicate a malfunction of the computer. When this occurs, step 4 of this procedure will correct the condition.

4. Press RESET
   Pressing RESET puts the computer in word mode and initializes the computer and peripheral interfaces. It forces the termination of any incomplete instructions.

2.4.2 Console Data Entry Procedure

The Console Data Entry Procedure is used to store data into selected registers or memory locations from the ALPHA 16/LSI Console. The general procedure is to enter the data into the Console Data register via the hex keyboard and then transfer the data to a target register or memory via the Register Select switches. The detailed procedure is as follows:

1. Ready Console
   Prepare the console and the computer for console operations as described in paragraph 2.4.1.

2. Turn SENSE/DATA Indicator off
   Enables Console Data register entry, display and transfer.

3. Turn WRITE/READ Indicator on
   Enables writing into a selected target register or memory

4. Memory Address → P
   Before writing into memory locations, the memory address where data is to be stored is entered into the Console Data register and the P switch is pressed to transfer the contents of the Console Data register to P. This step is not required to enter data into the A, X, I or P registers only.

5. Data → Target Register or Memory

   The data is entered into the Console Data register. The appropriate register select switch is pressed to transfer the contents of the Console Data register to the target register or memory.

6. Sequential Memory Stores

   The P register is automatically incremented each time the M Register Select switch is pressed. To store data in sequential memory locations, go back to step 5 for each succeeding word. To store data in a new location, go back to step 4.

2.4.3 Console Display Procedure

The Console Display Procedure is used to display the contents of selected registers or memory locations. The general procedure is to transfer the data from a register or memory location to the Console Data register by use of the appropriate Register Select switch. The detailed procedure is as follows:

1. Ready Console

   Prepare the console and the computer for console operations as described in paragraph 2.4.1.

2. Turn SENSE/DATA Indicator off

   Enables Console Data register, entry, display and transfer.

3. Turn WRITE/READ Indicator on

   Enables writing into a selected register or memory location. (Required only prior to displaying memory locations.)

4. Memory Address → P

   The address of the memory location to be displayed is entered into the Console Data register and the P switch is pressed. (Required only prior to displaying memory locations.)

5. Turn WRITE/READ Indicator off

   Enables reading from a selected register or memory location.

6. Target Register or Memory → Console

   When the appropriate Register Select switch is pressed, the contents of the target register or memory are copied into the Console Data register and displayed.

7. Sequential Memory Displays

   The P Counter is incremented each time M is pressed. Therefore, to display data in sequential memory locations, go back to step 6.

2.4.4 Program Execution

Programs to be executed may be entered into memory by a number of different means. Short programs may be entered using the Console Data Entry Procedure described in paragraph 2.4.2. Longer programs may be entered using the Autoload feature or various Loader programs. (Autoload may execute automatically.) Regardless of the means used to get a program into memory, the method used to execute that program is generally the same. The Program Counter (P register) must be set to the starting address of the program, and the computer Run mode must be entered. The following steps are used to start program execution from the Console:

1. Ready Console — Prepare the console and the computer for console operations as described in paragraph 2.4.1.

2. Start Address → P — Enter the starting address of the program to be executed in the P register.

NOTE

Enter any required starting information associated with the program in the A, X or SENSE register as appropriate.

3. Press STOP — This enables Run mode, but does not cause the computer to enter Run mode.

4. Press RUN — Pressing the RUN switch causes the computer to enter the Run mode. The computer will continue to run until it executes a Halt instruction, or until the STOP switch is pressed.

2.5 UNATTENDED OPERATION

If for any reason the computer is left unattended when executing a program, it is recommended that the Console be disabled by placing the Console Enable switch to the Disable position.

Section 3
INSTRUCTIONS AND DIRECTIVES

3.1 INTRODUCTION

This section deals with the various instructions and directives recognized by the assembler. The Beta Assembler translates programs which are written in a symbolic language (mnemonics, etc.) into an object language (machine code - see Appendices C and D) which may be loaded into the ALPHA LSI computer. Outputs from the assembler consist of the program object code (normally a punched paper tape) and the program assembly listing. The Beta Assembler is a two-pass assembler. A symbol table for the program is compiled on the first pass and the program object code and assembly listing are produced on the second pass.

3.1.1 Instruction and Directive Classes

The instruction and directive classes listed below in Figure 3-1 are discussed in this section:

CLASS 1	SINGLE-WORD MEMORY REFERENCE INSTRUCTIONS
CLASS 2	DOUBLE-WORD MEMORY REFERENCE INSTRUCTIONS
CLASS 3	BYTE IMMEDIATE INSTRUCTIONS
CLASS 4	CONDITIONAL JUMP INSTRUCTIONS
CLASS 5	SHIFT INSTRUCTIONS
CLASS 6	REGISTER CHANGE INSTRUCTIONS
CLASS 7	CONTROL INSTRUCTIONS
CLASS 8	INPUT/OUTPUT INSTRUCTIONS
CLASS 9	ASSEMBLER CONTROL DIRECTIVES
CLASS 10	DATA AND SYMBOL DEFINITION DIRECTIVES
CLASS 11	PROGRAM LINKAGE DIRECTIVES
CLASS 12	SUBROUTINE DEFINITION DIRECTIVES
CLASS 13	LISTING FORMAT AND ASSEMBLER INPUT DIRECTIVES
CLASS 14	USER DEFINED OPERATION CODE DIRECTIVE

Figure 3-1. Instruction and Directive Classes

3.1.2 Symbolic Notation

The symbolic source code input to the Beta Assembler consists of individual symbolic statements. All of the statements taken together make up a program which is to be translated.

All instructions and certain directives generate an object code. Other directives serve only to control the assembly process.

A source statement represents either an instruction or a directive. It contains four fields - the Label field, the Operation Code field, the Operand field and the Comments field. Adjacent fields are separated by one or more spaces, which allows free-form symbolic input to the assembler. A space in the first character position of a source statement indicates no label present. The listing output from the assembler is formatted for ease in reading, with the Operation Code, Operand and the Comments fields beginning at fixed positions on the listing. Source statements on paper tape are terminated with a carriage return. Line feeds and "rubouts" are ignored. All source statements are limited to 72 characters.

3.1.3 Assembler Source Statement Fields

The following paragraphs discuss the four assembler source statement fields. The relative positions of the fields are shown below in Figure 3-2.

LABEL FIELD	OP CODE FIELD	OPERAND FIELD	COMMENTS FIELD

Figure 3-2. Source Statement Format.

3.1.3.1 Label Field

The label field may contain a name which can be referenced by other instruction statements. It is identified by an alphabetic (A-Z) character in the first position of the source statement. This first character may be followed by as many as five alphanumeric (A-Z, 0-9) or colon (:) characters. This field is terminated by one or more spaces.

At assembly time, the label is assigned the current value and relocation attribute of the program location counter. The same name should not appear in the label field of more than one source statement in a given program (except when used with the SET directive).

3.1.3.2 Op Code Field

The Op Code field contains a legally defined symbolic instruction or directive. In addition, user-defined operation codes may appear in this field. The Operation Code field consists of not less than one nor more than four characters, and is terminated by one or more spaces. The Op Code field of a source instruction statement must be present.

3.1.3.3 Operand Field

The various instructions and directives may or may not require operands. In any case, the syntax of the operand field depends on the type of instruction or directive with which it is associated. The Operand field syntax description is contained in the discussions of the instructions and directives. If the Operand field is present, it contains an expression consisting of one of the following:

1. The currency symbol ($), representing the current program location.
2. A single symbolic term.
3. A single numeric term.
4. A combination of symbolic terms, numeric terms and/or the currency symbol joined by the arithmetic operators plus (+) or minus (-).
5. A text string.
6. A literal (=xx).

The value assigned the currency symbol by the assembler is the value of the program location counter at the time the currency symbol is encountered. The value is absolute if an absolute assembly is being performed and relative if a relocatable assembly is being performed. The currency symbol allows the programmer to reference memory locations relative to the instruction being written rather than assigning labels to the referenced location, thus saving space in the symbol table and reducing the probability of multiple label definitions.

Symbolic terms (names) may be absolute or relative, depending on the assembly mode under which they have been defined.

Numeric terms are always absolute. They consist of decimal, octal and hexadecimal numbers. Decimal numbers can be any value in the range -32768 through +32767. The first digit of the number must be non-zero. Octal numbers can be any octal value in the range 0 through 0177777. The first - or leading - digit of the number must be zero to specify octal numbers. Hexadecimal numbers can be any hexadecimal value in the range :0 through :FFFF. The number must be preceded by a colon (:). Although octal and hexadecimal numbers may be signed, they are normally used to generate a bit pattern or reference a particular memory location rather than to generate a signed numeric value.

Combinations of terms (including the currency symbol) can be achieved by using the arithmetic operators plus (+) and minus (-). The value of the final expression must be in the range :0 thru :FFFF. Combinations of relative and absolute terms are governed by additional restrictions (see Sec. 3.1.5).

Text strings consist of any sequence of characters surrounded by single quotes ('). Inclusion of a single quote within the character string is accomplished using two adjacent single quotes. The object code generated consists of 8-bit ASCII character codes, packed two characters per word, or one 8-bit ASCII character in the LSB byte of an instruction (e.g., the operands of Immediate instructions). When a DATA directive is used, the text string may consist of one or two characters. When one character is specified, the 8-bit code appears in the LSB byte of the computer word, with the MSB byte set to zero.

If two characters are specified, the code for the first character is put in the MSB byte of the computer word and the code for the second character is put in the LSB byte of the computer word. When the TEXT directive is used, the text string may consist of as many as 57 characters. The characters are packed two per word, with the code for the first character appearing in the MSB byte of the computer word and the code for the second character appearing in the LSB byte of the computer word. Trailing character positions are filled with blanks (:A0) - e.g., TEXT 'A' would generate a value of :C1A0 for the specified computer word.

Literals are designated by preceding the expression in the operand with an equals (=) sign. This affects the entire expression - not just one term in the expression. When a literal is encountered by the assembler, a word is reserved in the scratchpad area of memory to hold the computed value of the expression in the operand field. Memory addressing is then generated to access the scratchpad location.

3.1.3.4 Comments Field

The comments field follows the operand field or, for those instructions which do not require operands, the op code/instruction field. This field generally contains programmer's notes, cryptic messages, helpful hints and that sort of thing. Needless to say, comments appear on the assembly listing, but do not generate object code.

3.1.4 Arithmetic Operations and Overflow

The ALPHA LSI computer performs two's-complement arithmetic. All additions and subtractions are performed on full 16-bit values. Thus, addition operations involving byte values place the 8-bit data in the least significant 8 bits of the adder and set the most significant 8 bits to zero (e.g., AXI :50 would add :0050 to the 16-bit X register). Subtraction operations involving byte values similarly obtain a 16-bit two's complement of the data (e.g., SXI :50 would add :FFB0 to the 16-bit X register).

Arithmetic overflow occurs when the result of an arithmetic operation exceeds the range -32768 through +32767. Specifically, this involves the carry from bit 14 to bit 15 of the adder, and the carry out of bit 15. If the carries are not equal (0 and 1, or 1 and 0), an arithmetic overflow has occurred and the OV (overflow) indicator is set. The operation is described below in Figure 3-3.

Carry In and Carry Out

```
              S
             1 1  111  1111  1111  11←carries
      -5 =   1  111  1111  1111  1011
   +(-5) =   1  111  1111  1111  1011
Sum = -10    1  111  1111  1111  0110
```

No Carry In and No Carry Out

```
             S
                                  1 1←carries
      +5 = 0  000  0000  0000  0101
   +5(+5) = 0  000  0000  0000  0101
Sum = +10 = 0  000  0000  0000  1010
```

Carry In and No Carry Out

```
            S
             1  111  1111  111  111
+32767      0   111  1111  1111 1111
+   (+1)    0   000  0000  0000 0001
Sum=32768   1   000  0000  0000 0000
```

Carry Out and No Carry In

```
              S
             1                       ←— carries
-32768       1  000  0000  0000  0000
+   (-1)     1  111  1111  1111  1111
Sum=-32769   0  111  1111  1111  1111
```

Figure 3-3. Arithmetic Overflow.

3.1.5 Relocatability

Relative and absolute programming modes are controlled by the REL and ABS directives. The default condition of the assembler is the relative mode (REL). The programmer should note that the ORG directive modifies the contents, but not the relocation attribute, of the program location counter.

An absolute program (or section of coding) can only be loaded and executed in the memory locations specified by the user at assembly time, whereas a relative (or relocatable) program may be loaded and executed in any area of memory specified by the user at load time. Out-of-range memory references are resolved through the use of the scratchpad area in the base page (the first 256 words of memory). The user should refer to the LAMBDA Object Loader documentation.

Multiple-term expressions are reduced by the assembler to a single expression which may be relocatable or absolute, according to the following rule:

R = (Number of added relocatable terms) - (Number of subtracted relocatable terms)

If R = 1, the expression is relocatable, if R = 0, the expression is absolute, and if R is not equal to 0 or 1, the expression is illegal. Relocatable expressions are modified by the load bias entered in the A register when the LAMBDA Object Loader is executed:

Relocated Expression value = Assembled Expression value + Load Bias

In addition, the location of the entire program (or block of coding) is offset by the same load bias:

Relocated program location = Assembled program location + Load Bias.

3.2 MEMORY REFERENCE INSTRUCTIONS

3.2.1 Word Mode Operations and Instruction Format

Word mode memory reference operations access full 16-bit memory operands. The default mode of the computer is the Word Mode - i.e., when no mode control instruction has been executed, the computer is in the Word Mode. SWM is the mode control instruction which places the computer in the Word Mode. In addition, the SIN, SIA and SIX instructions force the computer into the Word Mode. The SIN instruction forces the Word Mode for the number of succeeding instructions specified by its associated operand. The SIA and SIX instructions unconditionally force the Word Mode. The format for the Word Mode memory reference instructions is shown below in Figure 3-4 .

```
LABEL     OP CODE                 [* |@| *@]EXPRESSION      [COMMENTS]

          No Operator = Direct Address
           * = Indirect Addressing  (multi-level)
           @ = Indexed Addressing
          *@ = Indirect Post-indexed Addressing (multi-level)
```

Figure 3-4. Word Mode Memory Reference Instruction Format

All (16-bit) word address pointers (defined by DATA statements) consist of fifteen bits of address in the least significant 15 bits. The most significant bit (bit 15) specifies indirect (=1) or direct (=0) addressing.

3.2.1.1 Word Mode Direct Addressing

Word Mode direct addressing allows any memory reference instruction to access the first 256 words of memory (the base page/scratchpad area) as well as 512 memory locations about the instruction itself (relative to P). Relative to P forward addressing includes 256 words forward (toward higher memory) of the instruction and relative to P backwards addressing includes the instruction itself and 255 memory locations backward from the instruction. When direct addressing is desired, the expression in the operand field should not be preceded by an * or @ character. When the assembler encounters a direct reference to an out of range memory location, it automatically generates an address pointer in the scratchpad area and references the associated memory location indirectly through the pointer.

3.2.1.2 Word Mode Indirect Addressing

Word Mode indirect addressing allows any memory reference instruction to access any memory location through an address pointer in the first 256 words of memory (the base page/scratchpad area) or an address pointer in the 511 memory locations about the instruction itself (relative to P). Relative to P forward indirect addressing allows the address pointer to reside in memory locations as many as 256 words forward (toward higher memory) of the instruction and relative to P backwards indirect addressing allows the address pointer to be in any memory location 255 words or less prior to the instruction. When indirect addressing is desired, the expression in the operand field should be preceded by an asterisk (*). Multi-level indirect addressing is accomplished by accessing address pointers in which the most significant bit (bit 15) is set. The memory operand is not accessed until an address pointer with the most significant bit reset (=0) is encountered. Indirect address pointers can be defined by the programmer through the use of the DATA directive by preceding the expression in the operand field with an asterisk (*).

3.2.1.3 Word Mode Direct Indexed Addressing

Word Mode direct indexed addressing allows any memory reference instruction to access memory locations by summing the contents of the X register and any offset value in the range 0 through 255. The offset value is defined by the expression in the operand field. When direct indexed addressing is desired, the expression in the operand field should be preceded by an @ symbol. When the assembler encounters an expression with a value greater than 255 in the operand field of a direct indexed memory reference instruction, it automatically generates an address pointer in the scratchpad area and references the associated memory location indirect post-indexed through the pointer.

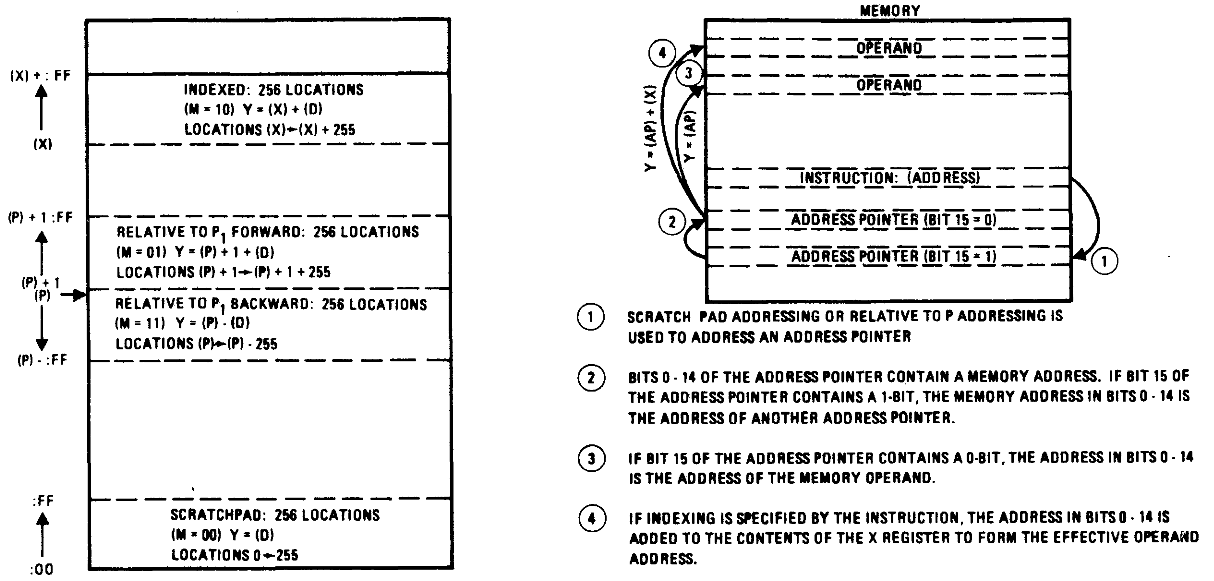
3.2.1.4 Word Mode Indirect Post-Indexed Addressing

Word Mode indirect post-indexed addressing allows any memory reference instruction to access memory locations by summing the contents of the X register and the contents of an address pointer in the base page/scratchpad area (the first 256 memory locations). If the most significant bit of the address pointer is set, it contains the address of another address pointer, which in turn may contain the address of another pointer, and so forth. When an address pointer with the most significant bit (bit 15) set to zero is found, the contents of the X register are added to it to form the effective memory address. The memory operand is then accessed. When indirect post-indexed addressing is desired, the expression in the operand field should be preceded by an asterisk (*) and an @ symbol.

Because the Scan Memory (SCM) instruction always uses indirect post-indexed addressing, the assembler automatically generates the necessary machine code and does not allow @ or * operators on the associated operand expression. The operand expression for this instruction should reference a user-defined address pointer in the base page.

3.2.1.5 Word Mode Summary

A summary of Word Mode addressing, including the associated machine code, is shown in Figure 3-5.



(1) SCRATCH PAD ADDRESSING OR RELATIVE TO P ADDRESSING IS USED TO ADDRESS AN ADDRESS POINTER

(2) BITS 0 - 14 OF THE ADDRESS POINTER CONTAIN A MEMORY ADDRESS. IF BIT 15 OF THE ADDRESS POINTER CONTAINS A 1-BIT, THE MEMORY ADDRESS IN BITS 0 - 14 IS THE ADDRESS OF ANOTHER ADDRESS POINTER.

(3) IF BIT 15 OF THE ADDRESS POINTER CONTAINS A 0-BIT, THE ADDRESS IN BITS 0 - 14 IS THE ADDRESS OF THE MEMORY OPERAND.

(4) IF INDEXING IS SPECIFIED BY THE INSTRUCTION, THE ADDRESS IN BITS 0 - 14 IS ADDED TO THE CONTENTS OF THE X REGISTER TO FORM THE EFFECTIVE OPERAND ADDRESS.

Direct Addressing

Indirect Addressing

Figure 3-5. Word Mode Addressing Summary

3.2.2 Byte Mode Operations and Instruction Format

Byte Mode memory reference operations access 8-bit byte operands. The Byte Mode is established by execution of the Set Byte Mode (SBM) instruction. Byte Mode memory reference operation is inhibited (the computer is forced into the Word Mode) by execution of the SIN, SWM, SIA and SIX instructions. The SIN instruction inhibits Byte Mode operations for the number of succeeding instructions specified by its associated operand. The SWM, SIA and SIX instructions unconditionally force the computer into the Word Mode. The format for the Byte Mode memory reference instructions is shown below in Figure 3-6.

```
[LABEL]        OP CODE          [* |@| *@] EXPRESSION          [COMMENTS]

          No Operator = Direct Address
            * = Indirect Addressing (One Level)
            @ = Indexed Addressing
           *@ = Indirect Post-Indexed Addressing (One Level)
```

Figure 3-6. Byte Mode Memory Reference Instruction Format

All (16-bit) byte address pointers (BAC directives) consist of fifteen bits of word address in the most significant 15 bits. The least significant bit (bit 0) specifies the most significant 8 bits (=0) or the least significant 8 bits (=1) of the word. Only one level of byte memory reference indirect addressing - specified in the instruction itself - is possible. Byte operands affecting the register are always right-justified - i.e., bytes cannot be loaded into, added to or stored from the most significant 8 bits of the A and X registers.

The IMS, MPY, DVD, NRM, JMP and JST instructions are not affected by the Byte Mode. They always use full 16-bit word operands.

3.2.2.1 Byte Mode Direct Addressing

Byte Mode direct addressing allows any byte memory reference instruction to access the first 256 bytes (128 words) of memory as well as 512 byte locations ahead (toward high-order memory) of the instruction itself. When direct addressing is desired, the expression in the operand field should not be preceded by an * or @ character. When the assembler encounters a direct reference to an out of range byte location, it automatically generates a byte address pointer in the scratchpad area and references the associated byte location indirectly through the pointer.

3.2.2.2 Byte Mode Indirect Addressing

Byte Mode indirect addressing allows any byte memory reference instruction to access any byte location through a byte address pointer in the first 256 words of memory (the base page/scratchpad area) or a byte address pointer in the 511 memory locations about the instruction itself (relative to P). Relative to P forward indirect addressing allows the byte address pointer to reside in memory locations as many as 256 words forward (toward higher memory) of the instruction and relative to P backwards indirect addressing allows the byte address pointer to be in any memory location 255 words or less prior to the instruction. When indirect addressing is desired, the expression in the operand field should be preceded by an asterisk (*). Byte address pointers to be used by indirect byte memory reference instructions can be defined by the programmer through the use of the BAC directive. Since a byte address pointer utilizes all 16 bits to specify a given byte location, indirect byte addressing is limited to one level.

3.2.2.3 Byte Mode Direct Indexed Addressing

Byte Mode direct indexed addressing allows any byte memory reference instruction to access byte locations by summing the contents of the X register and any base value in the range 0 through 255. The base value is defined by the expression in the operand field. When direct indexed addressing is desired, the expression in the operand field should be preceded by an @ symbol. When the assembler encounters an expression with a value greater than 255 in the operand field of a direct indexed byte memory reference instruction, it automatically generates a byte address pointer in the scratchpad area and references the associated byte memory location indirect post-indexed through the byte address pointer.

3.2.2.4 Byte Mode Indirect Post-Indexed Addressing

Byte Mode indirect post-indexed addressing allows any byte memory reference instruction to access byte locations by summing the contents of the X register and the contents of a byte address pointer in the base page/scratchpad area (the first 256 memory locations). When indirect post-indexed byte addressing is desired, the expression in the operand field should be preceded by an asterisk (*) and an @ symbol.

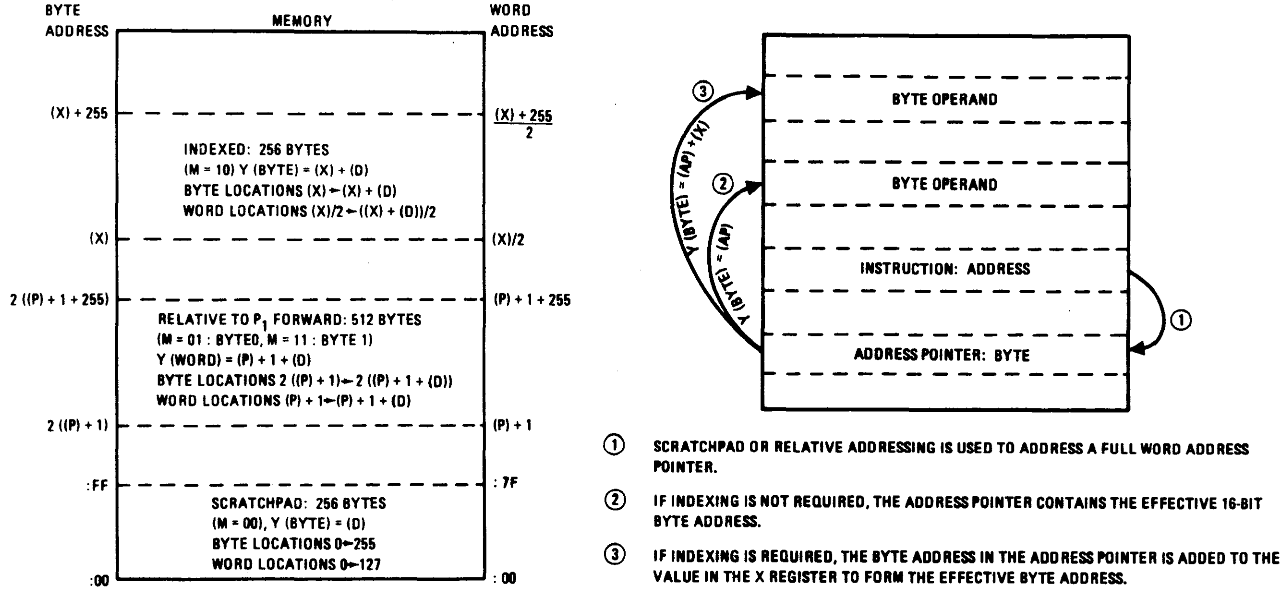
Because the Scan Memory (SCM) instruction always uses indirect post-indexed addressing, the assembler automatically generates the necessary machine code and does not allow @ or * operators on the associated operand expression. When performing byte scans, the operand expression for this instruction should reference a user-defined byte address pointer in the base page.

3.2.2.5 Byte Mode Summary

A summary of Byte Mode addressing, including the associated machine code, is shown in Figure 3-7.

3.2.3 Arithmetic Memory Reference Instructions

ADD	ADD TO A. Adds the contents of the effective memory location to the A register. OV is set if arithmetic overflow occurs.
ADDB	ADD BYTE TO A. Adds the contents of the effective byte location to the A register. OV is set if arithmetic overflow occurs.
SUB	SUBTRACT FROM A. Subtracts the contents of the effective memory location from the A register. OV is set if arithmetic overflow occurs.
SUBB	SUBTRACT BYTE FROM A. Subtracts the contents of the effective byte location from the A register. OV is set if arithmetic overflow occurs.



① SCRATCHPAD OR RELATIVE ADDRESSING IS USED TO ADDRESS A FULL WORD ADDRESS POINTER.

② IF INDEXING IS NOT REQUIRED, THE ADDRESS POINTER CONTAINS THE EFFECTIVE 16-BIT BYTE ADDRESS.

③ IF INDEXING IS REQUIRED, THE BYTE ADDRESS IN THE ADDRESS POINTER IS ADDED TO THE VALUE IN THE X REGISTER TO FORM THE EFFECTIVE BYTE ADDRESS.

Direct Addressing

Indirect Addressing

Figure 3-7. Byte Mode Addressing Summary

3.2.4 Logical Memory Reference Instructions

AND	AND TO A. Logically AND's the contents of the effective memory location with the A register.
ANDB	AND BYTE TO A. Logically AND's the contents of the effective byte location with the A register. Resets the most significant 8 bits of the A register to zero.
IOR	INCLUSIVE OR TO A. Inclusively OR's the contents of the effective memory location with the A register.
IORB	INCLUSIVE OR BYTE TO A. Inclusively OR's the contents of the effective byte location with the A register. The most significant 8 bits of the A register are unchanged.
XOR	EXCLUSIVE OR TO A. Exclusively OR's the contents of the effective memory location with the A register.
XORB	EXCLUSIVE OR BYTE TO A. Exclusively OR's the contents of the effective byte location with the A register. The most significant 8 bits of the A register are unchanged.

3.2.5 Data Transfer Memory Reference Instructions

EMA EXCHANGE MEMORY AND A. Simultaneously stores the A register in the effective memory location and loads the contents of the effective memory location into the A register.

EMAB EXCHANGE MEMORY BYTE AND A. Simultaneously stores the least significant 8 bits of the A register into the effective byte location and loads the contents of the effective byte location into the least significant 8 bits of the A register. Resets the most significant 8 bits of the A register to zero.

LDA LOAD A. Loads the contents of the effective memory location into the A register.

LDAB LOAD A BYTE. Loads the contents of the effective byte location into the least significant 8 bits of the A register. Resets the most significant 8 bits of the A register to zero.

LDX LOAD X. Loads the contents of the effective memory location into the X register.

LDXB LOAD X BYTE. Loads the contents of the effective byte location into the least significant 8 bits of the X register. Resets the most signigicant 8 bits of X register to zero.

STA STORE A. Stores the A register in the effective memory location.

STAB STORE A BYTE. Stores the least significant 8 bits of the A register in the effective byte location.

STX STORE X. Stores the X register in the effective memory location.

STXB STORE X BYTE. Stores the least significant 8 bits of the X register in the effective byte location.

3.2.6 Program Transfer Memory Reference Instructions

CMS COMPARE AND SKIP IF HIGH OR EQUAL. Compares the contents of the effective memory location with the A register. If the A register is greater than the contents of the effective memory location, a one-word skip occurs. If the A register is equal to the contents of the effective memory location, a two-word skip occurs.

CMSB COMPARE BYTE AND SKIP IF HIGH OR EQUAL. Compares the contents of the effective byte location with the A register. If the A register is greater than the contents of the effective byte location, a one-word skip occurs. If the A register is equal to the contents of the effective byte location, a two-word skip occurs. All 16 bits of the A register are compared to the contents of the effective byte location, so the most significant 8 bits of A register should be zeros.

IMS INCREMENT MEMORY AND SKIP ON ZERO RESULT. Contents of the effective memory location are incremented by one. If the increment causes the result to become zero, a one-word skip occurs. OV is set if arithmetic overflow occurs.

NOTE

IMS is often used as an interrupt instruction (See Sec. 4.3).

JMP JUMP UNCONDITIONAL. The P counter is loaded with the value of the effective memory location, causing an unconditional branch to that address.

JST JUMP AND STORE. The contents of the P counter are stored in the effective memory location and the P counter is then loaded with the effective memory location plus one.

NOTE

JST is often used as an interrupt instruction (See Sec. 4.3).

SCM SCAN MEMORY. Compares the A register with area of memory defined by the Address Pointer in Scratchpad (Base Address of Buffer - 1) and the contents of the X register (Buffer Length). If a match is found, the Scan is terminated and the next instruction in sequence is executed. The X register is decremented once for each word scanned. When a match is found, the X register contains the number of words remaining to be scanned. The remainder of the buffer can be scanned simply by executing the SCN instruction again. If a match is not found by the time X reaches zero, a one place skip occurs and the instruction terminates. This instruction operates under both Word and Byte modes, the Address Pointer in Scratchpad should be constructed accordingly (DATA or BAC, respectively - See Sec. 3.10).

NOTE

This instruction is interruptable. Upon completion of interrupt processing, SCM resumes operation at the point where the interrupt occured.

3.3 DOUBLE-WORD MEMORY REFERENCE INSTRUCTIONS

3.3.1 Format

The Double-Word Memory Reference instructions require two consecutive words of memory and allow direct and indirect addressing. Indexed addressing is not allowed and is, in fact, not useful, since these instructions manipulate both the A and X registers. The format for Double-Word Memory Reference instructions is shown in Figure 3-8.

[LABEL] OP-CODE [*] EXPRESSION 1[,EXPRESSION 2] [COMMENTS]

No Operator = Direct Address
* = Indirect Addressing (multi-level)
EXPRESSION 1: any absolute or relative expression defining the effective memory location.
EXPRESSION 2: an optional instruction count in the range 1 through 16 for DVD and MPY and 1 through 31 for NRM.

Figure 3-8. Double Word Memory Reference Format

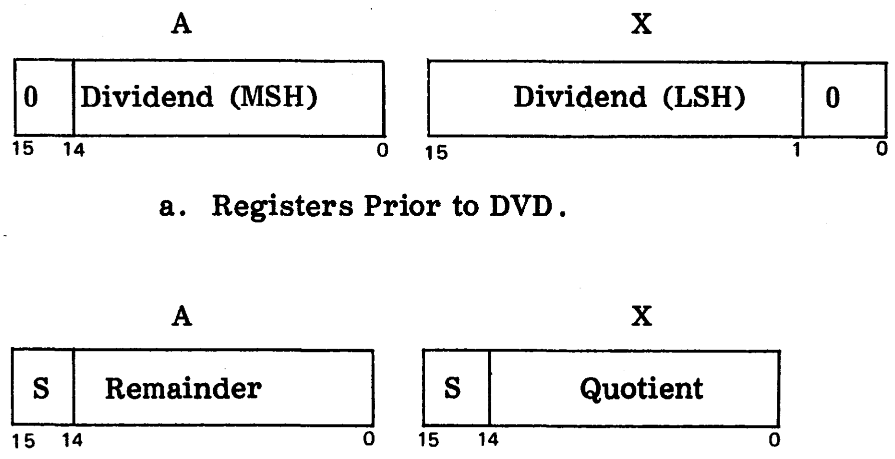
3.3.2 Instructions

DVD DIVIDE. Divides the contents of the A and X registers by the contents of the memory location addressed by the second word of the instruction (Expression 1). This address pointer may be direct or indirect.

Prior to execution of the instruction, the A and X registers contain the signed 30 bit dividend (as shown in Figure 3-9), and the addressed memory location contains the signed full-word divisor. Both dividend and divisor must be positive.

The quotient is returned in the X register (sign plus 15 bits) and the remainder in the A register (sign plus 15 bits). OV is set if a divide fault occurs (divisor ≤ Most Significant Half (MSH) of the dividend), or else is returned in the same state as entered.

A full divide is performed if no instruction count (Expression 2) is specified. Partial divides are executed according to the specified instruction count.



a. Registers Prior to DVD.

b. Registers after DVD

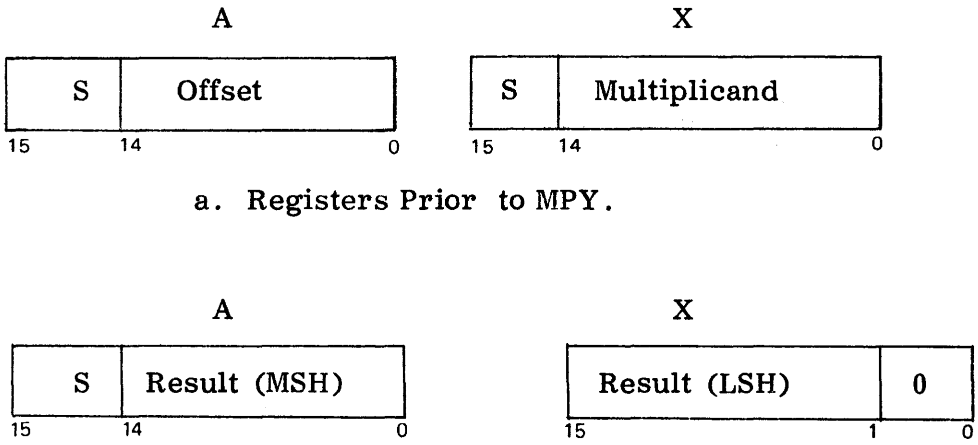
Figure 3-9. Divide

MPY MULTIPLY AND ADD. Multiplies the contents of the X register by the contents of the memory location addressed by the second word of the instruction (Expression 2) and adds the contents of the A register to the product. The address pointer may be direct or indirect.

Prior to execution of the MPY instruction, the X register contains the signed full-word multiplicand, the addressed memory location contains the signed full-word multiplier, and the A-register contains the signed "offset" to be added. (Refer to Figure 3-10.) Multiplicand and offset must all be positive or zero.

The result is returned in the A and X registers (sign plus 30 bits). A full multiply and add is performed if no instruction count (Expression 2) is specified. Partial multiplication is executed according to the specified instruction count.

In all cases OV will be reset (=0) at completion of a full multiply.



a. Registers Prior to MPY.

b. Registers After MPY.

Figure 3-10. Multiply and Add

NRM NORMALIZE A AND X. Contents of A and X registers are arithmetically shifted left (see Figure 3-11) until A15 is not equal to A14 or until the maximum shift count specified (Expression 2) is reached. The exponent, addressed by the second word of the instruction (Expression 1), is decremented once for each shift. The address of the exponent may be direct or indirect.

The NRM instruction treats the A and X register as a combined 31-bit plus sign register.

OV is reset (=0) if normalization occurs, otherwise it is returned unaltered In either case, the exponent will be decremented once for each shift performed.

A full 31-bit normalize is performed if no instruction count (Expression 2) is specified. Otherwise, the specified count will determine the maximum shifts performed.

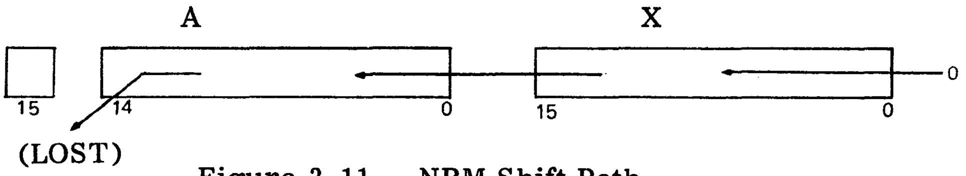


Figure 3-11. NRM Shift Path

3.4 IMMEDIATE INSTRUCTIONS

3.4.1 Format

Immediate instructions are similar to Memory Reference instructions in that they perform logical and arithmetic operations involving memory data and operating registers. The memory data, however, is stored within the immediate instruction itself rather than in a separate operand word or byte. The operands of the instructions may be any absolute expression which is within the range 0 - :FF (i.e., any absolute expression which fits into eight bits). The Immediate Instruction format is shown in Figure 3-12.

```
[LABEL]     OP-CODE              EXPRESSION              [COMMENTS]

        EXPRESSION:  must be absolute and in the range : 0 through : FF
```

Figure 3-12. Immediate Instruction Format

3.4.2 Instructions

AXI ADD TO X IMMEDIATE. The operand is added to the contents of the X register. OV is set if arithmetic overflow occurs.

CAI COMPARE TO A IMMEDIATE. The operand is compared to the least significant 8 bits of the A register. If unequal, a skip of one place occurs. If equal, the next instruction in sequence is executed. The contents of A are not disturbed. The most significant 8 bits of A do not take part in the comparison.

CXI COMPARE TO X IMMEDIATE. The operand is compared to the least significant 8 bits of the X register. If unequal, a skip of one place occurs. If equal, the next instruction in sequence is executed. The contents of X are not disturbed. The most significant 8 bits of X do not take part in the comparison.

LAM LOAD A MINUS IMMEDIATE. The operand is negated (two's complemented) and loaded as a 16-bit word into the A register.

LAP LOAD A POSITIVE IMMEDIATE. The operand is loaded into the least significant 8 bits of the A register. The most significant 8 bits of A are set to zero.

LXM LOAD X MINUS IMMEDIATE. The operand is negated (two's complemented) and loaded as a 16-bit word into the X register.

LXP LOAD X POSITIVE IMMEDIATE. The operand is loaded into the least significant 8 bits of the X register. The most significant 8 bits of X are set to zero.

SXI SUBTRACT FROM X IMMEDIATE. The operand is negated (two's complemented) and added as a 16-bit word to the X register. OV is set if arithmetic overflow occurs.

3.5 CONDITIONAL JUMP INSTRUCTIONS

3.5.1 Format

Conditional Jump instructions test conditions within the computer and perform program branches depending on the results of the test. A Jump occurs if the specified conditions are satisfied. All branches are direct and relative to the P register (location of the Conditional Jump instruction). The range of Conditional Jumps is:

Forward Jumps:	P+1 through P+64
Backward Jumps:	P through P-63

3.5.2 Microcoding

A general code, JOC, for Jump On Condition, is provided so the programmer can microcode jump conditions. There are five different conditions which may be tested either individually or in combination:

1. Sign of A (positive or negative)
2. Contents of A (zero or not zero)
3. Contents of X (zero or not zero)
4. Overflow Indicator (Set or Reset)
5. SENSE Indicator (on or off)

The conditions may be tested individually or in combination. Figure 3-13 shows the format for the JOC instruction:

```
[LABEL]   JOC              EXPRESSION 1, EXPRESSION 2      [COMMENTS]

      EXPRESSION 1:  must be absolute and in the range :0 through :3F
      EXPRESSION 2:  must represent a location with -63 through +64
                     computer words.
```

Figure 3-13. JOC Jump On Condition Format

JOC commands consist of two groups. The AND group and the OR group. The AND test group requires that all of the test conditions specified by bits 0 thru 4 of Expression 1 be true for the jump to take place. The OR group requires that any one or more of the test conditions specified be true if the jump is to take place. Expression 1 consists of 6 bits as defined by Figure 3-14. Bit 5 specifies which test group is used. Bits 0 thru 4 specify inclusion of a specific test condition if set equal to 1. If equal to 0, the associated test condition is not examined.

$$\underbrace{\text{JOC :XX, ADR}}_{B_5\ B_4\ B_3\ B_2\ B_1\ B_0}$$

B_5 = 1 for AND test group
B_5 = 0 for OR test group

	AND GROUP	OR GROUP
B_4 = 1	X ≠ 0	X = 0
B_3 = 1	SENSE on	SENSE off
B_2 = 1	OV reset	OV set (resets OV)
B_1 = 1	A ≠ 0	A = 0
B_0 = 1	A positive	A negative

Figure 3-14. JOC Microcode Bit Functions

The following Conditional Jump instructions are special cases of the general JOC instruction. Since they are more often utilized than the other conditional jumps, they have been given their own mnemonics. Figure 3-15 illustrates the general format for the Conditional Jump instructions.

[LABEL]	OP-CODE	EXPRESSION	[COMMENTS]

EXPRESSION: must represent a location within -63 through +64 computer words.

Figure 3-15. Conditional Jump Format

3.5.3 Arithmetic Conditional Jump Instructions

JAG JUMP IF A GREATER THAN ZERO. Jump occurs if the A register is greater than zero.

JAL JUMP IF A LESS THAN ONE. Jump occurs if the A register is less than or equal to zero.

JAM JUMP IF A MINUS. Jump occurs if the A register is less than zero (A_{15}= 1).

JAN JUMP IF A NOT ZERO. Jump occurs if the A register is not zero.

JAP JUMP IF A POSITIVE. Jump occurs if the A register is greater than or equal to zero (A_{15} = 0).

JAZ JUMP IF A ZERO. Jump occurs if the A register is zero.

JXN JUMP IF X NOT ZERO. Jump occurs if the X register is not zero.

JXZ JUMP IF X ZERO. Jump occurs if the X register is zero.

3.5.4 Control Conditional Jump Instructions

JOR JUMP IF OVERFLOW RESET. Jump occurs if the OV equals zero.

JOS JUMP IF OVERFLOW SET. Jump occurs if OV equals one. OV is reset to zero during jump.

JSR JUMP IF SENSE INDICATOR RESET. Jump occurs if the SENSE Indicator is off.

JSS JUMP IF SENSE INDICATOR SET. Jump occurs if the SENSE Indicator is on.

3.6 SHIFT INSTRUCTIONS

3.6.1 Operand Restrictions and Instruction Format

Shift instructions move bit patterns in the computer registers either right or left. Shifts may involve a single register (A or X), a single register and the overflow (OV) register, or both the A and X registers and the OV indicator. The Processor provides logical, arithmetic and rotate shifts. The operands (n) for single register and double register instructions can be any absolute value from 1 through 8 and 16, respectively. The single register shift instruction format is shown in Figure 3-16 and the instruction format for double register (long) shifts is shown in Figure 3-17.

[LABEL] OP-CODE EXPRESSION [COMMENTS]
EXPRESSION: must be absolute and in the range 1 through 8.

Figure 3-16. Single Register Shift Format

[LABEL] OP-CODE EXPRESSION [COMMENTS]
EXPRESSION: must be absolute and in the range 1 through 16.

Figure 3-17. Double Register (Long) Shift Format

3.6.2 Arithmetic Shift Instructions

The shift paths for the arithmetic shift instructions are illustrated below in Figure 3-18 and Figure 3-19.

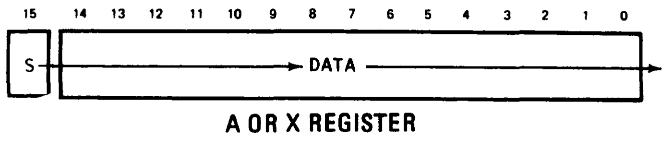


Figure 3-18. Arithmetic Right Shift

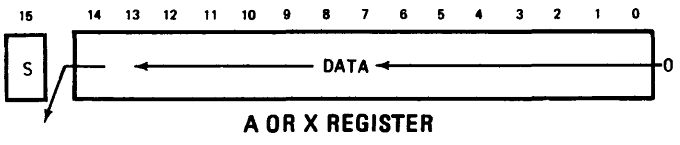


Figure 3-19. Arithmetic Left Shift

ALA ARITHMETIC SHIFT A LEFT. Contents of A Register (bits 0-14) are shifted left n places. The sign bit (bit 15) is unchanged. Zeros are shifted into bit 0 and bits shifted out of bit 14 are lost.

ALX ARITHMETIC SHIFT X LEFT. Contents of X Register (bits 0-14) are shifted left n places. The sign bit (bit 15) is unchanged. Zeros are shifted into bit 0 and bits shifted out of bit 14 are lost.

ARA ARITHMETIC SHIFT A RIGHT. Contents of A Register are shifted right n places. The sign bit (bit 15) is unchanged and is shifted into and propagated through bit 14. Bits shifted out of bit 0 are lost.

ARX ARITHMETIC SHIFT X RIGHT. Contents of X Register are shifted right n places. The sign bit (bit 15) is unchanged and is shifted into and propagated through bit 14. Bits shifted out of bit 0 are lost.

3.6.3 Logical Shift Instructions

The shift paths for the logical shift instructions are illustrated below in Figure 3-20 and Figure 3-21.

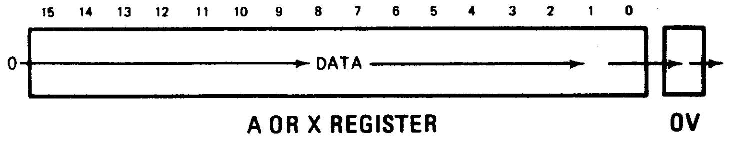


Figure 3-20. Logical Right Shift

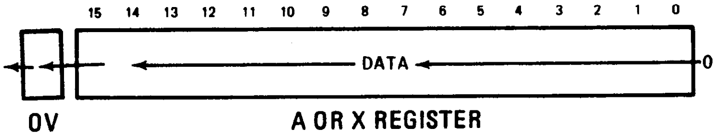


Figure 3-21. Logical Left Shift

LLA LOGICAL SHIFT A LEFT. Contents of A Register are shifted left n places through OV. Zeros are shifted into bit 0. Bits are shifted from bit 15 of A into OV. Bits shifted out of OV are lost. A and OV act as a 17-bit register.

LLX LOGICAL SHIFT X LEFT. Contents of X Register are shifted left n places through OV. Zeros are shifted into bit 0. Bits are shifted from bit 15 of X into OV. Bits shifted out of OV are lost. X and OV act as a 17-bit register.

LRA LOGICAL SHIFT A RIGHT. Contents of A Register are shifted right n places through OV. Zeros are shifted into bit 15. Bits are shifted from bit 0 of A into OV. Bits shifted out of OV are lost. A and OV act as a 17-bit register.

LRX LOGICAL SHIFT X RIGHT. Contents of X Register are shifted right n places through OV. Zeros are shifted into bit 15. Bits are shifted from bit 0 of A into OV. Bits shifted out of OV are lost. X and OV act as a 17-bit register.

3.6.4 Rotate Shift Instructions

The shift paths for the rotate shift instructions are illustrated below in Figure 3-22 and Figure 3-23.

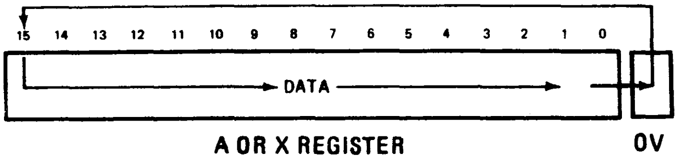


Figure 3-22. Rotate Right

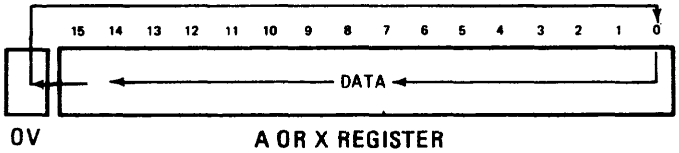


Figure 3-23. Rotate Left

RLA ROTATE A LEFT WITH OVERFLOW. Contents of A Register are shifted left n places through OV. OV is shifted into bit 0 and bit 15 is shifted into OV. No bits are lost when this shift is executed. A and OV act as a 17-bit register.

RLX ROTATE X LEFT WITH OVERFLOW. Contents of X Register are shifted left n places through OV. OV is shifted into bit 0 and bit 15 is shifted into OV. No bits are lost when this shift is executed. X and OV act as a 17-bit register.

RRA ROTATE A RIGHT WITH OVERFLOW. Contents of A Register are shifted right n places through OV. OV is shifted into bit 15 and bit 0 is shifted into OV. No bits are lost when this shift is executed. A and OV act as a 17-bit register.

RRX ROTATE X RIGHT WITH OVERFLOW. Contents of X Register are shifted right n places through OV. OV is shifted into bit 15 and bit 0 is shifted into OV. No bits are lost when this shift is executed. X and OV act as a 17-bit register.

3.6.5 Double Register (Long) Rotate Shift Instructions

The shift paths for the Long Logical Shift instructions are shown below in Figures 3-24 and 3-25.

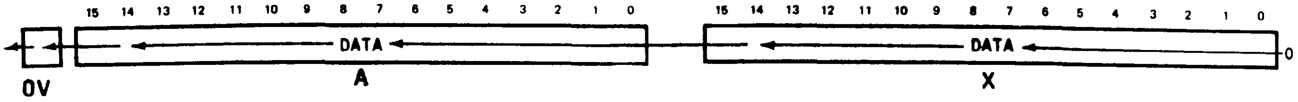


Figure 3-24. Long Left Shift

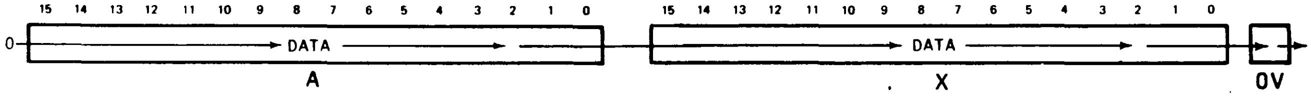


Figure 3-25. Long Right Shift

LLL LONG LOGICAL SHIFT LEFT. Contents of A and X Registers are logically shifted left through OV n places. For each bit position shifted, zero is shifted into X_0 , X_{15} is shifted into A_0 and A_{15} is shifted into OV. The previous contents of OV are lost. A, X and OV act as a 33-bit register.

LLR LONG LOGICAL SHIFT RIGHT. Contents of A and X Registers are logically shifted right through OV n places. For each bit position shifted, zero is shifted in A_{15} , A_0 is shifted into X_{15} and X_0 is shifted into OV. The previous contents of OV are lost. A, X and OV act as a 33-bit register.

3.6.6 Double Register (Long) Shift Instructions

Shift paths for the Long Rotate Shift instructions are shown below in Figures 3-26 and 3-27.

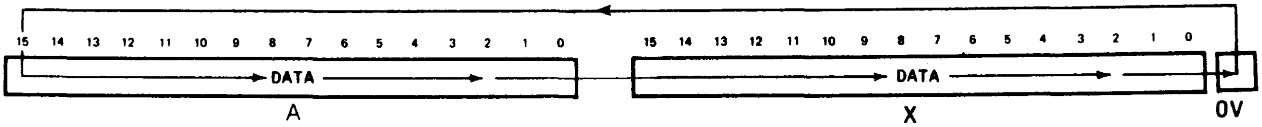


Figure 3-26. Long Rotate Right

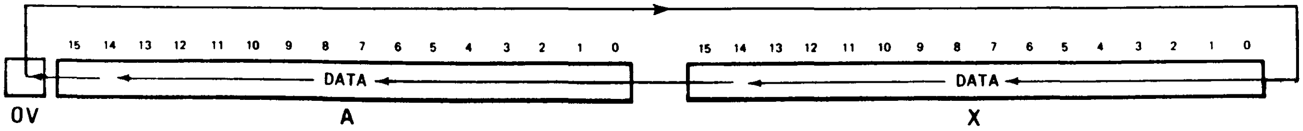


Figure 3-27. Long Rotate Left

LRL LONG ROTATE LEFT. Contents of A and X Registers are shifted left through OV n places. OV is shifted into X_0 , X_{15} is shifted into A_0 and A_{15} is shifted into OV. No bits are lost when this shift is executed. A, X and OV act as a 33-bit register.

LRR LONG ROTATE RIGHT. Contents of A and X Registers are shifted right through OV n places. OV is shifted into A_{15} , A_0 is shifted into X_{15} and X_0 is shifted into OV. No bits are lost when this shift is executed. A, X and OV act as a 33-bit register.

3.7 REGISTER CHANGE INSTRUCTIONS

3.7.1 Format

Register change instructions perform arithmetic and logical operations involving the A register, the X register and/or the overflow (OV) indicator. There are no operands in this class. The Register Change Instruction format is shown in Figure 3-28.

[LABEL] OP-CODE [COMMENTS]

There is no expression in the operand field.

Figure 3-28. Register Change Format

3.7.2 A Register Change Instructions

ARM A REGISTER TO MINUS ONE. Sets the A register to -1 (:FFFF).

ARP A REGISTER TO PLUS ONE. Sets the A register to +1.

CAR COMPLEMENT A REGISTER. Performs one's complement of the A register.

DAR DECREMENT A REGISTER. Subtracts one from the A register. OV is set if arithmetic overflow occurs.

IAR INCREMENT A REGISTER. Adds one to the A register. OV is set if arithmetic overflow occurs.

NAR NEGATE A REGISTER. Performs two's complement of the A register. OV is set if arithmetic overflow occurs.

ZAR ZERO A REGISTER. Sets the A register to zero.

3.7.3 X Register Change Instructions

CXR COMPLEMENT X REGISTER. Performs one's complement of the X register.

DXR DECREMENT X REGISTER. Subtracts one from the X register. OV is set if arithmetic overflow occurs.

IXR INCREMENT X REGISTER. Adds one to the X register. OV is set if arithmetic overflow occurs.

NXR NEGATE X REGISTER. Performs two's complement of the X register. OV is set if arithmetic overflow occurs.

XRM X REGISTER TO MINUS ONE. Sets the X register to -1 (:FFFF).

XRP X REGISTER TO PLUS ONE. Sets the X register to +1.

ZXR ZERO X REGISTER. Sets the X register to zero.

3.7.4 OV Register Change Instructions

COV COMPLEMENT OVERFLOW. Complements OV.

LAO LSB OF A TO OVERFLOW. Bit 0 of the A register is copied into OV. The A register is unchanged.

LXO LSB OF X TO OVERFLOW. Bit 0 of the X register is copied into OV. The X register is unchanged.

ROV RESET OVERFLOW. Resets OV (to zero).

SAO SIGN OF A TO OVERFLOW. Bit 15 of the A register is copied into OV. The A register is unchanged.

SOV SET OVERFLOW. Sets OV (to one).

SXO SIGN OF X TO OVERFLOW. Bit 15 of the X register is copied into OV. The X register is unchanged.

3.7.5 Multi-Register Change Instructions

ANA AND OF A AND X TO A. The A and X registers are logically ANDed; result is placed in A. The X register is unchanged.

ANX AND OF A AND X TO X. The A and X registers are logically ANDed; result is placed in X. The A register is unchanged.

AXM A AND X REGISTERS TO MINUS ONE. Sets the A and X registers to -1 (:FFFF).

AXP A AND X REGISTERS TO PLUS ONE. Sets the A and X registers to +1.

CAX COMPLEMENT OF A TO X. Performs one's complement of A register and places result in the X register. The A register is unchanged.

CXA COMPLEMENT OF X TO A. Performs one's complement of the X register and places result in the A register. The X register is unchanged.

DAX DECREMENT A TO X. Subtracts one from the A register and places the result in X. The A register is unchanged. OV is set if arithmetic overflow occurs.

DXA DECREMENT X TO A. Subtracts one from the X register and places the result in A. The X register is unchanged. OV is set if arithmetic overflow occurs.

IAX INCREMENT A TO X. Adds one to the A register and places the result in X. The A register is unchanged. OV is set if arithmetic overflow occurs.

IXA INCREMENT X TO A. Adds one to the X register and places the result in A. The X register is unchanged. OV is set if arithmetic overflow occurs.

NAX NEGATE A TO X. Performs two's complement of the A register and places the result in X. The A register is unchanged. OV is set if arithmetic overflow occurs.

NRA NOR OF A AND X TO A. Performs logical NOR of the A and X registers and places the result in A. The X register is unchanged.

NRX NOR OF A AND X TO X. Performs logical NOR of the A and X registers and places the result in X. The A register is unchanged.

NXA NEGATE X TO A. Performs two's complement of the X register and places the result in A. Th X register is unchanged. OV is set if arithmetic overflow occurs.

TAX TRANSFER A TO X. Transfers the A register to the X register. The A register is unchanged.

TXA TRANSFER X TO A. Transfers the X register to the A register. The X register is unchanged.

ZAX ZERO A AND X. Sets the A and X registers to zero.

3.7.6 Console Register Instructions

ICA INPUT CONSOLE DATA REGISTER TO A. Loads the 16-bit contents of the Console Data register into the A register.

ICX INPUT CONSOLE DATA REGISTER TO X. Loads the 16-bit contents of the Console Data register into the X register.

ISA INPUT CONSOLE SENSE REGISTER TO A. Loads the 4-bit contents of the Console Sense register into the least significant 4 bits of the A register. The most significant 12 bits of the A register are set to zero.

ISX INPUT CONSOLE SENSE REGISTER TO X. Loads the 4-bit contents of the Console Sense register into the least significant 4 bits of the X register. The most significant 12 bits of the X register are set to zero.

OCA OUTPUT A TO CONSOLE DATA REGISTER. Transfers the A register to the 16-bit Console Data register. The A register is unchanged.

OCX OUTPUT X TO CONSOLE DATA REGISTER. Transfers the X register to the 16-bit Console Data register. The X register is unchanged.

3.8 CONTROL INSTRUCTIONS

3.8.1 Format

Control instructions are used for general status manipulation in the computer. The general format for these instructions is shown in Figure 3-29.

[LABEL]	OP-CODE	[EXPRESSION]	[COMMENTS]

There is no expression in the operand field, except for the SIN and STOP instructions.
For SIN, the expression must be absolute and in the range 1 through 6.
For STOP, the expression must be absolute and in the range 1 through 255.

Figure 3-29. Control Format

3.8.2 Processor Control Instructions

HLT HALT. Halts the computer

NOP NO OPERATION. Causes a pause in the program

STOP HALT WITH OPERAND. Halts the computer with the specified operand occupying the least significant 8 bits of the I (instruction) register. The operand may be any absolute expression in the range 0 through 255. As an example, STOP 5 would halt with :0805 in the I register.

WAIT WAIT FOR INTERRUPT. Executes as JMP $. Program loops on one location waiting for an interrupt. After the interrupt is serviced, the return is made to the WAIT instruction to wait for further interrupts.

3.8.3 Mode Control Instructions

SBM SET BYTE MODE. Conditions the computer to address byte (8 bit) operands rather than word operands when executing Memory Reference instructions (see Sec. 3.2.2).

SWM SET WORD MODE. Conditions the computer to address word (16 bit) operands rather than byte operands when executing Memory Reference instructions (see Sec. 3.2.1). The "reset" condition of the computer is the Word Mode.

3.8.4 Status Control Instructions

The format of the 8-bit Computer Status Word is shown in Figure 3-30:

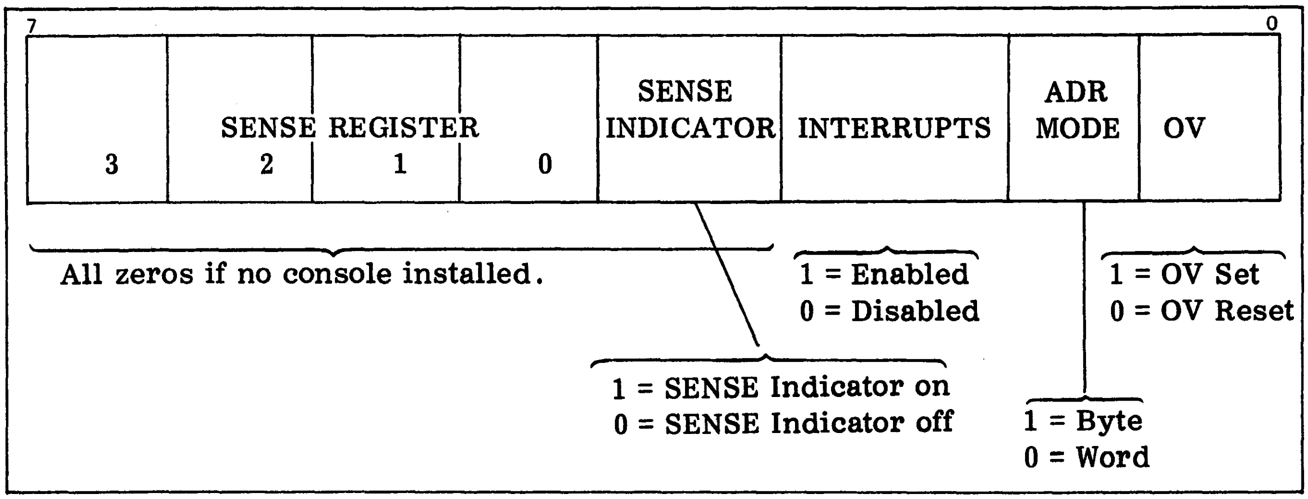


Figure 3-30. Computer Status Word Format

SIA STATUS INPUT TO A. Reads the Computer Status Word into the least significant 8 bits of the A register. Resets OV and sets the Address Mode to the Word Mode. The state of interrupts is unchanged. The most significant 8 bits of the A register are set to zero.

SIN STATUS INHIBIT. Inhibits interrupts and places the computer in the Word Mode for the number of succeeding instructions specified by the operand. The operand may be any absolute expression in the range 1 through 6. As an example, execution of the SIN 4 instruction will force Word Mode operation for the four succeeding instructions and will inhibit interrupt acknowledgement until after completion of five succeeding instructions since interrupts are serviced at the end of instruction execution.

SIX STATUS INPUT TO X. Reads the Computer Status Word into the least significant 8 bits of the X register. Resets OV and sets the Address Mode to the Word Mode. The state of interrupts is unchanged. The most significant 8 bits of the X register are set to zero.

SOA STATUS OUTPUT FROM A. Writes the least significant 8 bits of the A register into the computer status register. However, this instruction does not alter the interrupt indicator.

SOX STATUS OUTPUT FROM X. Writes the least significant 8 bits of the X register into the computer status register. However, this instruction does not alter the interrupt indicator.

3.8.5 Interrupt Control Instructions

CID CONSOLE INTERRUPT DISABLE. Disables the Console interrupt.

CIE CONSOLE INTERRUPT ENABLE. Enables the Console interrupt. Console interrupts are generated each time the INT switch is pressed when the computer is in the Run mode. Console interrupts are also under the control of the EIN/DIN instructions. A special jumper option on the processor option card allows the console interrupt to be enabled independent of the EIN/DIN instructions.

DIN DISABLE INTERRUPTS. Prevents the Processor from responding to any interrupts. A special jumper option on the processor option card allows Power Fail, Console and Trap interrupt operation independent of DIN.

EIN ENABLE INTERRUPTS. Enables the recognition of external interrupts by the computer. This instruction does not take effect until completion of the next instruction in sequence.

PFD POWER FAIL INTERRUPT DISABLE. When the option placing Power Fail Interrupts outside EIN and DIN control is selected, the Power Fail Interrupt Disable (PFD) instruction inhibits recognition of Power Fail interrupts.

PFE POWER FAIL INTERRUPT ENABLE. When the option placing Power Fail Interrupt outside EIN and DIN control is selected, the Power Fail Interrupt Enable (PFE) instruction allows recognition of Power Fail interrupts. If Power Fail interrupts were disabled at the issuance of PFE, the PFE does not take effect until after two succeeding instructions have been executed.

TRP TRAP. Generates an interrupt to the Console interrupt location if interrupts are enabled or if the special jumper option placing Power Fail, Console and Trap interrupts outside EIN/DIN control is in use. In the latter case, there is no enable or disable instruction associated with the trap interrupt.

3.9 INPUT/OUTPUT INSTRUCTIONS

Input/Output instructions are either single word or multiple word instructions. All single word instructions use the same format, illustrated in Figure 3-31. Multiple word formats are described separately in Sections 3.9.4 and 3.9.5. All Input/Output instructions have 8 bits available for addressing a particular peripheral device and a particular register or function within a device. These 8 bits are arbitrarily divided into a 5 bit Device Address field to address one of 32 devices and a 3 bit Function Code field to specify one of 8 registers or functions within a device. The device address and function code may be expressed as either one or two self-defined (i.e., numeric expressions) or predefined absolute expressions. If a single expression is used, it must be in the range :0 through :FF and it represents both the device address and function code. If two expressions are used, the first must be the device address in the range :0 through :1F and the second must be the function code in the range :0 through :7.

[LABEL]	OP CODE	EXPRESSION 1 [,EXPRESSION 2][COMMENTS]
	If EXPRESSION 2 is not present, EXPRESSION 1 must be absolute and in the range :0 through :FF. If EXPRESSION 2 is present, EXPRESSION 1 must be absolute and in the range :0 through :1F. EXPRESSION 2 must be absolute and in the range :0 through :7.	

Figure 3-31. Single Word Input/Output Instruction Format

Both Word and Byte Input/Output instructions are available. Whether a full 16-bit word or an 8-bit byte is transferred depends upon the instruction used and is not effected by the word/byte addressing mode flip-flop (SWM/SBM) used by Memory Reference Instructions.

3.9.1 Control Input/Output Instructions

The Control Input/Output instructions are divided into Sense and Select instructions. Sense instructions are used to test the status of a function within the addressed peripheral device. Select instructions are used to control the operation of specific functions within the addressed peripheral device. The functions tested or controlled depend upon the individual peripheral device. Control Input/Output instructions use the Single Word Input/Output format as shown in Figure 3-31.

3.9.1.1 Sense Instructions

SEN SENSE AND SKIP ON RESPONSE. Tests the specified function in the specified peripheral device. If a true response is obtained, a one-word skip is generated. If a false response is obtained, the next instruction in sequence is executed.

SSN SENSE AND SKIP ON NO RESPONSE. Tests the specified function in the specified peripheral device. If a false response is obtained, a one-word skip is generated. If a true response is obtained, the next instruction in sequence is executed.

3.9.1.2 Select Instructions

SEL SELECT FUNCTION. Transmits the specified function code to the specified peripheral device along with a Select Control signal. All zeros are placed on the Data bus. Any action generated is a function of the interface design of the peripheral device.

SEA SELECT AND PRESENT A. Transmits the specified function code to the specified peripheral device along with a Select Control signal. The contents of the A register are placed on the Data bus. Any action generated is a function of the interface design of the peripheral device.

SEX SELECT AND PRESENT X. Transmits the specified function code to the specified peripheral device along with a Select Control signal. The contents of the X register are placed on the Data bus. Any action generated is a function of the interface design of the peripheral device.

3.9.2 Word Input/Output Instructions

Word Input/Output instructions transmit 16 bits of data at a time. They are divided into Unconditional and Conditional instructions. Conditional instructions are automatically repeated until a true sense response is obtained, at which time the data transmission occurs and the next instruction in sequence is executed. Response to an interrupt may occur "within" a conditional input/output instruction - i.e., during a false sense response an interrupt can be acknowledged and the computer will return to execution of the conditional input/output instruction after servicing the interrupt. If a word input is requested from an 8-bit device, the upper 8 bits will be input as zeros. If an output is performed to an 8-bit device the upper 8 bits will be ignored by the device.

3.9.2.1 Unconditional Word Input/Output Instructions

INA INPUT TO A REGISTER. Unconditionally transfers a full 16-bit data word from the addressed peripheral device to the A register.

INAM INPUT TO A REGISTER MASKED. Unconditionally transfers a full 16-bit data word from the addressed peripheral device to the A register. The data word is logically ANDed with the previous contents of the A register. The results are placed in the A register.

INX INPUT TO X REGISTER. Unconditionally transfers a full 16-bit data word from the addressed peripheral device to the X register.

INXM INPUT TO X REGISTER MASKED. Unconditionally transfers a full 16-bit data word from the addressed peripheral device to the X register. The data word is logically ANDed with the previous contents of the X register. The results are placed in the X register.

OTA OUTPUT A REGISTER. Unconditionally transfers the full 16-bit contents of the A register to the addressed peripheral device.

OTX OUTPUT X REGISTER. Unconditionally transfers the full 16-bit contents of the X register to the addressed peripheral device.

OTZ OUTPUT ZERO. Unconditionally transfers a 16-bit zero word to the addressed peripheral device.

3.9.2.2 Conditional Word Input/Output Instructions

RDA READ WORD TO A REGISTER. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred to the A register.

RDAM READ WORD TO A REGISTER MASKED. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred to the A register and logically ANDed with the previous contents of the A register. The results are placed in the A register.

RDX READ WORD TO X REGISTER. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit data word is transferred to the X register.

RDXM READ WORD TO X REGISTER MASKED. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, a full 16-bit word is transferred to the X register and logically ANDed with the previous contents of the X register. The results are placed in the X register.

WRA WRITE FROM A REGISTER. Senses the specified condition in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, the full 16-bit contents of the A register are transferred to the addressed peripheral device.

WRX WRITE FROM X REGISTER. Senses the specified condition in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, the full 16-bit contents of the X register are transferred to the addressed peripheral device.

WRZ WRITE ZERO. Senses the specified condition in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, a 16-bit zero word is transferred to the addressed peripheral device.

3.9.3 Byte Input Instructions

Byte Input instructions input 8 bits of data to the least significant byte of a target register leaving the MSB byte unchanged. They are divided into Unconditional and Conditional instructions. Conditional instructions are automatically repeated until true sense responses are obtained, at which time the data transmission occurs and the next instruction in sequence is executed. Response to an interrupt may occur "within" a conditional input/output instruction - i.e., during a false sense response an interrupt can be acknowledged and the computer will return to execution of the conditional instruction after servicing the interrupt. Byte Input instructions use the Single Word Input/Output Instruction format as shown in Figure 3-31.

3.9.3.1 Unconditional Byte Input Instructions

IBA INPUT BYTE TO A REGISTER. Unconditionally transfers an 8-bit byte from the addressed peripheral device to the least significant 8 bits of the A register. The most significant 8 bits of the A register are unchanged.

IBAM INPUT BYTE TO A REGISTER MASKED. Unconditionally transfers an 8-bit byte from the addressed peripheral device to the least significant 8 bits of the A register. The data byte is logically ANDed with the previous contents of the least significant 8 bits of the A register. The results are placed in the least significant 8 bits of the A register and the most significant 8 bits of the A register are unchanged.

IBX INPUT BYTE TO X REGISTER. Unconditionally transfers an 8-bit byte from the addressed peripheral device to the least significant 8 bits of the X register. The most significant 8 bits of the X register are unchanged.

IBXM INPUT BYTE TO X REGISTER MASKED. Unconditionally transfers an 8-bit byte from the addressed peripheral device to the least significant 8 bits of the X register. The data byte is logically ANDed with the previous contents of the least significant 8 bits of the X register. The results are placed in the least significant 8 bits of the X register and the most significant 8 bits of the X register are unchanged.

3.9.3.2 Conditional Byte Input Instructions

RBA READ BYTE TO A REGISTER. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred to the least significant 8 bits of the A register. The most significant 8 bits of the A register are unchanged.

RBAM READ BYTE TO A REGISTER MASKED. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred to the least significant 8 bits of the A register and logically ANDed with the previous contents of the least significant 8 bits of the A register. The results are placed in the least significant 8 bits of the A register and the most significant 8 bits of the A register are unchanged.

RBX READ BYTE TO X REGISTER. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred to the least significant 8 bits of the X register. The most significant 8 bits of the X register are unchanged.

RBXM READ BYTE TO X REGISTER MASKED. Senses the specified data source in the addressed peripheral device. If a false response is received, the instruction is repeated (and interrupts may be acknowledged). When a true response is received, an 8-bit data byte is transferred to the least significant 8 bits of the X register and logically ANDed with the previous contents of the least significant 8 bits of the X register. The results are placed in the least significant 8 bits of the X register and the most significant 8 bits of the X register are unchanged.

3.9.4 Block Input/Output Instructions

The two instructions in this class provide for high-speed, full 16-bit word data transfers. The Processor is totally dedicated to these instructions until the specified block of data has been completely transferred - i.e., no interrupts may be serviced until the instructions have been executed to completion.

The Block Transfer instructions are double-word instructions. The second word of the instruction contains the base address minus one of the associated data buffer in memory. The X register contains the (positive) number of words to be transferred - i.e., the length of the data buffer. The memory location of each word transferred is obtained by summing the base address minus one and the contents of the X register. As each data word is transmitted, the X register is decremented by one. Thus, the data buffer is output or input in descending order, beginning with the highest memory location and ending with the lowest memory location (base address plus length -1). When the X register is decremented to zero, the next instruction in sequence is executed.

The format for the Block Transfer instructions is shown in Figure 3-32.

```
[LABEL]     OP-CODE     EXPRESSION 1  [,EXPRESSION 2]   [COMMENTS]

[LABEL]     OP-CODE     EXPRESSION 3                    [COMMENTS]
```

If EXPRESSION 2 is not present, EXPRESSION 1 must be absolute and in the range : 0 through : FF.
If EXPRESSION 2 is present, EXPRESSION 1 must be absolute and in the range : 0 through : 1F.
EXPRESSION 2 must be absolute and in the range : 0 through : 7.
EXPRESSION 3 is an absolute or relocatable expression giving the base address -1 of the buffer.

Figure 3-32. Block Input/Output Instruction Format

The expressions in the operand field of these instructions must be either self-defining (i.e., numeric expressions) or predefined absolute expressions. If only one expression is present, it must be in the range : 0 through : FF. The high-order 5 bits represent the peripheral device address and the low-order 3 bits represent the function code. If two expressions are present, the first must be in the range : 0 through : 1F and the second must be in the range : 0 through : 7. The first expression represents a peripheral device address, and the second expression represents a function code.

The expression in the operand field of the DATA statement must not be an indirect address (no*). It represents the memory location one less than the start (the low-order memory location) of the data buffer.

BIN   BLOCK IN. Senses the specified data source in the addressed peripheral device and inputs a full 16-bit data word from the selected device each time a true sense response is received. The instruction executes until all data words have been input. Interrupts may be acknowledged only after completion of the instruction. The A register is unchanged.

BOT   BLOCK OUT. Senses the specified data source in the addressed peripheral device and outputs a full 16-bit data word to the selected device each time a true sense response is received. The instruction executes until all data words have been output. Interrupts may be acknowledged only after completion of the instruction. The A register is unchanged.

3.9.5 Automatic Input/Output Instructions

The Automatic Input/Output instructions (Auto I/O) provide data transfers directly between memory and peripheral devices without affecting the A and X registers. These multiple word instructions effectively constitute complete I/O subroutines, thus facilitating their use as interrupt instructions. They increment a (negative) data word or byte counter, increment a data word or byte pointer and transfer a data word or byte between memory and a peripheral device.

Each Auto I/O instruction occupies three words in memory. The first word contains the instruction itself, the second word contains the two's complement (negative) of the word or byte count for the data buffer and the third word contains an address pointer specifying the location one less than the first (lower-order memory) location of the data buffer. The data buffer is input or output in order of ascending memory locations (low-order to high-order). The format for these instructions is shown in Figure 3-33.

```
[LABEL]     OP-CODE           EXPRESSION 1  [,EXPRESSION 2]  [COMMENTS]

[LABEL]     DATA              EXPRESSION 3                   [COMMENTS]

[LABEL]    ( BAC )            EXPRESSION 4                   [COMMENTS]
           (  or )
           ( DATA)
```

If Expression 2 is not present, Expression 1 must be absolute and in the range : 0 thru : FF.
If Expression 2 is present, Expression 1 must be present and in the range : 0 thru : 1F.
Expression 2 must be absolute and in the range : 0 thru : 7.
Expression 3 is the negative word or byte count of the buffer.
Expression 4 is an absolute or relocatable expression defining the base address -1 of the buffer.

Figure 3-33. Automatic Input/Output Instruction Format

The expressions in the operand fields of the first two statements must be either self-defined (i.e., numeric expressions) or predefined absolute expressions. If only one expression is present in the operand field of the instruction, it must be in the range : 0 through : FF. The high-order 5 bits represent the device address and the low order 3 bits represent the function code. If two expressions are present, the first must be in the range : 0 through : 1F, and the second must be in the range : 0 through : 7. The first expression represents a peripheral device address, and the second expression represents a function code.

The absolute expression for the second word represents the negative (two's complement) data word or byte count for the buffer being transmitted. This word is incremented once prior to each data word or byte transfer and must be preset each time a block of data is to be transferred.

The expression in the operand field of the third word of the instruction is an address pointer specifying the byte or word location one less than the start of the data buffer. This word is incremented once prior to each data word or byte transferred and must be preset each time a block of data is to be transferred.

Operation of Automatic I/O Instructions differ depending upon usage. When used as a normal in-line program instruction, the Automatic I/O instruction sequence is as shown in Figure 3-34. Each time the instruction is executed the word/byte count and address pointer are incremented, one word or byte of data is transferred and then the incremented word count is examined. If the word count has not yet reached zero, the next instruction executed is from location P+4. If the word count reached zero, the next instruction executed is at location P+3(End of Block exit location). Since Automatic I/O instructions do not sense for the peripheral device to be ready prior to data transfer, a sense instruction should be used prior to each execution (one word transferred) of the instruction, i.e., to transfer a block location, P+4 would normally contain a jump back to a sense instruction prior to location P.

P	Automatic I/O Instruction
P+1	Word/Byte Counter (negative)
P+2	Address Pointer (starts address -1)
P+3	End of Block Exit (Word Count = 0)
P+4	Next Instruction (Word Count ≠ 0)

Figure 3-34. In-line Auto I/O Instruction Sequence

Automatic I/O instructions may also be used under interrupt control at an interrupt location to implement a Direct Memory Channel. In this application, the Automatic I/O instruction is executed once each time the peripheral device indicates that it is ready for a data transfer by interrupting to the location containing the Automatic I/O instruction. Since the Automatic I/O instructions do not alter any processor registers, no jumping to an interrupt subroutine to save registers, status, and return location is required. The Automatic I/O instruction is, itself, a one word subroutine. When executed under interrupts, the skips after execution are suppressed. Instead, if the word count has not reached zero after a data transfer, control is passed directly back to the main-line program at the point it was interrupted. If the word count did reach zero, a special signal(ECHO-) is sent to the peripheral device to indicate that it should stop requesting further data transfers. The Automatic I/O instruction transfers control back to the main-line program whether ECHO is set or not. Upon receipt of ECHO, the peripheral device stops data transfer requests, performs any stop action required (i.e. CRC checking or generation for magnetic tape), and then generates an End-of-Block interrupt so that the program can process the data block input or prepare another block for output. Although the End-of-Block interrupt can be vectored to any location by the peripheral controller, it is standard practice for the controller to vector this interrupt to four locations beyond the data transfer interrupt location. Figure 3-35 illustrates the typical usage of Automatic I/O instructions under interrupts.

Data Transfer Interrupt Location	I	Automatic I/O Instruction
	I+1	Word/Byte Counter (negative)
	I+2	Word/Byte Counter (negative)
	I+3	Unused
End-of-Block Interrupt Location	I+4	JST*$-1 (jump and store to END-of-Block subroutine)

Figure 3-35. Interrupt Location Auto I/O Instruction Sequence

AIB AUTOMATIC INPUT BYTE TO MEMORY. Increments the byte counter and the byte address pointer and unconditionally inputs one 8-bit byte from the specified data source in the addressed peripheral device to the updated byte location in memory addressed by the address pointer. When the byte count is incremented to zero, the normal one-word skip after the data transfer does not take place, or when used as an interrupt instruction, an Echo signal to the addressed peripheral device is generated.

AIN AUTOMATIC INPUT WORD TO MEMORY. Increments the data word counter and the address pointer and unconditionally inputs a full 16-bit data word from the specified data source in the addressed peripheral device to the updated word location in memory addressed by the address pointer. When the word count is incremented to zero, the normal one-word skip after the data transfer does not take place, or when used as an interrupt instruction an ECHO signal to the addressed peripheral device is generated.

AOB AUTOMATIC OUTPUT BYTE FROM MEMORY. Increments the byte counter and the byte address pointer and unconditionally outputs one 8-bit byte from the updated byte location in memory addressed by the byte address pointer to the specified data source in the addressed peripheral device. When the byte count is incremented to zero, the normal one-word skip does not take place, or when used as an interrupt instruction, an ECHO signal to the addressed peripheral device is generated.

AOT AUTOMATIC OUTPUT WORD FROM MEMORY. Increments the data word counter and the address pointer and unconditionally outputs a full 16-bit data word from the updated word location in memory addressed by the address pointer to the specified data source in the addressed peripheral device. When the word count is incremented to zero, the normal one-word skip after the data transfer does not take place, or when used as an interrupt instruction, an Echo signal to the addressed peripheral device is generated.

3.10 ASSEMBLER CONTROL DIRECTIVES

The assembler control directives provide for conditional assembly of source statements and establish and/or alter the value and relocatability of the program location counter. If a label is present on any of these control directives, it is in general assigned the current value and relocation attribute of the program location counter. These directives do not generate computer instruction words.

3.10.1 Conditional Assembly Controls

The IFF (If False) and IFT (If True) directives are provided to conditionally assemble subsequent lines of source code. The format for these two instructions is the following:

```
[LABEL]     OP CODE    ABSOLUTE EXPRESSION       [COMMENTS]
                            =0 ——→False
                            ≠0 ——→True
```

Figure 3-36. Begin Conditional Assembly Directives Format

The absolute expression must be previously defined (but not as an external). The last line affected must be an ENDC directive which signals the end of the conditional assembly. The ENDC directive has the following format:

```
[LABEL]     ENDC                                  [COMMENT]
            There is no expression in the operand field.
```

Figure 3-37. End Conditional Assembly Directive Format

IFF and IFT directives must not be nested - i.e., no other IFF or IFT directive can appear between a given IFF or IFT directive and its associated ENDC directive. If the value of the absolute expression is zero, it is defined as false. If it is not equal to zero, it is defined as true. If the value of the expression satisfies the condition of the directive (false for IFF and true for IFT) the source lines between the directive and its associated ENDC directives are assembled. If the conditions are not met, the source lines are deleted (not assembled). The program END directive must not appear between an IFF or IFT directive and its associated ENDC directive.

3.10.2 Program Location Controls

The following directives control the contents and relocation attributes of the program location counter. The format for these directives is the following:

LABEL	OP CODE	EXPRESSION	COMMENTS

Figure 3-38. Location Control Directive Format

If an expression is present, it must be predefined or self-defined (e.g., a numeric expression). It cannot be externally defined. Each program should start with an ABS, REL or ORG directive and end with an END directive.

ABS ABSOLUTE ASSEMBLY. Sets the relocation attribute of the program location counter to absolute. If an expression is present, the program location counter is set to the value of the expression. Otherwise, the contents of the program location counter are unchanged. Comments may appear on an ABS directive only if an expression is present. If a label is present, it is assigned the value of the expression.

REL RELOCATABLE ASSEMBLY. Sets the relocation attribute of the program location counter to relative. If an expression is present, the program location counter is set to the value of the expression. If no expression is present, the contents of the program location counter are unchanged and the comments field must be blank. If a label is present, it is assigned the value of the expression.

ORG ORIGIN. Sets the program location counter to the value of the expression. The expression must be present and defined. If a label is present, it is assigned the value of the expression. The relocation attribute of the program location counter is unchanged.

END END OF ASSEMBLY. Signifies the end of an assembly. If an expression is present, it is interpreted by the object loader as the execution transfer address at the end of a successful load. Since the object loader does not distinguish between END directives in main programs and subprograms, only the main program should include a transfer address. Comments may appear on an END directive only if an expression is present. If a label is present, it is assigned the current value of the program location counter.

3.11 DATA AND SYMBOL DEFINITION DIRECTIVES

3.11.1 Formats

The directives discussed in this section define various types of data, including buffers, address pointers and character strings. Symbol Definition directives are also discussed. The various formats involved are shown below in Figure 3-39.

LABEL	BAC	EXPRESSION	COMMENTS
LABEL	DATA	* EXPRESSION 1 , * EXPRESSION 2..	COMMENTS
LABEL	TEXT	'CHARACTER STRING'	COMMENTS
LABEL	RES	EXPRESSION , EXPRESSION	COMMENTS
LABEL	EQU or SET	EXPRESSION	COMMENTS

Figure 3-39. Data and Symbol Definition Directive Format

3.11.2 Directives

BAC BYTE ADDRESS CONSTANT. Generates a byte address constant (or pointer). Symbolic items in the expression are assumed to be "word address" values and numeric items are assumed to be "byte counts" or "byte address" values. Values of symbolic items are "doubled" to generate byte address values.

DATA DATA DEFINITION. Places values of expressions in sequential memory locations. The operand field contains one or more expressions separated by commas. Any valid expression may be used. The expressions are evaluated one at a time and generated as sequential constants. If a label is present, it is assigned the location of the first constant generated. Indirect address pointers are specified by preceding the expressions in the operand field with asterisks (*).

TEXT TEXT STRING. Generates an 8-bit ASCII character string, packed two characters per word, for use as data. Characters are packed left to right in the most significant byte then the least significant byte of sequential memory words. Trailing character positions are filled with blanks (:A0) to complete full words. The character string must be surrounded by single quotes ('). When a quote is desired as a character in the string, two contiguous single quotes must appear within the string. If a label is present, it is assigned the location of the first pair of characters generated.

RES RESERVE STORAGE. Reserves storage for the number of words specified by the first expression. If the second expression is present, it defines a constant which is to be stored in each of the reserved memory locations. Both expressions must be either self-defined (e.g., a numeric expression), or predefined, absolute expressions. If the second expression is not present, the object loader will not alter the reserved memory locations at load time. If a label is present, it is assigned the location of the first reserved memory word.

EQU EQUATE SYMBOL. Assigns the value and relocatability of the expression in the operand field to the symbol in the label field. The symbol in the label must not be defined elsewhere. The expression must be either a self-defined (e.g., a numeric expression) or a predefined expression. No machine instructions are generated.

SET SET SYMBOL. Assigns the value and relocatability of the expression in the operand field to the symbol in the label field. This directive is identical to the EQU directive, except that the symbol being defined may be redefined by another SET directive. No machine instructions are generated.

3.12 PROGRAM LINKAGE DIRECTIVES

3.12.1 Formats

The directives discussed in this section provide for linkages between programs which have been assembled separately, but are to be loaded and executed together. The formats for the three directives are shown below in Figure 3-40.

```
           { NAM  }
LABEL      {  or  }    EXPRESSION 1 , EXPRESSION 2,... [COMMENTS]
           { EXTR }
[LABEL]      REF                                       [COMMENTS]
```

Figure 3-40. Program Linkage Directive Formats

Expressions must be symbolic names defined within the program segment for NAM or referenced by the program segment for EXTR. REF may not have an Operand Field expression.

3.12.2 Directives

NAM EXTERNAL NAME DEFINITION. Defines external entry or reference points within the current program. The operand field of the NAM directive contains one or more symbols separated by commas. Each name (or symbol) appearing in the operand field must be defined in the body of the program. When this directive is used, it must precede all data generating statements. If a label is present, it is assigned a zero value and a relative relocation attribute. No machine instructions are generated.

EXTR EXTERNAL REFERENCE-SCRATCHPAD. Declares external symbols referenced by the current program. The object loader links these declared external symbols through the scratchpad (first 256 words of memory) at load time. Each name or symbol appearing in the operand field and also referenced by the current program is output to the object loader at load time. Since they are not defined within the current program, these symbols must not be used in multi-term expressions. References to an EXTR-defined symbol must be direct, since the assembler automatically generates indirect references through the scratchpad. If a label is present, it is assigned the current value and relocation attribute of the program location counter. No machine instructions are generated.

REF EXTERNAL REFERENCE-POINTER. Defines the current location as linkage for reference to the external symbol contained in the label field. At load time, the address at which the external symbol is loaded is stored in the memory location of the REF directive.

3.13 SUBROUTINE DEFINITION DIRECTIVES

The following directives are provided primarily for documentation purposes. They are used to facilitate determining the limits of subroutines in assembler output listings. The formats are described below in Figure 3-41.

LABEL	ENT		[COMMENTS]
[LABEL]	RTN	EXPRESSION	[COMMENTS]

Figure 3-41. Subroutine Definition Directive Formats

No operand field is allowed for ENT. The expression for RTN may be any expression defining the location of a subroutine return pointer (normally the label for the subroutine ENT).

ENT SUBROUTINE ENTRY. Reserves a word to hold the return address from a subroutine call (JST). The assembler generates a HLT instruction for this directive. Any source statement which causes one word to be reserved could be used in its place.

RTN SUBROUTINE RETURN. Generates an indirect Jump via the symbol in the operand field (JMP *Expression). Note that the expression is direct.

3.14 LISTING FORMAT AND ASSEMBLER INPUT CONTROLS

The following controls are provided for the purpose of formatting assembler output listings. With the exception of the TITL directive, these controls are simply special characters in the first column or position of a source line. The format for the TITL directive is shown below in Figure 3-42:

```
        TITL (one blank) ANY COMBINATION OF ALPHANUMERIC CHARAC-
                         TERS NOT EXCEEDING 72 CHARACTERS IN LENGTH
```

Figure 3-42. Title Directive Format

No label field is allowed for TITL.

TITL PAGE EJECT WITH TITLE. Generates a Top-of-Form to the assembler listing device. The page number is then printed, followed (on the same line) by the character string specified in the operand field. The same character string is printed with the page number at the top of each page until a new TITL directive is encountered. If these directives are to be used throughout a program, the first TITL directive should appear as the first source line of the program - ahead of comments, user-defined op code definitions and origin statements.

. (Period) PAGE EJECT WITHOUT TITLE. Generates a Top-of-Form to the assembler listing device. This control must appear as the first character of a source statement. The rest of the input line will be ignored.

*(Asterisk) COMMENT LINE. Allows source line comments to be exactly duplicated on the assembler listing device. This control must appear as the first character of the source statement. All characters following the asterisk on the source statement are duplicated on the output listing. Comments may appear before origin statements at the beginning of a program.

↑(Up arrow) PAUSE. Halts the assembler. Assembly is continued by pressing the RUN button. This control is most useful when paper tape input is used. The up-arrow must appear as the first character of a source line. The rest of the input line will be ignored.

3.15 USER-DEFINED OPERATION CODE DIRECTIVE

User-defined operation code directives allow the user to name or define his own instruction mnemonics for the current assembly. If included in a program, user-defined op code directives must precede all source statements other than comments or TITL directives. The user is referred to the applicable Assembler Reference Manual for a detailed discussion of their usage.

Section 4

INPUT/OUTPUT AND INTERRUPT OPERATIONS

4.1 INTRODUCTION

4.1.1 Discussion of Input/Output Operations

Communication with the standard peripheral devices generally consists of operations which can be treated as members of three major categories - Control, Sense, and Input/Output operations. The precise definitions of the various instructions, function codes and status words depend on the design of the individual peripheral interfaces.

4.1.1.1 Control

Initialization and mode/status control of peripheral devices are usually accomplished via the Select (SEL) and Select-and-Present (SEA and SEX) instructions. When a teletype is to be used, for instance, the teletype must first be commanded into the Keyboard Mode. The SEL instruction has the following format:

```
SEL    DA,FC
```

A given peripheral device (DA) can have as many as eight different function codes (FC = 0 through 7). The SEA and SEX instructions are useful for devices which contain status or address registers which must be set or initialized by transmitting data from the A or X registers.

The SEL instruction, in effect, commands the peripheral device and puts all zeros on the data lines to the devices. The SEA and SEX instructions are used for devices which require non-zero data values during command sequences (e.g., the Teletype for Full-Duplex operation).

The Control instructions prepare the peripheral devices for data transmission, but do not necessarily insure a true (device ready) Sense Response.

4.1.1.2 Sense

Once a peripheral device has been prepared for transmission of data with the proper commands, it is necessary to determine whether the device is ready to accept or send the data. This is accomplished using the Sense-Skip-on-Response (SEN) and Sense-Skip-on-no-Response (SSN) instructions. One or the other of these instructions should immediately precede an unconditional data transmission sequence such that an appropriate Sense Response is detected prior to the data transfer

```
         .
         .
         .
SEN     DA,FC
JMP     $-1
data transmission
         .
         .
         .
```

or:

```
         .
         .
         .
SSN     DA,FC
data transmission
```

In the first example, the Sense instruction is executed until a true Sense Response is detected and the Jump instruction is skipped. The data transmission is then performed. In the second example, the Sense instruction is executed only once. If a false Sense Response is detected, the data transmission instruction is skipped.

4.1.1.3 Data Transmission

Unconditional data transmission is accomplished using the Input-to-Register (INA and INX) and Output-from-Register (OTA, OTX and OTZ) instructions:

```
         .
         .
         .
SEN     DA,FC
JMP     $-1
INA    DA,FC
         .
         .
         .
```

or:

```
             .
             .
             .
        SEN   DA,FC
        JMP   $-1
        OTX   DA,FC
```

When the Sense Response is true, the Jump instruction is skipped and the data transmission instruction is executed.

In addition, the Sense operations can be combined with data transmission using the Read-to-Register (RDA, RDX, RBA and RBX) and Write-from-Register (WRA, WRX and WRZ) instructions:

```
             .
             .
             .
        RBA   DA,FC
             .
             .
             .
```

or:

```
             .
             .
             .
        WRX   DA,FC
             .
             .
             .
```

These instructions are executed repeatedly until a true sense response is received. The data transmission then occurs and the next instruction in sequence is executed. The Sense and unconditional data transfer operations can be combined in a conditional data transfer instruction only when the function codes for the two operations are the same. The conditional data transmission instructions are interruptable.

Block data transmissions are performed using the Block-Input-to-Memory (BIN) and Block-Output-from-Memory (BOT) instructions:

```
             .
             .
             .
LDX     COUNT
BIN     DA,FC
DATA    Base Address - 1
             .
             .
             .


             .
             .
             .
LDX     COUNT
BOT     DA, FC
DATA    Base Address - 1
             .
             .
             .
```

These instructions are executed repeatedly, transmitting one word of data each time a true Sense Response is received, until all data has been transmitted. The data is transmitted in reverse order in order of decreasing addresses. The next instruction in sequence is then executed. The function code associated with these instructions is the same as the function code used by the incorporated Sense. The block data transmission instructions are not interruptable.

In-Line automatic data transmissions are performed using the Automatic-Input-to-Memory (AIN and AIB) and Automatic-Output-from-Memory (AOT and AOB) instructions:

```
                                 .
                                 .
                                 .
          SENSE         SEN     DA,FC
                        JMP     $-1
                        AIN     DA,FC
                        DATA    Negative Data Count (Word)
                        DATA    Base Address -1 (Word)
                        JMP     EOB
                        JMP     SENSE
                                 .
                                 .
                                 .
```

or:

```
                                 .
                                 .
                                 .
          SENSE         SEN     DA,FC
                        JMP     $-1
                        AOB     DA,FC
                        DATA    Negative Data Count (Byte)
                        DATA    Base Address -1 (Byte)
                        JMP     EOB
                        JMP     SENSE
                                 .
                                 .
                                 .
```

These instructions unconditionally transmit one word/byte of data each time they are executed and are therefore preceded by an appropriate sense command. In addition, the Base Address pointer and the Negative Data Count are incremented, with the Data Count eventually becoming zero and generating an exit to the End-of-Block processing routine (EOB). Automatic I/O instructions may be used under interrupts, in which case the sense instruction is not required and the exits are replaced by a return to the mainline program. A second interrupt to a different location is generated by the peripheral controller when the buffer is completely transferred.

4.1.2 Interrupt Operations

Interrupts constitute a means of reacting instantly to random, external stimuli without consuming valuable processing time in a continuous polling environment. Peripheral devices which are to be operated under interrupt control are assigned reserved memory locations anywhere in memory. These interrupt addresses are generated by the individual peripheral controllers and generally have jumper selectable locations within the first 512 locations in memory. Appendix B includes a table of standard interrupt address assignments. When an interrupt is recognized, the instruction at the associated interrupt location is executed. If the instruction does not modify the program counter, control is immediately restored to the main program. Otherwise, processing continues at the location specified by the new contents of the P register. Any of the instructions in the ALPHA's repertoire can be used in the reserved locations as interrupt instructions, but certain of them are more useful than others - IMS, JST and the Auto I/O instructions. Any memory reference instruction performing relative to P backwards addressing should not be used as an interrupt instruction (the instruction would reference the location one less than the location actually programmed - e.g., $-9 instead of $-8). Before a given peripheral device can be operated under interrupt control, the interrupts for that device must be enabled. This enables the device to generate an interrupt request when the associated event occurs. In addition, the CPU interrupts must be enabled. This is accomplished using the EIN instruction and allows the CPU to respond to the interrupt request of the peripheral device.

4.1.2.1 Non-Input/Output

The Increment-Memory-and-Skip-on-Zero instruction is used in interrupt programming as a counter or timer for external events. As interrupt instructions, increment results of zero do not generate skips. They generate instead a signal (called an Echo) to the peripheral interface which caused the interrupt. Usually this signal is used by the device to generate a second interrupt to another reserved location, at which a JST (Jump-and-Store) instruction to a counter/timer maintenance subroutine would be located.

The Jump-and Store instruction is used in interrupt programming as a means of transferring control to an interrupt subroutine in a manner such that return to the main program at the interrupted location can be accomplished upon completion of the operations required by the interrupt. JST is the only instruction which disables the CPU interrupts when it is used as an interrupt instruction. Before returning to the main program the CPU interrupts should be re-enabled.

4.1.2.2 Input/Output

The Automatic-Input-to-Memory (AIN and AIB) and Automatic-Output-from-Memory (AOT and AOB) instructions have been specifically designed as interrupt instructions. Used to transfer blocks of data between the computer memory and the peripheral devices, these instructions contain their own word/byte count and memory word/byte address. They do not affect the A and X registers, the OV indicator or the P register when transferring data as interrupt instructions. As each data word/byte is transmitted, the associated pointer and counter are automatically incremented.

4.1.2.3 Word and Block Interrupts

When either the IMS or the Automatic Input/Output instructions are used as interrupt instructions, increment results of zero (any memory location for IMS and the negative word/byte count for the Auto I/O instructions) produce "Echo" signals which are typically used by the various peripheral devices to generate End-of-Block interrupt requests to different reserved interrupt locations.

4.2 NON-INTERRUPT INPUT/OUTPUT EXAMPLES

The examples shown in Figures 4-1 through 4-5 are discussed in the paragraphs that follow.

```
LABEL      INSTRUCTION    OPERANDS    COMMENTS

Optional   SEL              4,4       Command Initialize Line Printer
           .
           .
           .
           LDA              CHAR      A = Char to Print
           SEN              4,1       Sense Line Printer Ready
           JMP              $-1       (not ready)
           OTA              4,1       Unconditionally output A
```

Figure 4-1. Initialization and Unconditional Output to Line Printer

```
LABEL       INSTRUCTION   OPERANDS    COMMENTS

Optional        .
                .
                .
Optional    SEN             7,3       Sense Teletype Ready
            JMP             $-1       (not ready)
            SEL             7,2       Command Step Read
            SEN             7,1       Sense Character Buffer Full
            JMP             $-1       (not full)
            INA             7,0       Unconditionally input character to A
                .
                .
                .
```

Figure 4-2. Unconditional Character Read from Teletype Paper Tape Reader

```
LABEL       INSTRUCTION   OPERANDS    COMMENTS

                .
                .
                .
Optional    SEL             4,4       Command Initialize Line Printer
                .
                .
                .
            LXP            :8C        Top of Form Character
            WRX             4,1       Output to Line Printer when Ready
```

Figure 4-3. Initialization and Conditional Control of Line Printer

```
LABEL       INSTRUCTION    OPERANDS    COMMENTS

            .
            .
            .
Optional    SEN              7,3       Sense Teletype Ready
            JMP              $-1       (not ready)
            .
            .
            .
            SEL              7,0       Set Auto-Echo
            .
            .
            .
            RBA              7,1       Input a Teletype Character to A
                                       When Ready
            LLA              8         Shift to most significant 8 bits
            RBA              7,1       Input another character to least
                                       significant 8 bits
            SEL              7,4       Turn Auto-Echo Off
            .
            .
            .
```

Figure 4-4. Conditional Input from Teletype Keyboard with Auto-Echo

```
LABEL       INSTRUCTION    OPERANDS    COMMENTS

            .
            .
            .
Optional    SEL              4,4       Command Initialize Line Printer
            .
            .
            .
            LDX              COUNT     X = Word Buffer Length
            BOT              4,1       Block Output to Line Printer
            DATA             BUF-1     Character Buffer Address Less One
            .
            .
            .
BUF         RES             COUNT      Data Buffer
```

Figure 4-5. Uninterruptable Block Output to Line Printer

```
LABEL       INSTRUCTION   OPERANDS   COMMENTS

            .
            .
            .
Optional    SEN           5,3        Sense Card Reader Ready
   .        JMP           $-1        (not ready)
   .        SEL           5,4        Command Initialize Card Reader
   .        SEL           5,3        Command Card Reader Read Card
LOOP        SEN           5,0        Sense Input Character Ready
   .        JMP           $-1        (not ready)
   .        AIB           5,0        Automatic Input Character to Buffer
            DATA          -80        Increment Counter
            BAC           BUF-1      Increment Byte Address
            JMP           $+2        Zero Counter Results - Exit
            JMP           LOOP       Loop on non-Zero Counter Results
            .
            .
            .
BUF         RES           40         80 Character (Byte) Data Buffer
```

Figure 4-6. Automatic Byte Input from Card Reader

4.2.1 Control Instructions

The SEL instruction is the most widely used control instruction for peripheral devices. It is used both for initializing the devices, as in Figures 4-1, 4-3, 4-5 and 4-6, and for causing the peripheral devices to perform specific functions, as in Figures 4-2, 4-4 and the second SEL instruction in Figure 4-6. Sometimes special characters are used for control functions (e.g., the Line Printer Top-of-Form character in Figure 4-3).

The SEN instruction is used to test whether the specified data source or destination in the addressed peripheral device is ready to transmit or receive data. Sometimes both the peripheral device and a particular buffer within the device must be ready for data transmission, as in Figures 4-2 and 4-6. In many cases, the Sense function can be incorporated into the Conditional I/O instructions, as in Figures 4-3 and 4-4.

4.2.2 Unconditional Instructions

Unconditional Input instructions consist of both word and byte instructions. While the Word input instructions replace all 16 bits of the register (Figure 4-2), the byte input instructions affect only the least significant 8 bits of the register. When byte-oriented peripheral devices are used, these instructions allow the programmer to pack the input data before storing it in memory.

The unconditional Output instructions are word-oriented instructions. Since byte-oriented peripheral devices accept only the least significant 8 bits of data output from a register, there is no need for byte Output instructions.

4.2.3. Conditional Instructions

The conditional I/O instructions incorporate both the Sense and the data transmission functions into one instruction. These instructions make sense, of course, only when the function codes for the Sense and data transmission operations are the same.

The conditional Input instructions consist of both word and byte instructions. While the word input instructions replace all 16 bits of the register, the byte input instructions affect only the least significant 8 bits of the register. When byte-oriented peripheral devices are used, these instructions allow the programmer to pack the input data before storing it in memory, as in Figure 4-4.

The conditional Output instructions are word-oriented instructions. Since byte-oriented peripheral devices accept only the least significant 8 bits of data output from a register, there is no need for byte output instructions.

Interrupts may be acknowledged during the execution of a conditional I/O instruction when the data source or destination in the addressed peripheral device generates false (not ready) sense responses.

4.2.4 Block Transfer Instructions

The Block Transfer instructions allow high-speed data transmissions between memory and peripheral devices. They essentially access each data buffer memory location by summing the contents of the X register and the data buffer pointer (buffer address - 1) in the second word of the instruction. Each time the addressed peripheral device generates a true sense response, data is transmitted and the X register is decremented. Thus, the data is transmitted from or to the end of the buffer (higher-order memory locations) first. The last word transmitted accesses the start (low-order memory location) of the buffer. Interrupts may be acknowledged only after the X register has been decremented to zero and the instruction has been completed - i.e., when all data words have been input or output.

These instructions access word memory operands only (see Figure 4-5). They do not affect the contents of the A register.

4.2.5 Automatic Transfer Instructions

Although the Automatic Transfer instructions have been designed specifically as interrupt instructions, they may also be used in non-interrupt, in-line programming. They are three-word instructions, with the second word containing the negative (two's complement) word or byte count and the third word containing a word or byte address point (buffer address - 1). Since they are unconditional transfer instructions, the specified data source or destination in the addressed peripheral device must generate true sense responses before data transmission occurs. Each data transmission increments both the data counter and the address pointer. Non-zero data counter increment results generate a one-word skip. Zero increment results cause the next instruction in sequence (the instruction after the address pointer which is skipped by non-zero increment results) to be executed (see Figure 4-6).

4.3 INTERRUPT STRUCTURE AND EXAMPLES

4.3.1 General Interrupt Handling

External interrupts cause the computer to execute one instruction outside of the main program. If the instruction does not modify the P register, the computer continues with the main program after executing the interrupt instruction. If the interrupt instruction modifies the P register (either a JST or JMP) the computer continues processing at the location specified by the new value in the P register.

If a peripheral device is to operate under interrupt control, reserved locations in memory are assigned to the device. The computer then executes the instruction at the reserved location when the peripheral device generates an interrupt to the computer. Each device may be assigned one or more reserved locations. For example, a device moving blocks of data to or from the computer may generate one interrupt for each word or byte of data moved and a second interrupt when the entire block of data has been moved. The interrupt for each word or byte would require one location and the interrupt indicating the end of the block of data would require another.

Before any interrupt can be recognized by the computer, several conditions must be met:

1. The interrupts must be enabled, in general. If any interrupts are to be recognized, the Enable Interrupts (EIN) instruction must be executed.

2. The specific peripheral device interrupt must be enabled. Specific interrupts are enabled by setting an interrupt enable flag in the peripheral device interface. Enable flags are generally set by executing a Select (SEL) instruction with a device address and function code specifying which interrupt is to be enabled. Using interrupt enable flags, the programmer can selectively enable and disable interrupts.

3. The interrupt condition must exist (i.e., the device must be ready to accept or transmit data). Many peripheral devices "remember" interrupt conditions generated prior to enabling the interrupt enable flags. Care should be taken to reset the peripheral device interrupts before enabling the enable flag so that false interrupts do not occur immediately after enabling the interrupts.

4. No higher priority interrupt must be waiting. Each peripheral interface or computer option has a definite priority assignment. Interrupts are processed by the computer in the order received, or according to priority if more than one interrupt is pending.

5. The computer must be in the RUN mode. Interrupts cannot be recognized when the computer is halted.

4.3.2 Examples of Initialization and Enabling Sequences

Initialization and interrupt enabling take place prior to the generation and use of the interrupts. The examples below involving a line printer and the Real-Time Clock are typical of initialization sequences.

```
      .
      .
      .
      SEN          4,1   Wait for Line Printer Buffer ready
      JMP          $-1   (not ready)
      SEL          4,7   Reset Interrupt Enable flags
      SEL          4,5   Enable Word Interrupt Enable flag
      SEL          4,6   Enable Echo/EOB Interrupt Enable flag
      EIN                Enable CPU interrupts
      .
      .
      .
```

Figure 4-7. Line Printer Interrupt Initialization Sequence

The interrupt enable flags may also be reset by the line printer initialization instruction SEL 4,4. Note that the Word interrupt enable flag is enabled before the Echo/EOB interrupt enable flag. When specific actions in a peripheral device are additionally required to generate interrupts (e.g., a card reader must read a card), the command (SEL) instruction causing the action must be executed before the interrupt can take place. The sequence in Example 4-7 is used in conjunction with an AOT or AOB instruction in the word interrupt location and a JST instruction to an End-of-Block routine at the Echo/EOB interrupt location.

```
        .
        .
        .
        SEL             8,3   Reset RTC Interrupt Enable flags
        SEL             8,2   Arm RTC Sync Interrupt Enable flag
        SEL             8,0   Enable RTC Time and Sync Interrupt Enable
                              flag
        EIN             Enable CPU Interrupts
        .
        .
        .
```

Figure 4-8. Real-Time Clock Interrupt Initialization Sequence

The interrupt enable flags may also be reset by the Real-Time Clock initialization instruction SEL 8,4. Note that the Sync interrupt enable flag is armed before the Time and Sync interrupt enable flags are enabled. This sequence is used in conjunction with an IMS instruction in the word interrupt location and a JST instruction to a Sync maintenance routine in the Echo/Sync interrupt location.

4.3.3 Examples of Interrupt Instructions

The contents of the interrupt locations associated with the above examples are illustrated below in Figures 4-9 and 4-10.

```
   Location      :42(Word)  AOB    4,1      Automatic Output Byte Instruction
                            DATA   -80      Negative Character Buffer Length
                                            (Byte Counter)
                            BAC    BUF-1    Byte Address Pointer (Start-1)
                            .
                            .
                            .
                 :46(EOB)   JST    SUB      Jump to End-of-Block Routine
                            .
                            .
                            .
 Main memory     SUB        ENT             Disable CPU Interrupts
                            .
                            .
                            .
```

Figure 4-9. Line Printer Interrupt Instructions

Since the byte counter and address pointer are modified during the data transmission, they must be preset each time a line of characters is to be printed prior to execution of the initialization sequence discussed in Sec. 4.3.1. When all the characters have been transferred, the instruction at location :46 is executed and control is transferred to the End-of-Block routine beginning at SUB. This routine might output a carriage-return character to cause the line to be printed, or perform any other line termination processing required. The last character of the buffer might be a carriage-return (see Line Printer Driver Documentation).

```
Location      :18(Time)  IMS    COUNT    Increment RTC counter COUNT

              :1A(Sync)  JST    SYNC     Transfer to Sync Subroutine,
                         .               Disable CPU Interrupts
                         .
                         .
                         .
Main memory   SYNC       ENT             Save Main Program Return Location
                         SIN    1        Inhibit Status (guarantee Word mode)
                                         to Save A Register
                         STA    ASAVE    Save A Register
                         SIA             Input Status to A Register
                         STA    STATUS   Save Status
                         STX    XSAVE    Save X Register
                         LAM    100      Reset
                         STA    COUNT    RTC counter COUNT
                         .
                         .
                         .               Perform specified Maintenance Function
                                         Function
                         .
                         .
                         .
                         LDX    XSAVE    Restore X Register
                         LDA    STATUS   Load Status into A Register
                         SOA             Restore Status
                         SIN    1        Inhibit Status (guarantee Word Mode)
                                         to Restore A Register
                         LDA    ASAVE    Restore A Register
                         EIN             Enable CPU Interrupts
                         RTN    SYNC     Return to Mainline Program
```

Figure 4-10. Real-Time Clock Interrupt Instructions

Each acknowledgement of a Time interrupt causes the RTC counter COUNT to be incremented. When COUNT is incremented to zero, recognition of the Sync interrupt (at location :1A) generates execution of the SYNC interrupt subroutine.

Interrupts are automatically disabled by execution of the JST instruction, but the addressing mode and the state of the overflow indicator are unchanged. Because the computer might be in the Byte addressing mode when the interrupt occurs, the Word mode is forced for one instruction so that the full 16-bit contents of the A register can be saved. When this is done, the computer status is input, which also sets the addressing mode to the Word mode and resets the overflow indicator. The Status and the contents of the X register are then saved. The Real-Time Clock counter COUNT is reset to a negative value as part of the required maintenance operations.

Restoration of the contents of the X register begins the exit sequence of the subroutine. The computer status is then restored and byte mode inhibited for one instruction to insure restoration of the full 16-bit contents of the A register. The interrupts are then re-enabled and the subroutine is exited prior to acknowledgement of any other interrupt (since the EIN instruction inhibits recognition of interrupts for the duration of the RTN SYNC instruction).

The save/restore sequences discussed here should be used at the beginning and end of any interrupt subroutine to which a JST instruction at an interrupt location refers. The Real-Time Clock counter COUNT should also be set to a negative value before the initialization sequence discussed in Sec. 4.3.1 is executed.

4.4 INTERRUPT LATENCY

Recognition of an interrupt request from a peripheral device by the computer is not always instantaneous. The conditions discussed below delay acknowledgement of interrupts.

4.4.1 Interrupt Service

Interrupt acknowledgement occurs "between" the execution of instructions - i.e., just after the completion of a given instruction. The conditional Input/Output instructions allow recognition of interrrupts before their completion as long as false (not ready) Sense responses are obtained from the specified data source or destination. After the interrupt is serviced, processing is resumed with the conditional Input/Output instruction. The Scan Memory (SCM) instruction similarly allows recognition of interrupts after each specified word or byte of memory is compared to the contents of the A register. If interrupts were off prior to issuing an instruction, the EIN delays recognition of any interrupt until after the next instruction in sequence is executed. This allows return from interrupt subroutines to the mainline program before acceptance of another interrupt. The Block Input/Output (BIN and BOT) instructions, the Double-Word Memory Reference instructions and all shift instructions must be completed before recognition of an interrupt may occur. Since their use in main-line programs may constitute non-trivial delays in the recognition of interrupts, the programmer should use such instructions with care. In addition, when Direct Memory Access (DMA) operations are in progress, recognition of interrupts is delayed for the duration of any concurrent data transmission.

4.4.2 Priority Resolution

Occasionally, multiple interrupt requests occur. When this happens, the interrupt having the highest priority is acknowledged first, then the next, and so forth down to the interrupt having the lowest priority. One instruction of the main-line program is executed between each recognition of an interrupt. The standard interrupt priorities are listed below in Figure 4-11.

ABSOLUTE PRIORITY

1 POWER FAIL

2 TRAP

3 CONSOLE

4 MEMORY PARITY

5 INTERRUPT LINE 1 (IL1)

6 INTERRUPT LINE 2 (IL2)

7 RTC SYNC INTERRUPT (IUR)

8 RTC TIME INTERRUPT (IUR)

9 TTY END-OF-BLOCK INTERRUPTS (IUR)

10 TTY WORD INTERRUPTS (IUR)

11 SLOT B200

12 SLOT B100

IUR CHAIN

13 SLOT C100

14 SLOT C200

15 SLOT D200

16 SLOT D100

17 SLOT E100

18 SLOT E200

19 EXPANSION CHASSIS SLOT A100

20 EXPANSION CHASSIS SLOT A200

21 EXPANSION CHASSIS SLOT B200

.
.
.

Slots B200 through E200 accommodate plug-in modules (either memory or I/O). All I/O modules may use the IUR line and must provide an interrupt address. Modules with multiple interrupt capabilities must have internal priority resolution and multiple addresses.

The continuity of the priority chain must not be broken. If broken, interrupts below the break may not be recognized or may be recognized erroneously.

Figure 4-11. Standard Interrupt Priorities

Section 5
PROCESSOR OPTIONS

5.1 TELETYPE

5.1.1 General Discussion

The Teletype (TTY) option interfaces a modified ASR-33 or ASR-35 to the ALPHA computer. It performs all of the data and control signal conversion required for the computer to control the TTY. An ASR-33 or ASR-35 Teletype provides four Input/Output features in one package: keyboard input, page printer, paper tape reader and paper tape punch.

The interface contains a data buffer register which performs parallel-to-serial data conversion for transferring data from the computer to the Teletype and serial-to-parallel conversion when transferring data from the Teletype to the computer. In addition, the interface has provisions for interrupt generation for both Word and Echo/End-of-Block interrupts.

The TTY Interface option has been assigned a standard device address of 7.

Output from the computer is printed on the page printer. If the punch is turned on, the output is also punched. The punch and page printer cannot be separately controlled by the computer. The operator must turn the punch on or off as desired.

Input to the computer is accomplished via the TTY keyboard and the paper tape reader. They are controllable separately from the computer. The paper tape reader can read bytes one at a time or continuously. Automatic Echo is a feature which allows any input to be echoed back to the TTY for printing.

The Teletype can be operated in either half-duplex or full-duplex mode. The TTY Initialize instruction (SEL 7,4) puts the Teletype interface in the half-duplex mode. Execution of the Select-and-Present instructions SEA 7,4 or SEX 7,4 with the register contents equal to 1 puts the Teletype interface in the full-duplex mode.

5.1.2 Half-Duplex Usage

Half-duplex Teletype operations involve either input from or output to the Teletype, but not simultaneously. Use of the Auto Echo feature causes input from the Teletype to be automatically "echoed" back to the Teletype for printing, thus eliminating the necessity for echoing characters back under software control.

The following are examples of typical half-duplex teletype I/O sequences:

```
        SBM              Set Byte Addressing Mode
        SEL     7,4      Initialize TTY Interface
LOOP    LDAB    *DATA    Load Byte/Character into LSB Byte
                         of A register
        IMS     DATA     Increment Byte Address Pointer
        WRA     7,1      Output Byte when TTY is Ready
        IMS     COUNT    Increment Negative Number of Characters
                         to be Transferred
        JMP     LOOP     Continue Data Output if Non-zero Increment
                         Results
        .
        .
        .                Exit
```

Figure 5-1. Program-Controlled Data Output to Half-Duplex Teletype

```
        SBM              Set Byte Addressing Mode
        SEL     7,0      Enable Auto Echo to print data being input
        SEL     7,3      Start the paper tape reader in a continuous
                         read mode.
LOOP    RBA     7,1      Input Byte when TTY is Ready
        STAB    *DATA    Store Character in Data Buffer in Memory
        IMS     DATA     Increment Byte Address Pointer
        IMS     COUNT    Increment Negative Number of Characters to
                         be Transferred
        JMP     LOOP     Continue Data Input if Non-zero Increment
                         Results
        SEL     7,4      Initialize the TTY Interface to Stop the Paper
                         Tape Reader and Disable the Auto Echo
        .
        .
        .
```

Figure 5-2. Program-Controlled Data Input from TTY Paper Tape Reader

The standard Word Interrupt Location for half-duplex operation of the Teletype is location : 0002. The Teletype Interface interrupts to this location when the Word Transfer Mask is set, interrupts are enabled and the Teletype is ready for either input or output. A jumper option allows this interrupt location to be relocated to location : 0022. The standard End-of-Block Interrupt Location for half-duplex operation of the Teletype is location : 0006. The Teletype Interface interrupts to this location when the Block Transfer Mask is set, interrupts are enabled and an Echo signal (from completion of an Auto I/O Interrupt Sequence) is received from the computer. A jumper option allows this interrupt location to be relocated to location : 0026. An additional jumper option allows Processor mounted option interrupts to be offset by : 0100 locations, the standard half-duplex Teletype interrupts can be relocated to locations : 0102 and : 0106 or : 0122 and : 0126.

5.1.3 Table of Half-Duplex Teletype Instructions

Instruction	Function
SEL 7,0	Enable Auto Echo. This instruction causes all input from the TTY keyboard or paper tape reader to be echoed back to the TTY for printing.
SEL 7,1	Select Keyboard. This instruction resets the Buffer Ready sense and puts the teletype interface in the read mode.
SEL 7,2	Step Read. This instruction causes the character under the read station on the paper tape reader to be read. The tape is then advanced one character. The reader switch on the Teletype must be in the RUN psoition. The Buffer Ready sense is reset.
SEL 7,3	Select Continuous Read. This instruction causes the paper tape reader to read continuously until the reader is stopped or the tape runs out. The reader switch must be in the RUN position. The Buffer Ready flip-flop is reset. Execution of any other SEL instruction resets Continuous Read.
SEL 7,4	Initialize Teletype Interface. This instruction resets the control flip-flops, stops the oscillator and puts the interface in a static marking condition. The Buffer Ready sense is set and the half-duplex mode is entered.
SEL 7,5	Enable Word Transfer Interrupt. This instruction sets an enable flip-flop in the interface to enable generation of interrupts by the Buffer Ready sense (becoming true).
SEL 7,6	Enable Echo/EOB Interrupt. This instruction sets an enable flip-flop in the interface to allow generation of an EOB interrupt when an Echo signal is received. (Must be used following SEL 7,5 or an EOB interrupt will be generated immediately.)

Instruction	Function
SEL 7,7	Disable Interrupts. This instruction disables both the Word Transfer and Echo/EOB interrupts in the Teletype interface by resetting the enable flip-flops.
SEN 7,1	Sense Buffer Ready. This instruction senses the state of the Buffer Ready flip-flop and generates a one-word skip if it is set (true).
SEN 7,2	Sense Word Transfer Interrupt Enabled. This instruction senses the state of the Word Transfer Interrupt Enable flip-flop and generates a one-word skip if it is set.
SEN 7,3	Sense TTY Not Busy. This instruction senses the state of the TTY controller and generates a one-word skip if it is not printing, punching or reading a character.
SEN 7,5	Sense Teletype Motor On. This instruction senses the state of the Motor On flip-flop and generates a one-word skip if it is set (on).
SEN 7,6	Sense No Parity/Framing Error. This instruction senses whether a parity or framing error occurred during the most recent input operation and generates a one-word skip if no error occurred.
OTZ 7,6	Turn Motor On. This instruction sets the Motor On flip-flop which turns the TTY motor on. This instruction can be used only with Teletype units that have been modified for remote motor on/off control. Turning the motor on generates a 600 millisecond delay in all interrupts and ready senses to allow the motor to come up to speed.
OTZ 7,7	Turn Motor Off. This instruction resets the Motor On flip-flop in the interface, which turns the Teletype motor off.
OTA 7,0 OTX 7,0	Output A or X register to TTY. These instructions unconditionally transfer the contents of the specified register to the Teletype interface, which causes the character to be printed and punched (if the punch is on).
INA 7,0 INX 7,0	Input Word from TTY to A or X register. These instructions unconditionally transfer the character in the Teletype interface buffer to the specified register. The data is placed in the least significant byte of the specified register and the MSB byte is set to all zeros.

Instruction Function

AIN 7,0
AIB 7,0

Input Word/Byte from TTY to Memory Automatically. These instructions unconditionally transfer the character in the Teletype interface register to memory, automatically. The AIN instruction causes the TTY character to be loaded into the LSB byte of the memory location and forces the MSB byte to zero. The AIB instruction causes the TTY data to be packed two bytes per word of memory.

BIN 7,1

Input Block from TTY to Memory. This instruction senses the state of the Buffer Ready flip-flop and transfers the character in the Teletype interface register to memory when Buffer Ready is true and decrements a word count after each transfer. When the word count reaches zero, the instruction terminates.

IBA 7,0
IBX 7,0

Input Byte from TTY to A or X register. These instructions unconditionally transfer the character in the Teletype interface buffer to the specified register. They do not affect the MSB byte of the specified register.

RBA 7,1
RBX 7,1

Read Byte from TTY to A or X register. These instructions sense the state of the Buffer Ready flip-flop and transfer the character in the Teletype interface buffer to the specified register when it is set (true). They do not affect the MSB byte of the specified register.

WRA 7,1
WRX 7,1

Write from A or X register to TTY. These instructions sense the state of the Buffer Ready flip-flop and transfer the character in the specified register to the Teletype interface buffer when it is set (true).

AOT 7,0
AOB 7,0

Output Word/Byte from Memory to TTY, Automatically. These instructions unconditionally transfer a word or byte from memory to the Teletype, automatically. In the case of the AOT instruction, the Teletype uses the LSB byte of the word and ignores the MSB byte. The AOB instruction automatically unpacks each byte during subsequent transfers.

BOT 7,1

Output Block from Memory to TTY. This instruction senses the state of the Buffer Ready flip-flop and transfers the full 16-bit memory word to the Teletype interface register when Buffer Ready is true and decrements a word count after each transfer. When the word count reaches zero, the instruction is terminated.

5.1.4 Full-Duplex Usage

Full-duplex Teletype operations allow simultaneous input and output. The interface contains two data buffers in this mode - one for input and one for output. Use of the Auto Echo feature causes input from the Teletype to be automatically "echoed" back to the Teletype for printing, thus eliminating the necessity for echoing characters back under software control. When this feature is used, normal output data and echoed data can be intermixed but care should be taken to assure that the resulting sequence of output characters makes sense.

Full-duplex Teletype operation also allows use of a special "loop-back" diagnostic feature. This mode is entered by executing the Select-and-Present instructions SEA 7,4 or SEX 7,4 with the register contents equal to 3. This feature connects the Output Data Buffer to the Input Data Buffer, allowing immediate comparison of transmitted data and received data.

The following are examples of typical full-duplex Teletype I/O sequences:

```
.
.
.
SBM             Set Byte Addressing Mode
ARP             Set A register to Plus One
SEA     7,4     Initialize TTY Interface to Full-Duplex
SEL     7,1     Select Keyboard Mode
RBA     7,0     Output Character When Ready
WRA     7,1     Output Character Just Input
.
.               Process Character
.
```

Figure 5-3. Program-Controlled Data Input from Full-Duplex Teletype

The above example is somewhat inefficient in that it does not use the Auto Echo feature to print the data being input. It is used here primarily to illustrate the different function codes involved with data input and data output.

```
 Location    :0022   AIB     7,0      Automatic Byte Input Instruction
                     DATA    -10      Negative Byte/Character Count
                     BAC     BUF-1    Buffer Address Pointer (Start-1)

             :0026   JST     END      Echo/EOB Termination
                     .
                     .
                     .
Main memory          ARP              Set A Register to Plus One
                     SEA     7,4      Initialize TTY to Full-Duplex Mode
                     SEL     7,1      Select Keyboard Mode
                     SEL     7,0      Enable Auto Echo
                     SEA     7,5      Enable Input Word Transfer Interrupt
                     SEA     7,6      Enable Input Echo/EOB Interrupt
                     EIN              Enable CPU Interrupts
                     WAIT             Wait for Echo/EOB interrupt
             END     ENT              Entry Point for End-of-Block Processing
                     ARP
                     SEA     7,7      Disable TTY Interrupts
                     .
                     .
                     .
```

Figure 5-4. Automatic Interrupt Data Input from Full-Duplex Teletype

Initialization of the interrupt locations is not shown in the above sequence. The example transfers ten characters input from the Teletype keyboard to a character buffer in memory. When the tenth character is input from the keyboard, the counter in the Auto I/O instruction at the Input Word Transfer interrupt location is incremented to zero and an Echo signal is transmitted to the Teletype interface. An EOB interrupt is then generated and control is transferred to the program sequence beginning at location END.

For full-duplex operation of the Teletype, the following standard and offset interrupt locations are provided:

	Standard Location	Offset Location	Priority
Output Word Transfer Interrupt	:0002	:0102	3
Output Echo/EOB Interrupt	:0006	:0106	1
Input Word Transfer Interrupt	:0022	:0122	4
Input Echo/EOB Interrupt	:0026	:0126	2

The jumper option for relocation to locations :0022 and :0026 (or :0122 and :0126) in the half-duplex has no affect on the interrupt locations for full-duplex operation. Note that the EOB interrupts have priority over the word interrupts.

5.1.5 Table of Full-Duplex Teletype Instructions

Instruction	Function
SEL 7,0	Enable Auto Echo. This instruction causes all input from the TTY keyboard or paper tape reader to be echoed back to the TTY for printing.
SEL 7,1	Select Keyboard. This instruction resets the input buffer full flip-flop and puts the Teletype interface in the read mode.
SEL 7,2	Step Read. This instruction causes the character under the read station on the paper tape reader to be read into the input buffer. The tape is then advanced one character. The reader switch on the Teletype must be in the RUN position. The Input Buffer Full flip-flop is set.
SEL 7,3	Select Continuous Read. This instruction causes the paper tape reader to read continuously until the reader is stopped or the tape runs out. The reader switch must be in the RUN position. The Input Buffer Full flip-flop is reset. Execution of any other Select instruction resets Continuous Read.
SEL 7,4	Initialize Teletype Interface to Half-Duplex. This instruction resets all controls and places the Teletype in the half-duplex mode. The Buffer Ready flip-flop is set.
SEA 7,4 SEX 7,4 (A or X = 1)	Initialize Teletype Interface to Full-Duplex. These instructions reset all controls and place the Teletype in the full-duplex mode. The Input Buffer Full flip-flop is reset and the Output Buffer Empty flip-flop is set.
SEA 7,4 SEX 7,4 (A or X = 3)	Initialize Teletype Interface to Full-Duplex Diagnostic. These instructions reset all controls and place the teletype in the full-duplex mode. In addition, the Output Data Buffer is connected to the Input Data Buffer. The Input Buffer Full flip-flop is reset and the Output Buffer Empty flip-flop is set. Any character output by the program will be received by the TTY input buffer.
SEL 7,5	Enable Output Word Transfer Interrupt. This instruction sets an enable flip-flop in the interface to allow generation of interrupts by the Output Buffer Empty flip-flop being set.
SEA 7,5 SEX 7,5 (A or X = 1)	Enable Input Word Transfer Interrupt. These instructions set an enable flip-flop in the interface to allow generation of interrupts by the Input Buffer Full flip-flop being set.

Instruction   Function

SEL 7,6   Enable Output Echo/EOB Interrupt. This instruction sets an enable flip-flop in the interface to allow generation of the Output EOB interrupt when an Echo signal generated as the result of an Output Word Transfer interrupt is received from the computer (Must be set after SEL 7,5 or an EOB interrupt will be generated immediately.)

SEA 7,6
SEX 7,6
(A or X = 1)   Enable Input Echo/EOB Interrupt. These instructions set an enable flip-flop in the interface to allow generation of an Input EOB interrupt when an Echo signal generated as the result of an Input Word Transfer interrupt is received from the computer (must be set after SEA/SEX 7,5 or an interrupt will be generated immediately.)

SEL 7,7   Disable Output Word Transfer and Echo/EOB Interrupts. This instruction disables the two Output interrupts in the Teletype interface by resetting the corresponding enable senses.

SEA 7,7
SEX 7,7
(A or X = 1)   Disable All Word Transfer and Echo/EOB Interrupts. These instructions disable both the Input and the Output Word Transfer and interrupts in the Teletype interface by resetting all four enable senses.

SEN 7,0   Sense Input Buffer Full. This instruction senses the state of the Input Buffer Full flip-flop and generates a one-word skip if it is set (true).

SEN 7,1   Sense Output Buffer Empty. This instruction senses the state of the Output Buffer Empty flip-flop and generates a one-word skip if it is set (true).

SEN 7,2   Sense Output Word Transfer Interrupt Enabled. This instruction senses the state of the Output Word Transfer Interrupt Enable flip-flop and generates a one-word skip if it is set.

SEN 7,3   Sense Teletype Not Busy. This instruction senses the state of the Teletype controller and generates a one-word skip if it is not printing, punching or reading a character.

SEN 7,5   Sense Teletype Motor On. This instruction senses the state of the Motor On flip-flop and generates a one-word skip if it is set (on).

SEN 7,6   Sense No Parity/Framing Error. This instruction senses whether a parity or framing error occurred during the most recent input operation and generates a one-word skip if no error occurred.

SEN 7,7   Sense Input Word Transfer Interrupt Enabled. This instruction senses the state of the Input Word Transfer Interrupt Enable flip-flop and generates a one-word skip if it is set.

Instruction	Function
OTZ 7,6	Turn Motor On. This instruction sets the Motor On flip-flop in the interface which turns the Teletype motor on. This instruction can be used only with Teletype units that have been modified for remote motor on/off control. Turning the motor on generates a 600 millisecond delay in all interrupts and ready senses to enable the motor to come up to speed.
OTZ 7,7	Turn Motor Off. This instruction resets the Motor On flip-flop in the interface which turns the Teletype motor off.
OTA 7,1 OTX 7,1	Output A or X Register to TTY. These instructions unconditionally transfer the contents of the specified register to the Output Data Buffer, which causes the character to be printed and (if the punch is on) punched.
INA 7,0 INX 7,0	Input Word from TTY to A or X Register. These instructions unconditionally transfer the character in the Teletype interface buffer to the specified register. The data is placed in the least significant byte of the specified register and the MSB byte is set to all zeros.
AIN 7,0 AIB 7,0	Input Word/Byte from TTY to Memory Automatically. These instructions unconditionally transfer the character in the Teletype interface register to memory, automatically. The AIN instruction causes the TTY character to be loaded into the LSB byte of the memory location and forces the MSB byte to zero. The AIB instruction causes the TTY data to be packed two bytes per word of memory.
BIN 7,0	Input Block from TTY to Memory. This instruction senses the state of the Input Buffer Full flip-flop and transfers the character in the Teletype interface register to memory when Input Buffer Full is true and decrements a word count after each transfer. When the word count reaches zero, the instruction terminates.
IBA 7,0 IBX 7,0	Input Byte from TTY to A or X Register. These instructions unconditionally transfer the contents of the Input Data Buffer to the specified register. The MSB byte of the specified register is not affected.
RBA 7,0 RBX 7,0	Read Byte from TTY to A or X Register. These instructions sense the state of the Input Buffer Full flip-flop and transfer the character in the Input Data Buffer to the specified register when the flip-flop is set (true). The MSB byte of the specified register is not affected.
WRA 7,1 WRX 7,1	Write from A or X Register to TTY. These instructions sense the state of the Output Buffer Empty flip-flop and transfer the contents of the specified register to the Output Data Buffer when the flip-flop is set (true).

Instruction   Function

AOT 7,0
AOB 7,0   Output Word/Byte from Memory to TTY, Automatically. These instructions unconditionally transfer a word or byte from memory to the Teletype, automatically. In the case of the AOT instruction, the Teletype uses the LSB byte of the word and ignores the MSB byte. The AOB instruction automatically unpacks each byte during subsequent transfers.

BOT 7,1   Output Block from Memory to TTY. This instruction senses the state of the Output Buffer Empty flip-flop and transfers the full 16-bit memory word to the Teletype interface register when Output Buffer Empty is true and decrements a word count after each transfer. When the word count reaches zero, the instruction is terminated.

5.2 REAL-TIME CLOCK

5.2.1 Discussion of Usage

The Real-Time Clock (RTC) option provides a means of determining elapsed time and/or creating a time-of-day clock with software. The RTC keeps time by responding to electrical pulses of known frequency, such as the output of a crystal oscillator or the input frequency of an AC power source. The standard configuration uses a 20 MHz crystal oscillator as the basic timing source. The 20 MHz clock is applied to a counter chain to produce 10 kHz, 1 kHz and 100 Hz clock sources (timing increments of 100 microseconds, 1 millisecond and 10 milliseconds, respectively). In addition, a 120 Hz clock source is available (100 Hz when the computer is used with 50 Hz power source). The desired clock source is selected by a jumper wire. An external timing source may be applied to the RTC option if some source other than the crystal oscillator or twice the AC line frequency is desired. This allows the use of almost any timing period. The default selection if no jumper is installed is the 100 Hz clock source.

If RTC interrupts are enabled, the RTC generates a time interrupt to the computer each time a clock pulse is detected from the clock source. This interrupt is usually serviced by an IMS instruction at the interrupt location. Increment results of zero cause the generation of an Echo signal to the RTC, which in turn generates a Sync interrupt to the computer. The Sync interrupt is normally serviced by a JST instruction to an interrupt subroutine. The RTC has been assigned a device address of 8.

In the following examples, an external device must be sampled once a second, using a 10 millisecond clock source:

```
Time Interrupt Location             IMS   COUNT   Increment timing counter
(: 0018 or 0118 if offset)

Sync Interrupt Location             JST   SYNC    Jump-and-Store to interrupt
(: 001A or : 011A if offset)                      subroutine, disable interrupts.

Initialization                      .
                                    .
                                    .
                            INIT    LAM   100     Set timing count to -100.
                                    STA   COUNT
                                    SEL   8,4     Initialize RTC and clear
                                                  unserviced interrupt requests.
                                    SEL   8,2     Arm sync-allow sync interrupts
                                                  when Echo is received.
                                    SEL   8,0     Enable RTC-allow generation of
                                    .             Time and Sync interrupts (since
                                    .             Sync is armed).
                                    .

Interrupt Subroutine:

                            SYNC    ENT           Reserved location for storage
                                    .             of P register.
                                    .
                                    .             Save contents of registers,
                                                  status, etc(see Sec. 4.3)
                                    .
                                    LAM   100     Reset Timing counter to -100.
                                    STA   COUNT
                                    .
                                    .
                                    .             Sample external device.
                                    .             Restore registers and status
                                    .             (see Sec. 4.3).
                                    .
                                    .
                                    EIN           Enable interrupts.
                                    RTN   SYNC    Return to mainline program.
```

The timing counter COUNT becomes zero after being incremented 100 times - after 100 Time interrupts, each 10 milliseconds apart. The RTC responds to the resulting Echo signal by generating a Sync interrupt which is serviced by the interrupt subroutine SYNC. The timing counter COUNT is reset to -100 and the external device is sampled.

5.2.2 Summary Table

Time Interrupt Location: :0018 (offset = :0118)
Sync Interrupt Location: :001A (offset = :011A)

Instruction	Function
SEL 8,0	Enable RTC. Allows Time and Sync interrupts to be generated (if Sync is armed).
SEL 8,2	Arm Sync. Allows generation of Sync interrupts if RTC is enabled and Echo received.
SEL 8,3	Clear RTC interrupts. Resets both Time and Sync interrupt requests. Does not disable or disarm interrupts, but instead removes interrupt request history from RTC.
SEL 8,4	Initialize RTC. Disarms, disables, and clears interrupt requests.
SEL 8,7	Disarm Sync. Prevents Sync interrupts from being generated without disabling Time interrupts.

5.3 AUTOLOAD

The Autoload (AL) option consists of a read only memory (ROM) preprogrammed with a binary loader and the necessary logic to execute the loader. The Autoload program is a complete binary program loader, not just a bootstrap. It includes appropriate input format and data error checking. Autoload uses locations :30 through :3B for scratch. No program occupying these locations can be loaded correctly with Autoload.

The Autoload option is a multi-device loader which reads programs in standard binary format and stores them in the computer memory. Autoload may read programs from a TTY paper tape reader, high speed paper tape reader, 9-track magnetic tape unit or cassette tape unit.

Device selection, and operation performed, is controlled by entering the appropriate hex code into the Sense Register prior to depressing the AUTO button. The options available are:

	TTY	HSPT	Mag Tape	Cassette
Load Absolute	:0	:1	:2	:3
Load Relocatable	:8	:9	:A	:B
Verify Absolute	:4	:5	:6	:7
Verify Relocatable	:C	:D	:E	:F

The Autoload logic causes all instruction cycles to fetch instructions from the ROM and all data cycles to access memory. Thus, the loader in the ROM is executed and the program being read from the peripheral device is treated as data which is stored in memory.

The presence of the Autoload option can be sensed using device address 0 and function code 0. This instruction is used primarily by diagnostic and executive programs. The Sense instruction takes the following form:

```
SEN   0,0        Sense Autoload not installed.
```

A true response causes a skip to occur (when Autoload is not installed).

5.4 POWER FAIL/RESTART

5.4.1 General

Power Fail/Restart (PFR) is an optional feature of the ALPHA computer. It allows the computer to operate from unreliable AC power sources without the requirement of human monitors. A low power condition or a temporary power outage is detected in time for the operating program to prepare for the power loss. When power returns to normal, the computer is automatically restarted without loss of data or operating position. Thus, unattended operation is possible.

5.4.2 Power Fail

When a power failure is detected, a power fail interrupt is generated to the Processor. If the Power Fail interrupt is enabled, the Processor is interrupted to a reserved location in memory (location : 001C or : 011C if offset). The Processor executes the instruction (usually a JST to a software power-down routine) at that location. The Processor has 0.9 milliseconds to complete the power-down routine once the PFR Down Sequence is started before the PFR option halts the computer and protects memory from transient power conditions.

5.4.3 Restart

When PFR detects power restoration to an acceptable level, a Power Up sequence is started. PFR re-enables memory, sets the P register to : 0000 and generates a run signal to the computer. The computer then executes the instruction (normally a JMP to a software power-up routine) at location : 0000. The computer always undergoes this sequence when power is applied. The software power-up routine must be completed within 0.9 milliseconds to allow enough time to process a Power Down interrupt if one should occur immediately after Power Up.

CAUTION

When the Power Fail/Restart option is installed, the computer will start running at location :0000 when power is applied whether the computer was running or not (i.e., independent of prior console setting) prior to removal of power. To avoid false starts it is customary for the Power Down subroutine to save a flag indicating that the computer was in fact running before power faded.

5.4.4 Interrupt Control Option

A hardware wiring option may place the Power Fail interrupt outside EIN/DIN control. Under this option, it is necessary to execute the PFE or PFD instructions to enable or disable the Power Fail interrupt. Without the option, the EIN or DIN instructions must be executed and PPE and PFD have no effect.

5.4.5 Programming Examples

The following are examples of simple Power Fail/Restart software routines:

Interrupt Location :0000	JMP	UP	This is the Power Up restart location. It contains an unconditional Jump to the Power Up subroutine.
Interrupt Location :001C (or :011C if offset)	JST	DOWN	This is the Power Down interrupt location. It contains a Jump and Store to the Power Down subroutine. Using a JST automatically saves the contents of the P register.
DOWN	ENT		Reserved location for storage of the P register when the JST instruction at the Power Fail interrupt location is executed.
	SIN	1	Inhibit the Byte Mode if it is set.
	STA	ASAVE	Save the A register
	SIA		Read the computer status word to the A register, set the Word Mode and reset the OV indicator.

```
        STA   STATUS   Save the computer status word.

        STX   XSAVE    Save the X register.

        IMS   PSTP     Save a flag indicating that the
                       computer was stopped by a
                       power failure.

UP      ZAR            The JMP instruction at the
                       Power Up restart location
                       enters here.

        EMA   PSTP     Check flag to see if computer
                       was stopped by a power failure.

        JAN   $+2

        HLT            No - do not restart.

        LDX   XSAVE    This instruction restores the
                       X register.

        LDA   STATUS   Load the computer status into
        SOA            the A register, then set the
                       computer status (Sense Switch,
                       Data Switches, OV indicator
                       and Address Mode).

        SIN   1        Inhibit the Byte Mode if it is
                       set.

        LDA   ASAVE    Restore the A register.

        EIN            Enable interrupts (including
                       Power Fail).

        JMP   *DOWN    Restart the main program by
                       executing an indirect Jump to
                       the location specified by the
                       saved contents of the P register.

ASAVE   DATA 0         A register save location.

XSAVE   DATA 0         X register save location

STATUS  DATA 0         Computer status word save
                       location.
```

```
                    PSTP     DATA 0          Flag indicating Processor
                                             was stopped by a power
                                             failure.
```

In these examples the contents of the A and X registers, the computer status and the mainline program location at the time of the Power Fail interrupt are saved during the Power Down sequence and restored during the Power Up sequence. Note that the Power Fail interrupt is under EIN/DIN control in this example. If the Power Fail interrupt were outside EIN/DIN control, the UP routine would have to include a PFE instruction just prior to the EIN instruction.

Appendix A
HEXADECIMAL TABLES

A.1 GENERAL

Table A-1 and A-2 are quick reference conversion tables that have been included for the convenience of the user.

Table A-1. Hexadecimal-Decimal Conversions

This table is designed to facilitate conversion of positive hexadecimal integers in standard single-precision or double-precision format to decimal equivalents. The fourth and eighth digit positions therefore contain only values in the range : 0 through : 7.

	DECIMAL EQUIVALENTS							
HEXADECIMAL	DIGIT 8	DIGIT 7	DIGIT 6	DIGIT 5	DIGIT 4	DIGIT 3	DIGIT 2	DIGIT 1
1	134217728	8388608	524288	32768	4096	256	16	1
2	268435456	16777216	1048576	65536	8192	512	32	2
3	402653184	25165814	1572864	98304	12288	768	48	3
4	536870912	33554432	2097152	131071	16384	1024	64	4
5	671088640	41943040	2621440	163840	20480	1280	80	5
6	805306368	50331648	3145728	196608	24576	1536	96	6
7	939524096	28720256	3670016	229376	28672	1792	112	7
8		67108864	4194304	262144		2048	128	8
9		75497472	4718592	294912		2304	144	9
A		83886080	5242880	327680		2560	160	10
B		92274688	5767168	360448		2816	176	11
C		100663296	6291456	393216		3072	192	12
D		109051904	6815744	425984		3328	208	13
E		117440512	7340032	458752		3584	224	14
F		125829120	7864320	491520		3840	240	15

Hexadecimal to decimal conversion is accomplished by summing the decimal equivalents of the hexadecimal digits. Decimal to hexadecimal conversion involves locating the next lower decimal number and its hexadecimal equivalent and then taking the difference. Each difference is treated similarly until the entire hexadecimal number is developed.

Table A-2. ASCII Teletype Codes

Symbol	Hexadecimal Code	Symbol	Hexadecimal Code
@	C0	b̸	A0
A	C1	!	A1
B	C2	"	A2
C	C3	#	A3
D	C4	$	A4
E	C5	%	A5
F	C6	&	A6
G	C7	'	A7
H	C8	(	A8
I	C9	)	A9
J	CA	*	AA
K	CB	+	AB
L	CC	,	AC
M	CD	-	AD
N	CE	.	AE
O	CF	/	AF
P	D0	0	B0
Q	D1	1	B1
R	D2	2	B2
S	D3	3	B3
T	D4	4	B4
U	D5	5	B5
V	D6	6	B6
W	D7	7	B7
X	D8	8	B8
Y	D9	9	B9
Z	DA	:	BA
[	DB	;	BB
\	DC	<	BC
]	DD	=	BD
↑	DE	>	BE
←	DF	?	BF
NULL	00	CR	8D
BELL	87	LF	8A
		RUBOUT	FF

Appendix B

RECOMMENDED DEVICE AND INTERRUPT ADDRESSES

B.1 GENERAL

Table B-1 and B-2 list recommended Device and Interrupt Addresses to prevent possible conflict during future expansion to other I/O modules.

Table B-1. **Recommended Device Addresses**

DEVICE	DEVICE ADDRESSES	
	STANDARD	ACTUAL
Power Fail Restart*	00	
	01	
Dual TTY/CRT (TTY1/CRT1)	02	
Dual TTY/CRT (TTY0/CRT0)	03	
Line Printer (LP)	04	
Card Reader (CR)	05	
Paper Tape Punch(PTP)	06(17)	
Paper Tape Reader (PTR)	06	
Processor TTY* (TTY)	07	
Real Time Clock* (RTC)	08	
Magnetic Tape (Mag Tape)	09	
	0A	
	0B	
	0C	
	0D	
	0E	
Disc	0F	
Cassette	10	
	11	
16-Bit I/O (A/D System)	12	
	13	
Plotter	14	
	15	
32-Bit Relay In (RCIM)	16	
Punch Alternate	17	
16-Bit Input/Output (16-Bit I/O)	18	
64-Bit Input (64-Bit In)	19	
64-Bit Output (64-Bit Out)	1A	
Priority Interrupt Module (PIM)	1B	
32-Bit Relay Out (RCOM)	1C	
103 Data Set Controller (103 DSC)	1D	
	1E	
	1F	

*Processor mounted options. Device Addresses non-alterable.
( ) Indicates suggested alternate.

Table B-2. Scratchpad/Page 0 Recommended Interrupt Address Map

	00-1F	20-3F	40-5F	60-7F	80-9F	A0-BF	C0-DF	E0-FF	
0	:00* POWER UP	:20* 64-BIT OUT			:80 PIM(0)				0
1									1
2	:02 TTY WORD	:22 MAG TAPE WORD	:42 LP WORD	:62 TTY0/CRT0 WORD	:82 PIM(1)	:A2 PLOTTER WORD			2
3									3
4					:84 PIM(2)				4
5									5
6	:06 TTY EOB	:26 MAG TAPE EOB	:46 LP EOB	:66 TTY0/CRT0 EOB	:86 PIM (3)	:A6 PLOTTER EOB			6
7									7
8					:88 PIM(4)				8
9									9
A	:0A M.H. DISC	:2A PTR/PTP WORD	:4A CR WORD	:6A TTY1/CRT1 WORD	:8A PIM(5)	:AA RCIM WORD			A
B									B
C					:8C PIM(6)				C
D									D
E		:2E PTR/PTP EOB	:4E CR EOB	:6E TTY1/CRT1 EOB	:8E PIM(7)	:AE RCIM EOB			E
F									F
0	:10* 64-BIT IN	:30** 103 DSC ANSWER	:50 CASSETTE ADDRESS		:90 PIM(8)				0
1									1
2	:12 MEMORY PARITY	:32** 103 DSC INPUT WORD	:52 CASSETTE WORD	:72 16-BIT I/O WORD	:92 PIM(9)				2
3									3
4					:94 PIM(10)				4
5									5
6		:36** 103 DSC IN EOB	:56 CASSETTE EOP	:76 16-BIT I/O EOB	:96 PIM(11)				6
7									7
8	:18 RTC CLOCK	:38** 103 DSC PAR ERR/FLT			:98 PIM(12)				8
9									9
A	:1A RTC SYNC	:3A** 103 DSC OUTPUT WORD	:5A RCOM WORD	:7A 16-BIT I/O WORD	:9A PIM(13)				A
B									B
C	:1C POWER DOWN				:9C PIM(14)				C
D									D
E	:1E CONS INT & TRAP	:3E** 103 DSC OUT EOB	:5E RCOM EOB	:7E 16-BIT I/O EOB	:9E PIM(15)				E
F									F

XX = Interface Generated Interrupt Address

* = Non-Alterable Address

** = Locations 30-3F are reserved for Autoload option, if used (103 DSC addresses must be relocated.)

EOB = End-of-Block

EOP = End-of-Operation

Appendix C
INSTRUCTION SET BY CLASS

This appendix contains the ALPHA LSI instruction set in alphabetical order. For each instruction, reference is made to one of the assembler syntax formats listed below.

```
[LABEL]     OP-CODE         [* |@| *@] EXPRESSION      [COMMENTS]

                            No Operator = Direct Addressing
                                      * = Indirect Addressing (multi-level)
                                      @ = Indexed Addressing
                                     *@ = Indirect Post-Indexed Addressing
                                          (multi-level)
```

Figure C-1. Class 1 - Single-Word Memory Reference Instruction Format

```
[LABEL]     OP-CODE      [*] EXPRESSION 1, [EXPRESSION 2]  [COMMENTS]

                            EXPRESSION 1 represents an address to be
                            stored in the second word of the instruction.
                            EXPRESSION 2 is an optional absolute instruc-
                            tion count in the range 0 through 16 for MPY
                            and DVD and 0 through 31 for NRM.
```

Figure C-2. Class 2 - Double-Word Memory Reference Instruction Format

```
[LABEL]     OP-CODE         EXPRESSION                 [COMMENTS]

                            EXPRESSION must be absolute and in the range
                            : 0 through : FF. This format is also used by the
                            STOP and SCM instructions.
```

Figure C–3. Class 3 - Byte Immediate Instruction Format

LABEL	OP-CODE	EXPRESSION	COMMENTS
		EXPRESSION must represent a location within -63 through +64 words.	

Figure C-4. Class 4 - Conditional Jump Instruction Format

LABEL	OP-CODE	EXPRESSION	COMMENTS
		EXPRESSION must be absolute and in the range 1 through 8 (single register) or 1 through 16 (double register). This format is also used by the SIN instruction with an upper range limit of 6.	

Figure C-5. Class 5 - Register Shift Instruction Format

LABEL	OP-CODE		COMMENTS
		There are no expressions in the operand field.	

Figure C.6. Class 6 - Nonvariable Instruction Format

LABEL	OP-CODE	EXPRESSION 1 ,EXPRESSION 2	COMMENTS
		Both EXPRESSION 1 and EXPRESSION 2 must be absolute. If EXPRESSION 2 is present, EXPRESSION 1 must be in the range : 0 through : 1F. If EXPRESSION 2 is not present, EXPRESSION 1 must be in the range : 0 through : FF.	

Figure C-7. Class 7 - Input/Output Instruction Format

LABEL	JOC	EXPRESSION 1, EXPRESSION 2	COMMENTS
		EXPRESSION 1 must be absolute and in the range : 0 through : 3F. EXPRESSION 2 must represent a location with -63 through +64 words.	

Figure C-8. Class 8 - JOC Jump-On-Condition Instruction Format

LIST OF INSTRUCTIONS

MEMORY REFERENCE (Class 1)

		Instruction Code in Hex	Reference Page
Arithmetic			
ADD	Add to A register	8800	3-10
ADDB	Add Byte to A	8800	3-10
SUB	Subtract from A register	9000	3-10
SUBB	Subtract Byte from A	9000	3-10
Logic			
AND	AND to A	8000	3-11
ANDB	AND byte with A	8000	3-11
IOR	Inclusive OR to A	A000	3-11
IORB	Inclusive OR Byte with A	A000	3-11
XOR	Exclusive OR to A	A800	3-11
XORB	Exclusive OR Byte with A	A800	3-11
Data Transfer			
LDA	Load A	B000	3-12
LDAB	Load A with Byte	B000	3-12
LDX	Load X	E000	3-12
LDXB	Load X with Byte	E000	3-12
STA	Store A	9800	3-12
STAB	Store Byte from A	9800	3-12
STX	Store X	E800	3-12
STXB	Store Byte from X	E800	3-12
EMA	Exchange A and Memory	B800	3-12
EMAB	Exchange A and Memory Byte	B800	3-12
Program Transfer			
JMP	Unconditional Jump	F000	3-13
JST	Jump and Store P Counter	F800	3-13
IMS	Increment Memory, Skip on Zero	D800	3-13
SCM	Scan Memory	CD00	3-13
CMS	Compare A with Memory, skip (high, low, equal test)	D600	3-12
CMSB	Compare A with Memory Byte, skip (high, low, equal test)	D600	3-13

DOUBLE WORD MEMORY REFERENCE (Class 2)

		Instruction Code in Hex	Reference Page
DVD	Divide	1970	3-14
MPY	Multiply and Add	1960	3-15
NRM	Normative A and X	1940	3-16

BYTE IMMEDIATE (Class 3)

AXI	Add to X Register Immediate	C200	3-17
SXI	Subtract from X Register Immediate	C300	3-18
CAI	Compare to A Immediate, skip if not equal	C000	3-17
CXI	Compare to X Immediate, skip if not equal	C100	3-17
LAP	Load A Positive Immediate	C600	3-17
LXP	Load X Positive Immediate	C400	3-18
LAM	Load A Minus Immediate	C700	3-17
LXM	Load X Minus Immediate	C500	3-17

CONDITIONAL JUMP (Class 4 or 8)

Microcoded (Class 8)

JOC	Jump on Condition Specified	2000	3-18

Arithmetic (Class 4)

JAG	Jump if A Greater than Zero	3180	3-20
JAP	Jump if A Positive	3080	3-20
JAZ	Jump if A Zero	2100	3-20
JAN	Jump if A Not Zero	3100	3-20
JAL	Jump if A less than or equal to		3-20
	Zero	2180	3-20
JAM	Jump if A Minus	2080	3-20
JXZ	Jump if X Zero	2800	3-20
JXN	Jump if X not Zero	3800	3-20

Control (Class 4)

JSS	Jump if Sense Sense Switch Set	3400	3-20
JSR	Jump if Sense Switch Reset	2400	3-20
JOS	Jump if OV Set	2200	3-20
JOR	Jump if OV Reset	3200	3-20

SHIFT CLASS (Class 5)

		Instruction Code in Hex	Reference Page
Single Register			
Arithmetic			
ARA	Arithmetic Right A	1000	3-21
ARX	Arithmetic Right X	10A8	3-21
ALA	Arithmetic Left A	1050	3-21
ALX	Arithmetic Left X	1028	3-21
Logical			
LRA	Logical Right A	1300	3-22
LRX	Logical Right X	13A8	3-22
LLA	Logical Left A	1350	3-22
LLX	Logical Left X	1358	3-22
Rotate			
RRA	Rotate Right A with OV	1100	3-23
RRX	Rotate Right X with OV	11A8	3-23
RLA	Rotate Left A with OV	1150	3-23
RLX	Rotate Left X with OV	1128	3-23
Double Register			
Logical			
LLL	Long Logical Left	1B00	3-24
LLR	Long Logical Right	1B80	3-24
Rotate			
LRL	Long Rotate Left with OV	1900	3-24
LRR	Long Rotate Right with OV	1980	3-24

REGISTER CHANGE (Class 6)

Accumulator			
ZAR	Zero A Register	0110	3-25
ARP	Set A Register to Positive 1	0350	3-25
ARM	Set A Register to Minus 1	0010	3-25
CAR	Complement (1's) A Register	0210	3-25
NAR	Negate A Register	0310	3-25
IAR	Increment A Register	0150	3-25
DAR	Decrement A Register	00D0	3-25

		Instruction Code in Hex	Reference Page
Index			
ZXR	Zero X Register	0108	3-26
XRP	Set X Register to Positive 1	0528	3-26
XRM	Set X Register to Minus 1	0008	3-26
CXR	Complement (1's) X Register	0408	3-26
NXR	Negate X Register	0508	3-26
IXR	Increment X Register	0128	3-26
DXR	Decrement X Register	00A8	3-26
Overflow			
SOV	Set Overflow	1400	3-26
ROV	Reset Overflow	1200	3-26
COV	Complement Overflow	1600	3-26
SAO	Sign of A to OV	1340	3-26
SXO	Sign of X to OV	1320	3-26
LAO	Least significant bit of A to OV	13C0	3-26
LXO	Least significant bit of X to OV	13A0	3-26
Multi-Register			
ZAX	Zero A and X Register	0118	3-27
AXP	Set A and X Registers to Positive 1	0358	3-27
AXM	Set A and X Registers to Minus 1	0018	3-27
TAX	Transfer A to X	0048	3-27
TXA	Transfer X to A	0030	3-27
ANA	AND of A and X to A	0070	3-26
ANX	AND of A and X to X	0068	3-26
NRA	NOR of A and X to A	0610	3-27
NRX	NOR of A and X to X	0608	3-27
CAX	1's Complement (A) and put in X	0208	3-27
CXA	1's Complement (X) and put in A	0410	3-27
NAX	Negate (A) and put in X	0308	3-27
NXA	Negate (X) and put in A	0510	3-27
IAX	Increment (A) and put in X	0148	3-27
IXA	Increment (X) and put in A	0130	3-27
DAX	Decrement (A) and put in X	00C8	3-27
DXA	Decrement (X) and put in A	00B0	3-27
Console Register			
ICA	Input Console Data Register to A	5804	3-28
ICX	Input Console Data Register to X	5A04	3-28
ISA	Input Console Sense Register to A	5801	3-28
ISX	Input Console Sense Register to X	5A01	3-28
OCA	Output A to Console Data Register	4404	3-28
OCX	Output X to Console Data Register	4604	3-28

CONTROL (Class 6)

		Instruction Code in Hex	Reference Page
Processor			
NOP	No operation	0000	3-29
HLT	Halt	0800	3-29
STOP	Halt with Operand	0800	3-29
WAIT	Wait for Interrupts	F600	3-29
Mode Control			
SBM	Set Byte Operand Mode	0E00	3-29
SWM	Set Word Operand Mode	0F00	3-29
Status			
SIN	Status Inhibit	6800	3-30
SIA	Status Input to A	5800	3-30
SIX	Status Input to X	5A00	3-30
SOA	Status Output from A	6C00	3-30
SOX	Status Output from X	6E00	3-30
Interrupts			
EIN	Enable Interrupts	0A00	3-30
DIN	Disable Interrupts	0C00	3-30
CIE	Console Interrupt Enable	4005	3-30
CID	Console Interrupt Disable	4006	3-30
PFE	Power Fail Interrupt Enable	4002	3-31
PFD	Power Fail Interrupt Disable	4003	3-31
TRP	Trap	4007	3-31

INPUT/OUTPUT (Class 7)

Control			
SEL	Select	4000	3-32
SEA	Select and Present A	4400	3-32
SEX	Select and Present X	4600	3-32
SEN	Sense and Skip on Response	4900	3-32
SSN	Sense and Skip on no Response	4800	3-32

		Instruction Code in Hex	Reference Page
Unconditional Word			
INA	Input Word to A	5800	3-33
INAM	Input Word to A Masked	5C00	3-33
INX	Input Word to X	5A00	3-33
INXM	Input Word to X Masked	5E00	3-33
OTA	Output A	6C00	3-33
OTX	Output X	6E00	3-33
OTZ	Output Zero's	6800	3-33
Conditional Word			
RDA	Read Word to A	5900	3-34
RDAM	Read Word to A Masked	5D00	3-34
RDX	Read Word to X	5B00	3-34
RDXM	Read Word to X Masked	5F00	3-34
WRA	Write A	6D00	3-34
WRX	Write X	6F00	3-34
WRZ	Write Zero's	6900	3-34
Unconditional Byte			
IBA	Input Byte to A	7800	3-35
IBAM	Input Byte to A Masked	7C00	3-35
IBX	Input Byte to X	7A00	3-35
IBXM	Input Byte to X Masked	7E00	3-35
Conditional Byte			
RBA	Read Byte to A	7900	3-35
RBAM	Read Byte to A Masked	7D00	3-36
RBX	Read Byte to X	7B00	3-36
RBXM	Read Byte to X Masked	7F00	3-36
Block			
BIN	Input Block to Memory	7100	3-37
BOT	Output Block from Memory	7500	3-37
Automatic			
AIN	Automatic Input to Memory--Word	5000	3-40
AOT	Automatic Output from Memory--Word	6000	3-40
AIB	Automatic Input to Memory--Byte	5400	3-40
AOB	Automatic Output from Memory--Byte	6400	3-40

Appendix D
INSTRUCTION SET IN NUMERICAL ORDER

This appendix contains the ALPHA LSI instruction set in machine code in numerical order. For each instruction, reference is made to one of the machine code formats listed below. Instructions with variable fields (D, K, etc.) have been appended with asterisks (*).

D = Address Field (0 to 255)
I = Direct/Indirect Address Bit
M = Address Mode Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	OP CODE				M		I	D							

M	I	Word Mode (Word Operand)	Byte Mode (Byte Operand)
00	0	Y = (D), Words : 00-: FF	Y = (D), Bytes : 00-: FF
01	0	Y = (D) + (P) + 1	Y = (D) + (P) 1, Byte 0
10	0	Y + (D) + (X)	Y = (D) + (X)
11	0	Y = (P) - (D)	Y = (D) + (P) + 1, Byte 1
00	1	AP = (D), AP = (AP) , Y = (AP)	AP = (D), Y = (AP)
01	1	AP = (D) + (P) + 1, AP = (AP) , Y = (AP)	AP = (D) + (P) + 1, Y = (AP)
10	1	AP = (D), AP = (AP) , Y = (AP) + (X)	AP = (D), Y = (AP) + (X)
11	1	AP = (P) - (D), AP = (AP) , Y = (AP)	AP = (P) - (D), Y = (AP)

Figure D-1. Single-Word Memory Reference Instruction Machine Code Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	1	0	OP CODE			K			
I	ADDRESS														

Op Code = 100 for NRM 16 through 31
= 101 for NRM 0 through 15
= 110 for MPY
= 111 for DVD

I = Indirect Addressing
1 = Indirect Address
0 = Direct Address
K = Instruction Count

Figure D-2. Double-Word Memory Reference Instruction Machine Code Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	OP CODE			D							

D = 8-Bit Immediate Operand

Figure D-3. Byte Immediate Instruction Machine Code Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	G	MICROCODE					R	D FIELD					

Bits	Field	Definition

12	G	Test Group Indicator:
		G = 1 for AND Group
		G = 0 for OR Group

7-11	Conditions	Microcode of Test Conditions:

Bit	AND Group	OR Group
7	A Positive	A Negative
8	A ≠ 0	A = 0
9	OV Reset	OV Set (Resets OV)
10	Sense Indicator on	Sense Indicator off
11	X ≠ 0	X = 0

Bits	Field	Definition

6	R	Jump Direction:
		R = 0 for Forward Jump
		R = 1 for Backward Jump

0-5	D Field	Jump Distance (-63 to +64)

Figure D-4. Conditional Jump Instruction Machine Code Format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td colspan="8">OP CODE</td><td colspan="3">K</td></tr>
</table>

K = Shift Control Count, Shift Will Move 1 + K Bit Positions.
Op Code = Shift Control Code Which Selects Source, Type of Shift, and Location of Results

Figure D-5 Single-Register Shift Instruction Machine Code Format

<table>
<tr><td>15.</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="7">OP CODE</td><td colspan="4">K</td></tr>
</table>

Op Code = Shift Control Code Which Selects the Type of Long Shift to be Executed
K = Shift Count. Shift Will Move 1 + K Bit Positions

Figure D-6 Double-Register Shift Instruction Machine Code Format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>.0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="8">OP CODE</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Op Code = The Register Change Control Code which specifies the Source, Operation, and Location of Results

Figure D 7. Register Change Instruction Machine Code Format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td colspan="7">OP CODE</td><td colspan="8">OP CODE, H or SC</td></tr>
</table>

H = Halt ID Indicator
SC = Sin Instruction Count - 1

Figure D-8. Control Instruction Machine Code Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	OP CODE						DEVICE ADDRESS					FUNCTION CODE		

Function Code =Specifies which device function or register
Device Address = The device's assigned address
Opcode = Operation Code Specifying One of the I/O Instructions

Figure D-9. Input/Output Instruction Machine Code Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	0	1	OP CODE		0	B	0	0	DEVICE ADDRESS					FUNCTION CODE		
P+1	1	BYTE/WORD COUNTER, WC (2'S COMPLEMENT)														
P+2	0	ADDRESS POINTER, AP (START LOCATION -1)														

Opcode; 01 = Input, 10 = Output
B = 0: Word Transfer
B = 1: Byte Transfer
Byte/Word Counter = Number of Executions Until Skip or Echo
Address Pointer = Memory Location of I/O Transaction

Figure D-10 Automatic Input/Output Instruction Machine Code Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	OP CODE						DEVICE ADDRESS					FUNCTION CODE		
0	BASE ADDRESS -1														

Figure D-11. Block Input/Output Instruction Machine Code Format

Instruction Code In Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
0000	NOP	NO OPERATION	7	3-24
0008	XRM	X REGISTER TO MINUS ONE	7	3-26
0010	ARM	A REGISTER TO MINUS ONE	7	3-25
0018	AXM	A AND X REGISTERS TO MINUS ONE	7	3-27
0030	TXA	TRANSFER X TO A	7	3-27
0048	TAX	TRANSFER A TO X	7	3-27
0068	ANX	AND OF A AND X TO X	7	3-26
0070	ANA	AND OF A AND X TO A	7	3-26
00A8	DXR	DECREMENT X REGISTER	7	3-26
00B0	DXA	DECREMENT X TO A	7	3-27
00C8	DAX	DECREMENT A TO X	7	3-27
00D0	DAR	DECREMENT A REGISTER	7	3-25
0108	ZXR	ZERO X REGISTER	7	3-26
0110	ZAR	ZERO A REGISTER	7	3-25
0118	ZAX	ZERO A AND X	7	3-27
0128	IXR	INCREMENT X REGISTER	7	3-26
0130	IXA	INCREMENT X TO A	7	3-27
0148	IAX	INCREMENT A TO X	7	3-27
0150	IAR	INCREMENT A REGISTER	7	3-25
0208	CAX	COMPLEMENT OF A TO X	7	3-27
0210	CAR	COMPLEMENT A REGISTER	7	3-25
0308	NAX	NEGATE A TO X	7	3-27
0310	NAR	NEGATE A REGISTER	7	3-25

Instruction Code in Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
0350	ARP	A REGISTER TO PLUS ONE	7	3-25
0358	AXP	A AND X REGISTERS TO PLUS ONE	7	3-27
0408	CXR	COMPLEMENT X REGISTER	7	3-26
0410	CXA	COMPLEMENT OF X TO A	7	3-27
0508	NXR	NEGATE X REGISTER	7	3-26
0510	NXA	NEGATE X TO A	7	3-27
0528	XRP	X REGISTER TO PLUS ONE	7	3-26
0608	NRX	NOR OF A AND X TO X	7	3-27
0610	NRA	NOR OF A AND X TO A	7	3-27
0800	HLT	HALT	8	3-29
0800	STOP*	HALT WITH OPERAND	8	3-29
0A00	EIN	ENABLE INTERRUPTS	8	3-30
0C00	DIN	DISABLE INTERRUPTS	8	3-30
0E00	SBM	SET BYTE MODE	8	3-29
1028	ALX*	ARITHMETIC SHIFT X LEFT	5	3-21
1050	ALA*	ARITHMETIC SHIFT A LEFT	5	3-21
10A8	ARX*	ARITHMETIC SHIFT X RIGHT	5	3-21
10D0	ARA*	ARITHMETIC SHIFT A RIGHT	5	3-21
1128	RLX*	ROTATE X LEFT WITH OVERFLOW	5	3-23
1150	RLA*	ROTATE A LEFT WITH OVERFLOW	5	3-23
11A8	RRX*	ROTATE X RIGHT WITH OVERFLOW	5	3-23
11D0	RRA*	ROTATE A RIGHT WITH OVERFLOW	5	3-23
1200	ROV	RESET OVERFLOW	5	3-26

Instruction Code in Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
1320	SXO	SIGN OF X TO OVERFLOW	5	3-26
1328	LLX*	LOGICAL SHIFT X LEFT	5	3-22
1340	SAO	SIGN OF A TO OVERFLOW	5	3-26
1350	LLA*	LOGICAL SHIFT A LEFT	5	3-22
13A0	LXO	LSB OF X TO OVERFLOW	5	3-26
13A8	LRX*	LOGICAL SHIFT X RIGHT	5	3-22
13C0	LAO	LSB OF A TO OVERFLOW	5	3-26
13D0	LRA*	LOGICAL SHIFT A RIGHT	5	3-22
1400	SOV	SET OVERFLOW	5	3-26
1600	COV	COMPLEMENT OVERFLOW	5	3-26
1900	LRL*	LONG ROTATE LEFT	6	3-24
1940	NRM	NORMALIZE A AND X	2	3-16
1960	MPY	MULTIPLY AND ADD	2	3-15
1970	DVD	DIVIDE	2	3-14
1980	LRR*	LONG ROTATE RIGHT	6	3-24
1B00	LLL*	LONG LOGICAL SHIFT LEFT	6	3-24
1B80	LLR*	LONG LOGICAL SHIFT RIGHT	6	3-24
2080-3F80 Forward 20C0-3FC0 Backward	JOC*	JUMP ON CONDITION	4	3-18
2080 Forward 20C0 Backward	JAM*	JUMP IF A MINUS	4	3-20
2100 Forward 2140 Backward	JAZ*	JUMP IF A ZERO	4	3-20
2180 Forward 21C0 Backward	JAL*	JUMP IF A LESS THAN ONE	4	3-20

Instruction Code in Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
2200 Forward 2240 Backward	JOS*	JUMP IF OVERFLOW SET	4	3-20
2400 Forward 2440 Backward	JSR*	JUMP IF SENSE SWITCH RESET	4	3-20
2800 Forward 2840 Backward	JXZ*	JUMP IF X ZERO	4	3-20
3080 Forward 30C0 Backward	JAP*	JUMP IF A POSITIVE	4	3-20
3100 Forward 3140 Backward	JAN*	JUMP IF A NOT ZERO	4	3-20
3180 Forward 31C0 Backward	JAG*	JUMP IF A GREATER THAN ZERO	4	3-20
3200 Forward 3240 Backward	JOR*	JUMP IF OVERFLOW RESET	4	3-20
3400 Forward 3440 Backward	JSS*	JUMP IF SENSE SWITCH SET	4	3-20
3800 Forward 3840 Backward	JXN*	JUMP IF X NOT ZERO	4	3-20
4000	SEL*	SELECT FUNCTION	9	3-32
4002	PFE	POWER FAIL ENABLE	9	3-31
4003	PFD	POWER FAIL DISABLE	9	3-31
4005	CIE	CONSOLE INTERRUPT ENABLE	9	3-31
4006	CID	CONSOLE INTERRUPT DISABLE	9	3-31
4007	TRP	TRAP	9	3-31
4400	SEA*	SELECT AND PRESENT A	9	3-31
4404	OCA	OUTPUT A TO CONSOLE REGISTER	9	3-28
4600	SEX*	SELECT AND PRESENT X	9	3-32

Instruction Code In Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
4604	OCX	OUTPUT X TO CONSOLE REGISTER	9	3-28
4800	SSN*	SENSE AND SKIP ON NO RESPONSE	9	3-32
4900	SEN*	SENSE AND SKIP ON RESPONSE	9	3-32
5000	AIN*	AUTOMATIC INPUT WORD TO MEMORY	10	3-40
5400	AIB*	AUTOMATIC INPUT BYTE TO MEMORY	10	3-40
5800	INA*	INPUT TO A REGISTER	9	3-33
5800	SIA	STATUS INPUT TO A	9	3-30
5801	ISA	INPUT SENSE REGISTER TO A	9	3-28
5804	ICA	INPUT CONSOLE REGISTER TO A	9	3-28
5900	RDA*	READ WORD TO A REGISTER	9	3-34
5A00	INX*	INPUT TO X REGISTER	9	3-33
5A01	ISX	INPUT SENSE REGISTER TO X	9	3-28
5A04	ICX	INPUT CONSOLE REGISTER TO X	9	3-28
5B00	RDX*	READ WORD TO X REGISTER	9	3-34
5C00	INAM*	INPUT TO A REGISTER MASKED	9	3-33
5D00	RDAM*	READ WORD TO A REGISTER MASKED	9	3-34
5E00	INXM*	INPUT TO X REGISTER MASKED	9	3-33
5F00	RDXM*	READ WORD TO X REGISTER MASKED	10	3-34
6000	AOT*	AUTOMATIC OUTPUT WORD FROM MEMORY	10	3-40
6400	AOB*	AUTOMATIC OUTPUT BYTE FROM MEMORY	10	3-40
6800	OTZ*	OUTPUT ZERO	9	3-33

Instruction Code In Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
6800	SIN*	STATUS INHIBIT	8	3-30
6900	WRZ*	WRITE ZERO	9	3-34
6C00	OTA*	OUTPUT A REGISTER	9	3-33
6C00	SOA	STATUS OUTPUT FROM A	9	3-30
6D00	WRA*	WRITE FROM A REGISTER	9	3-34
6E00	OTX*	OUTPUT X REGISTER	3	3-33
6E00	SOX	STATUS OUTPUT FROM X	9	3-30
6F00	WRX*	WRITE FROM X REGISTER	3	3-34
7100	BIN*	BLOCK IN	11	3-30
7500	BOT*	BLOCK OUT	11	3-37
7800	IBA*	INPUT BYTE TO A REGISTER	9	3-35
7900	RBA*	READ BYTE TO A REGISTER	9	3-35
7A00	IBX*	INPUT BYTE TO X REGISTER	9	3-35
7B00	RBX*	READ BYTE TO X REGISTER	9	3-36
7C00	IBAM*	INPUT BYTE TO A REGISTER MASKED	9	3-35
7D00	RBAM*	READ BYTE TO A REGISTER MASKED	9	3-36
7E00	IBXM*	INPUT BYTE TO X REGISTER MASKED	9	3-35
7F00	RBXM*	READ BYTE TO X REGISTER MASKED	9	3-36
8000	AND*	AND TO A	1	3-11
8000	ANDB*	AND BYTE TO A	1	3-11
8800	ADD*	ADD TO A	1	3-10
8800	ADDB*	ADD BYTE TO A	1	3-10

Instruction Code In Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
9000	SUB*	SUBTRACT FROM A	1	3-40
9000	SUBB*	SUBTRACT BYTE FROM A	1	3-10
9800	STA*	STORE A	1	3-12
9800	STAB*	STORE A BYTE	1	3-12
A000	IOR*	INCLUSIVE OR TO A	1	3-11
A000	IORB*	INCLUSIVE OR BYTE TO A	1	3-11
A800	XOR*	EXCLUSIVE OR TO A	1	3-11
A800	XORB*	EXCLUSIVE OR BYTE TO A	1	3-11
B000	LDA*	LOAD A	1	3-12
B000	LDAB*	LOAD A BYTE	1	3-12
B800	EMA*	EXCHANGE MEMORY AND A	1	3-12
B800	EMAB*	EXCHANGE MEMORY BYTE AND A	1	3-12
C000	CAI*	COMPARE TO A IMMEDIATE	3	3-17
C100	CXI*	COMPARE TO X IMMEDIATE	3	3-17
C200	AXI*	ADD TO X IMMEDIATE	3	3-17
C300	SXI*	SUBTRACT FROM X IMMEDIATE	3	3-18
C400	LXP*	LOAD X POSITIVE IMMEDIATE	3	3-18
C500	LXM*	LOAD X MINUS IMMEDIATE	3	3-17
C600	LAP*	LOAD A POSITIVE IMMEDIATE	3	3-17
C700	LAM*	LOAD A MINUS IMMEDIATE	3	3-17
CD00	SCM*	SCAN MEMORY	1	3-13
D000	CMS*	COMPARE AND SKIP IF HIGH OR EQUAL	1	3-12

Instruction Code In Hex	Instruction Mnemonic	Description	Machine Code Format	Reference Page
D000	CMSB*	COMPARE BYTE AND SKIP IF HIGH OR EQUAL	1	3-13
D800	IMS*	INCREMENT MEMORY AND SKIP ON ZERO RESULT	1	3-13
E000	LDX*	LOAD X	1	3-12
E000	LDXB*	LOAD X BYTE	1	3-12
E800	STX*	STORE X	1	3-12
E800	STXB	STORE X BYTE	1	3-12
F000	JMP*	JUMP UNCONDITIONAL	1	3-13
F600	WAIT	WAIT FOR INTERRUPTS	1	3-29
F800	JST*	JUMP AND STORE	1	3-13

Appendix E
ALPHA LSI EXECUTION TIMES

E.1 GENERAL

The Appendix defines the execution time of each instruction in the ALPHA LSI instruction set. A variety of memories, with varying access times, are offered with the ALPHA LSI. The variation in memory access time makes a tabulation of execution times difficult. For this reason time calculation algorithms are provided. These algorithms are useful with any memory access time by making the appropriate substitution. Table E-1 lists the standard core memory variables that are used in the execution time algorithms. Table E-2 provides a complete tabulation of instructions by class and subclass along with the time calculation algorithm and representative standard core memory timing.

Table E-1. Standard Core Memory Algorithm Variables

c = Memory Cycle Time + 1.6μs
m = Memory Access Time, Effective = 0.6μs
t = Processor Microcycle Time + 1.6μs
i = Number of Indirect Address Levels
w = Number of Words or Bytes Transferred or Scanned
n = Number of Shifts
A = Address Calculation Time for Memory Reference Instructions:

DIRECT SCRATCHPAD	t+m	2.2
DIRECT RELATIVE	t+m	2.2
DIRECT INDEXED	2t + m	3.8
INDIRECT SCRATCHPAD	(2t + m)i	3.8i
INDIRECT RELATIVE	t + (2t + m)i	1.6 + 3.8i
INDIRECT INDEXED	t + (2t + m)i	1.6 + 3.8i

Table E-2. Execution Time Algorithms

	TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING (t=1.6µs) in microseconds
MEMORY REFERENCE		
ARITHMETIC		
ADD	4t + m + A	7 + A
SUB	4t + m + A	7 + A
LOGICAL		
AND	4t + m + A	7 + A
IOR	4t + m + A	7 + A
XOR	4t + m + A	7 + A
DATA TRANSFER		
EMA	5t + 2m + A	9.2 + A
LDA	3t + m + A	4.8 + A
LDX	3t + m + A	4.8 + A
STA	3t + m + A	4.8 + A
STX	3t + m + A	4.8 + A
PROGRAM TRANSFER		
CMS	8t + m + A	13.4 + A
IMS	6t + 2m + A	10.8 + A
JMP	3t + A	4.2 + A
JST (Non-Interrupt)	5t + m + A	8.6 + A
JST (Interrupt)	4t + m + A	7 + A
SCN	(8t + m + A)w	(13.4 + A)w
DOUBLE WORD MEMORY REFERENCE		
MPY	69t + 3m + (2t +m)i	112.2 + 3.8i
DVD	74t + 3m + (2t +m)i	120.2 + 3.8i
NRM (count expires)	11t + 4m +6nt + (2t + m)i	20.0 + 9.6n +3.8i
NRM (count does not expire)	13t + 4m + 6nt + (2t +m)i	23.2 + 9.6n + 3.8i
BYTE IMMEDIATE		
AXI	3t + m	4.8
CAI	4t + m	7.0
CXI	4t + m	7.0
LAM	3t + m	4.8
LAP	3t + m	4.8
LXM	3t + m	4.8
LXP	2t + m	4.8
SXI	3t + m	4.8

Table E-2. Execution Time Algorithms (Cont'd)

	TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING (t=1.6μs) in microseconds
DOUBLE REGISTER LOGICAL SHIFTS		
LLL	2t + m + 2nt	3.8 + 3.2n
LLR		
DOUBLE REGISTER ROTATE SHIFTS		
LRL	2t + m + 2nt	3.8 + 3.2n
LRR		
REGISTER CHANGE		
A REGISTER CHANGE		
ARM	3t + m	5.4
ARP		
CAR		
DAR		
IAR		
NAR		
ZAR		
X REGISTER CHANGE		
CXR	3t + m	5.4
DXR		
IXR		
NXR		
XRM		
XRP		
ZXR		
OVERFLOW REGISTER CHANGE		
COV	3t + m	5.4
LAO	4t + m	7.0
LXO	4t + m	5.4
ROV	3t + m	5.4
SAO	3t + m	5.4
SOV	3t + m	5.4
SXO	3t + m	5.4

Table E-2. Execution Time Algorithms (Cont'd)

	TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING (t=1.6μs) in microseconds
CONDITIONAL JUMP		
MICROCODED		
JOC		
ALL Double Register Tests	9t + m	15.0
ALL Others	4t + m	7.0
ARITHMETIC		
JAG	4t + m	7.0
JAL		
JAM		
JAP		
JAZ		
JXN		
JXZ		
CONTROL		
JOC	4t + m	7.0
JOR		
JOS		
JSR		
JSS		
SHIFT		
ARITHMETIC SHIFTS		
ALA	2t + m + nt	3.8 + 1.6n
ALX		
ARA		
ARX		
LOGICAL SHIFTS		
LLA	2t + m + nt	3.8 + 1.6n
LLX		
LRA		
LRX		
ROTATE SHIFTS		
RLA	2t + m + nt	3.8 + 1.6n
RLX		
RRA		
RRX		

Table E-2. Execution Time Algorithms (Cont'd)

	TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING (t=1.6μs) in microseconds
MULTI-REGISTER CHANGE		
ANA	3t + m	5.4
ANX	3t + m	5.4
AXM	4t + m	7.0
AXP	4t + m	7.0
CAX	3t + m	5.4
CXA	3t + m	5.4
DAX	3t + m	5.4
DXA	3t + m	5.4
IAX	3t + m	5.4
IXA	3t + m	5.4
NAX	3t + m	5.4
NRA	3t + m	5.4
NRX	4t + m	7.0
NXA	4t + m	7.0
TAX	3t + m	5.4
TXA	3t + m	5.4
ZAX	4t + m	7.0
CONSOLE REGISTER		
ICA	3.375t + m	6.0
ICX	3.375t + m	6.0
ISA	3.375t + m	6.0
ISX	3.375t + m	6.0
OCA	3.375t + m	6.0
OCX	3.375t + m	6.0
CONTROL		
PROCESSOR CONTROLS		
HLT (STOP)	3t + m	5.4
NOP		
MODE CONTROLS		
SBM	3t + m	5.4
SWM	3t + m	5.4
STATUS CONTROLS		
SIA	3.375t + m	6.0
SIN		
SOA		
SOX		

Table E-2. Execution Time Algorithms (Cont'd)

		TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING (t=1.6μs in microseconds
INTERRUPT CONTROLS			
	CID	3.375t + m	6.0
	CIE	3.375t + m	6.0
	DIN	4t + m	7.0
	EIN	3t + m	5.4
	PFD	3.375t + m	6.0
	PFE	3.375t + m	6.0
	TRP	3.375t + m	6.0
CONTROL			
	SEN	4.375t + m	7.6
	SEA	3.375t + m	6.0
	SEL	3.375t + m	6.0
	SEX	3.375t + m	6.0
	SSN	4.375t + m	7.6
UNCONDITIONAL WORD			
	INA	3.375t +m	6.0
	INAM	4.375t + m	7.6
	INX	3.375t + m	6.0
	INXM	4.375t + m	7.6
	OTA	3.375t + m	6.0
	OTX	3.375t + m	6.0
	OTZ	3.375t + m	6.0
CONDITIONAL WORD			
	RDA	4.375t + m	7.6
	RDAM	6.375t + m	10.8
	RDX	4.375t + m	7.6
	RDXM	6.375t + m	10.8
	WRA	4.375t + m	7.6
	WRX	4.375t + m	7.6
	WRZ	4.375t + m	7.6
UNCONDITIONAL BYTE			
	IBA	4.375t + m	7.6
	IBAM	5.375t + m	9.2
	IBX	4.375t + m	7.6
	IBXM	5.375t + m	9.2

Table E-2. Execution Time Algorithms (Cont'd)

		TIME CALCULATION ALGORITHM	REPRESENTATIVE TIMING ($t=1.6\mu s$) in microseconds
CONDITIONAL BYTE			
	RBA	6.375t + m	10.8
	RBAM	7.375t + m	12.4
	RBX	6.375t + m	10.8
	RBXM	7.375t + m	12.4
BLOCK			
	BIN	7t + 2m +(4.375 +m)w	12.4 + 7.6w
	BOT	7t + 2m + (4.375 +m)w	12.4 + 7.6w
AUTOMATIC			
	AIB	14.375t + 5m	26.0
	AIB (Under Interrupts)	12.375t + 5m	22.8
	AIN	14.375t + 5m	26.0
	AIN (Under Interrupts)	12.375t + 5m	22.8
	AOB	14.375t + 5m	26.0
	AOB (Under Interrupts)	12.375t + 5m	22.8
	AOT	14.375t + 5m	26.0
	AOT (Under Interrupts)	12.375t + 5m	22.8

Appendix F
SOFTWARE SUMMARY

F.1 INTRODUCTION

This appendix contains short usage summaries of the standard system support software offered by Computer Automation, Inc.

Table F-1. Assembler Directives

ABS	Define Absolute Assembly
DATA	Data Definition (:Hex, 0 Octal, 'ASCII', Address)
EQU	Equate Symbol
ORG	Define Origin
REF	External Reference - Pointer
RES	Reserve Storage
TEXT	'ASCII Message'
NAM	External Name Definition
ENT	Subroutine Entry
RTN	Subroutine Return
IFT	Conditional Assembly if True
IFF	Conditional Assembly if False
ENDC	End of Conditional Assembly
REL	Define Relocatable Assembly
END	End of Assembly
BAC	Byte Address Constant
SET	Set Symbol Redefinable
EXTR	External Reference = Scratchpad
TITL	Page Eject with Title
. (Period)	Page Eject without Title
*(Asterisk)	Comment Line
↑ (Up Arrow)	Pause

F.2 BOOTSTRAP

To Enter:

Set P = :nFF8
Set WRITE mode
Enter Data } Once per word
Depress M }

To Display:

Set P = :nFF8
Set READ mode
Depress M } Once per word

Loc	TTY	HSPT
:nFF8	403B	4033
:nFF9	7939	7931
:nFFA	1357	1357
:nFFB	7939	7931
:nFFC	9C00	9C00
:nFFD	0128	0128
:nFFE	3145	3145
:nFFF	0800	0800

F.3 SOFTWARD OPERATION SUMMARY

F.3.1 Autoload

RESET
To relocate, set X = load address
Enter sum of options in Sense Register:

Device Mode	TTY	HSPT	MT	Cassette	Disk
Load ABS	:0	:1	:2	:3	:4
Load Rel	:8	:9	:A	:B	:C

For Load and Go, set SENSE Switch
Ready Device
AUTO

F.3.2 Binary Loader (BLD)

Load BLD; set P = BLD rel. zero
To relocate; set X = load address, enter : 8 into Sense register:
Ready tape in reader (TTY or HSPT)
RUN

F.3.3 Binary Dump/Verify (BDP/VER)

Load BDP; set P = BDP rel. zero
Set A = Initial location
Set X = Last location
Enter sum of options in Sense register:

Device Mode	TTY	HSPT
Punch Abs	: 0	: 1
Punch Rel	: 8	: 9

To verify, add : 4
To suppress EOF, add : 2
For transfer address, set SENSE switch
RUN
If Halt (I = : 0802), set A = transfer address, RUN

F.3.4 Object Loader (LAMBDA)

Load LAMBDA; set P = LAMBDA zero
Set A = Relocation Bias or zero
Set X = Base Page Bias or zero
Enter sum of options in Sense register:

Print Symbols Load Mode	Defined and Undefined	Defined Only	Undefined Only	Neither
Library	: 0	: 2	: 4	: 6
Unconditional	: 8	: A	: C	: E

Ready tape in reader (TTY or HSPT)
RUN

F.3.5 BETA-4 Assembler

Load BETA-4; set P = : 0100; RUN
Enter sum of options in Sense register:

List Punch	TTY	LP
TTY	: 0	: 2
HSP	: 4	: 6

For Error Only list, add : 1
To repeat Pass 2, add : 8
Ready source in reader (TTY or HSPT)
RUN

F.3.6 BETA-8 Assembler

Load BETA-8; set P = : 0100; RUN
Select Options

For Enter	SI=	LO=	BO=	SD=	P#=
B	BATCH	Error	Error	Error	Error
L	Error	Error	Library	Error	Error
0	Punch EOF	No Listing	No Binary	No Save	1
1	Keyboard	TTY	TTY	Core	1
2	TTY	D.P.	Error	Unit 0	2
3	HSPT	Cent.	HSPT	Unit 1	1
4	C.R.	Cent.	TTY	Unit 2	1
5	C.R.	Cent.	TTY	Unit 3	1

F.3.7 OMEGA Conversational Assembler

Load OMEGA; Set P = : 0100; RUN
Command Summary:

>AF.	Add keyboard lines to buffer after last line.
>An.	Add keyboard lines to buffer after line n.
>B.	Clear the buffer.
>CInLnOn.	Connect devices.
>CI0.	Punch EOF.
>DF.	Delete the last buffer line.
>Dn.	Delete buffer line n.
>Dn@m.	Delete buffer lines n through m.
>Eh.	Set end of buffer to h (hexadecimal) and intialize OMEGA.
>I.	Initialize OMEGA
>LF.	List the last buffer line.
>Ln.	List buffer line n.
>Ln@m.	List buffer lines n through m.
>PLT@1@F.	Punch the buffer with leader and trailer.
>PL@n@m.	Punch buffer lines n through m with leader.
>P@n@m.	Punch buffer lines n through m.
>PT@n@m.	Punch buffer lines n through m with trailer.
>Rn.	Read source to line n and add to buffer.
>Sn.	Read source to line n-1, add to buffer, and skip line n.
>Sn@m.	Read source to line n-1, add to buffer and skip lines n through m.
>T.	Reset tape line count to zero.
>Tn.	Reset tape line count to n.
>XA.	Assemble.
>XE.	Assemble with ERROR only listing.
>XA2. or XE2.	Assemble starting with Pass 2.

Device Selection

Input: (I)	Object: (O)	List: (L)
0 = none	0 = none	0 = none
1 = Teletype Keyboard	1 = Teletype Paper Tape	1 = Teletype
2 = Teletype Paper Tape	2 = N/A	2 = Data Products Printer
3 = High Speed Paper Tape	3 = High Speed Paper Tape	3 = Centronics Printer
4 = Card Reader		
5 = Core (assemble)		

F.3.8 Source Tape Preparation Program

Load STP; set P = : 0100; RUN
Command Summary:

≥ AF.	Add keyboard lines to buffer after last line.
≥ An.	Add keyboard lines to buffer after line n.
≥ B.	Clear the buffer.
≥ CTT.	Connect teletype reader and teletype punch.
≥ CRT.	Connect high speed reader and teletype punch.
≥ CRP.	Connect high speed reader and high speed punch.
≥ CTP.	Connect teletype reader and high speed punch.
≥ DF.	Delete the last buffer line.
≥ Dn.	Delete buffer line n.
≥ Dn@m.	Delete buffer lines n through m.
≥ Eh.	Set end of buffer to h (hexadecimal).
≥ I.	Initialize STP (clear buffer and set T to zero).
≥ LF.	List the last buffer line.
≥ Ln.	List buffer line n.
≥ Ln@m.	List buffer lines n through m.
≥ PLT@1@F	Punch the buffer with leader and trailer.
≥ PL@n@m.	Punch buffer lines n through m with leader.
≥ P@n@m.	Punch buffer lines n through m.
≥ PT@n@m.	Punch buffer lines n through m with trailer.
≥ Qn.	Set ADD function termination character to n.
≥ Rn.	Read tape to line n and add to buffer.
≥ Sn.	Read tape to line n-1, add to buffer, and skip line n.
≥ Sn@m.	Read tape to line n-1, add to buffer, and skip lines n through m.
≥ T.	Reset tape line count to zero.
≥ Tn.	Reset tape line count to n.

F.3.9 Debug (DBG)

Debug is a 'binary relocatable' program and, as such, may be loaded any place in memory by the ALPHA LSI Binary Loader program (BLD). Transferring to the first location in Debug (enter start location of Debug into the P register and depress RUN) will initialize Debug to accept any of the Debug commands summarized below:

COMMAND SUMMARY

≥ A.	Display pseudo A register
≥ Av.	Set pseudo A register to value v.
≥ Ba.	Continue breakpoint to location a.
≥ Ba,b.	Continue breakpoint to location (a or b).
≥ Ba@b.	Breakpoint from location a to b.
≥ Ba@b,c.	Breakpoint from location a to location (b or c).
≥ Ca@b@c.	Copy locations a through b at c and following.
≥ Fa@b@v.	Fill locations a through b with value v.
≥ Ia.	Inspect location a.
≥ Ja.	Jump to location a.
≥ Ma.	Modify memory starting at location a.
≥ O.	Display pseudo O register.
≥ Ov.	Set pseudo O register to value v.
≥ Pa@b.	Print locations a through b.
≥ Rn.	Display relocation register Rn.
≥ Rn@n.	Set relocation register Rn to value v.
≥ Sa@b@v.	Search locations a through b for value v.
≥ Sa@b@v@m.	Search for value v using mask word m.
≥ T.	Enable console interrupt (TRAP).
≥ Tn.	Enable console interrupt and enable interrupts
≥ X.	Display pseudo X register.
≥ Xv.	Set pseudo X register to value v.

F.3.10 Concordance (CONC)

Load CONC; set P = CONC zero; RUN
Select Options:

SI=	
B	BATCH
1	Keyboard
2	TTY
3	HSR
4	CR

SI=	
5	Unit 0
6	Unit 1
7	Unit 2
8	Unit 3

LO=	
L	List
1	TTY
2	D.P.
3	Cent.

F.3.11 OS Command Summary

	COMMAND	RESPONSE
1.	/ASSIGN	unit=device [, unit=device ...]
2.	/BATCH	device
3.	/BEGIN	address [, parameters ...]
4.	/CANCEL *program-name,base page,limits,core limits,flag,time P register, A register, X register, CPU status	
5.	/COMMENT	
6.	/DATE *date	[mm/dd/yy]
7.	/DECLARE *time	device, {UP,DOWN}
8.	/EXEC	program-name [,parameters ...]
9.	/JOB *date, time	
10.	/LIST *date, time *lu pu UP/DOWN errors	
11.	/NJOB *JOB/NJOB time, current time	

12. /RESUME [parameters ...]
*time

13. /STATUS
*program-name,base page limits,core limits,flag,time
P register, A register, X register, CPU

14. /TIME [hh: mm: ss]
*time

15. /TYPE

COMPUTER AUTOMATION, INC.
the NAKED MINI company

18651 Von Karman, Irvine, Calif. 92664
tel. 714-833-8830   TWX 910-595-1767