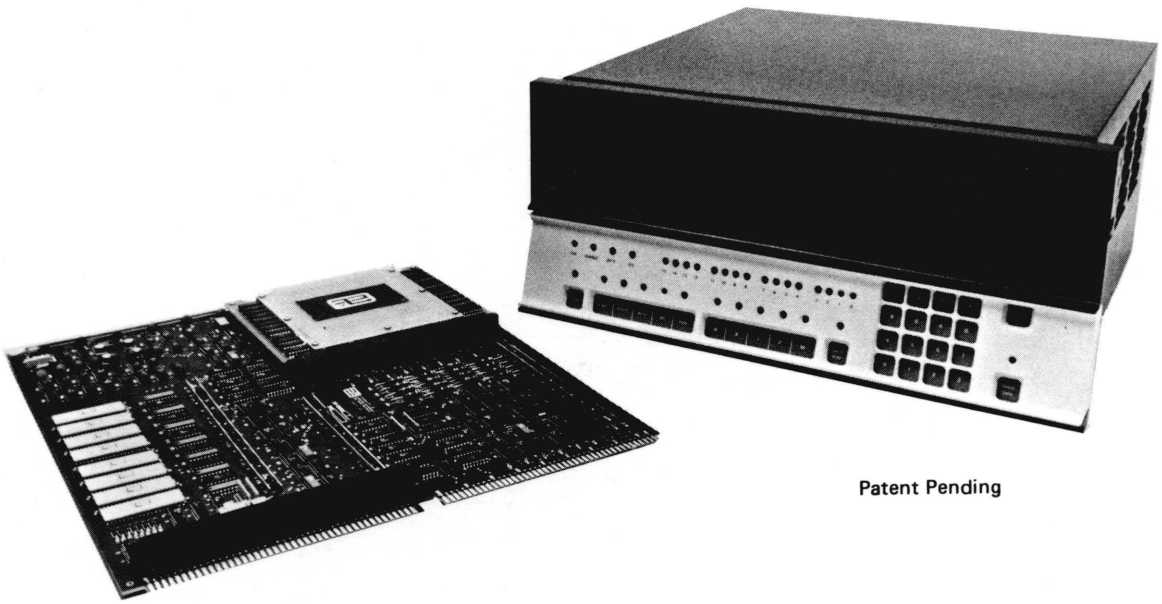
10077-00B1
SEPTEMBER 1973

NAKED MINI®/ALPHA LSI SERIES

PROGRAMMING REFERENCE MANUAL



Patent Pending

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	
1.1	INTRODUCTION	1-1
1.1.1	The ALPHA LSI Family	1-1
1.1.2	Upward Compatibility	1-1
1.1.3	General Features	1-1
1.2	THE NAKED MINI LSI CONCEPT	1-1
1.3	THE ALPHA LSI	1-2
1.4	CHARACTERISTICS	1-2
1.4.1	Processor	1-3
1.4.2	Instruction Set	1-3
1.4.3	Memory Addressing	1-4
1.4.4	I/O Structure	1-5
1.4.5	Processor Options	1-6
1.4.6	Plug-In Options	1-7
1.4.7	Peripheral Equipment	1-8
1.5	DATA HANDLING CHARACTERISTICS	1-8
1.5.1	Data Word Format	1-8
1.5.1.1	Bit Identification	1-8
1.5.1.2	Bit Values	1-9
1.5.1.3	Signed Numbers	1-9
1.5.1.4	Positive Numbers	1-9
1.5.1.5	Negative Numbers	1-9
1.5.2	Data Byte Format	1-10
1.5.2.1	Byte Mode Processing	1-11
1.5.2.2	Register Load	1-11
1.5.2.3	Arithmetic Operations	1-11
1.5.2.4	Data Packing	1-11
1.5.3	Memory Address Formats	1-12
1.5.3.1	Word Addressing	1-13
1.5.3.2	Byte Addressing	1-13
1.5.3.3	Indirect Addressing	1-14

TABLE OF CONTENTS (Cont'd)

Section		Page
	Section 2. CONSOLE	
2.1	INTRODUCTION	2-1
2.2	SWITCHES AND INDICATORS	2-1
2.3	MACHINE MODES	2-6
2.3.1	Stop Mode	2-6
2.3.2	Step Mode	2-7
2.3.3	Run Enable Mode	2-7
2.3.4	Run Mode	2-7
2.4	CONSOLE OPERATION	2-7
2.4.1	Console Preparation	2-7
2.4.2	Console Data Entry Procedure	2-8
2.4.3	Console Display Procedure	2-9
2.4.4	Program Execution	2-10
2.5	UNATTENDED OPERATION	2-10
	Section 3. INSTRUCTIONS AND DIRECTIVES	
3.1	INTRODUCTION	3-1
3.1.1	Instruction and Directive Classes	3-1
3.1.2	Symbolic Notation	3-1
3.1.3	Assembler Source Statement Fields	3-2
3.1.3.1	Label Field	3-2
3.1.3.2	Op Code Field	3-3
3.1.3.3	Operand Field	3-3
3.1.3.4	Comments Field	3-4
3.1.4	Arithmetic Operations and Overflow	3-4
3.1.5	Relocatability	3-5
3.2	MEMORY REFERENCE INSTRUCTIONS	3-6
3.2.1	Word Mode Operations and Instruction Format	3-6
3.2.1.1	Word Mode Direct Addressing	3-6
3.2.1.2	Word Mode Indirect Addressing	3-7
3.2.1.3	Word Mode Direct Indexed Addressing	3-7

TABLE OF CONTENTS (Cont'd)

Section		Page
	Section 3. INSTRUCTIONS AND DIRECTIVES (Cont'd)	
3.2.1.4	Word Mode Indirect Post-Indexed Addressing	3-7
3.2.1.5	Word Mode Summary	3-8
3.2.2	Byte Mode Operations and Instruction Format	3-8
3.2.2.1	Byte Mode Direct Addressing	3-9
3.2.2.2	Byte Mode Indirect Addressing	3-9
3.2.2.3	Byte Mode Direct Indexed Addressing	3-10
3.2.2.4	Byte Mode Indirect Post-Indexed Addressing	3-10
3.2.2.5	Byte Mode Summary	3-10
3.2.3	Arithmetic Memory Reference Instructions	3-10
3.2.4	Logical Memory Reference Instructions	3-11
3.2.5	Data Transfer Memory Reference Instructions	3-12
3.2.6	Program Transfer Memory Reference Instructions	3-12
3.3	DOUBLE-WORD MEMORY REFERENCE INSTRUCTIONS	3-14
3.3.1	Format	3-14
3.3.2	Instructions	3-15
3.4	IMMEDIATE INSTRUCTIONS	3-18
3.4.1	Format	3-18
3.4.2	Instructions	3-18
3.5	CONDITIONAL JUMP INSTRUCTIONS	3-19
3.5.1	Format	3-19
3.5.2	Microcoding	3-19
3.5.3	Arithmetic Conditional Jump Instructions	3-20
3.5.4	Control Conditional Jump Instructions	3-21
3.6	SHIFT INSTRUCTIONS	3-21
3.6.1	Operand Restrictions and Instruction Format	3-21
3.6.2	Arithmetic Shift Instructions	3-22
3.6.3	Logical Shift Instructions	3-22
3.6.4	Rotate Shift Instructions	3-23
3.6.5	Double Register (Long) Rotate Shift Instructions	3-24
3.6.6	Double Register (Long) Shift Instructions	3-25
3.7	REGISTER CHANGE INSTRUCTIONS	3-25
3.7.1	Format	3-25
3.7.2	A Register Change Instructions	3-25
3.7.3	X Register Change Instructions	3-26

TABLE OF CONTENTS (Cont'd)

Section		Page
3.7.4	OV Register Change Instructions	3-26
3.7.5	Multi-Register Change Instructions	3-27
3.7.6	Console Register Instructions	3-28
3.8	CONTROL INSTRUCTIONS	3-29
3.8.1	Format	3-29
3.8.2	Processor Control Instructions	3-29
3.8.3	Mode Control Instructions	3-30
3.8.4	Status Control Instructions	3-30
3.8.5	Interrupt Control Instruction	3-31
3.9	INPUT/OUTPUT INSTRUCTIONS	3-32
3.9.1	Control Input/Output Instructions	3-33
3.9.1.1	Sense Instructions	3-33
3.9.1.2	Select Instructions	3-33
3.9.2	Word Input/Output Instructions	3-34
3.9.2.1	Unconditional Word Input/Output Instructions	3-34
3.9.2.2	Conditional Word Input/Output Instructions	3-35
3.9.3	Byte Input Instructions	3-36
3.9.3.1	Unconditional Byte Input Instructions	3-36
3.9.3.2	Conditional Byte Input Instructions	3-36
3.9.4	Block Input/Output Instructions	3-37
3.9.5	Automatic Input/Output Instructions	3-39
3.10	ASSEMBLER CONTROL DIRECTIVES	3-42
3.10.1	Conditional Assembly Controls	3-42
3.10.2	Program Location Controls	3-43
3.11	DATA AND SYMBOL DEFINITION DIRECTIVES	3-44
3.11.1	Formats	3-44
3.11.2	Directives	3-44
3.12	PROGRAM LINKAGE DIRECTIVES	3-45
3.12.1	Formats	3-45
3.12.2	Directives	3-46
3.13	SUBROUTINE DEFINITION DIRECTIVES	3-46
3.14	LISTING FORMAT AND ASSEMBLER INPUT CONTROLS	3-47
3.15	USER-DEFINED OPERATION CODE DIRECTIVE	3-48

TABLE OF CONTENTS (Cont'd)

Section		Page
	Section 4. INPUT/OUTPUT AND INTERRUPT OPERATION	
4.1	INTRODUCTION	4-1
4.1.1	Discussion of Input/Output Operations	4-1
4.1.1.1	Control	4-1
4.1.1.2	Sense	4-2
4.1.1.3	Data Transmission	4-2
4.1.2	Interrupt Operations	4-5
4.1.2.1	Non-Input/Output	4-6
4.1.2.2	Input/Output	4-6
4.1.2.3	Word and Block Interrupts	4-6
4.2	NON-INTERRUPT INPUT/OUTPUT EXAMPLES	4-6
4.2.1	Control Instructions	4-9
4.2.2	Unconditional Instructions	4-9
4.2.3	Conditional Instructions	4-10
4.2.4	Block Transfer Instructions	4-10
4.2.5	Automatic Transfer Instructions	4-11
4.3	INTERRUPT STRUCTURE EXAMPLES	4-11
4.3.1	General Interrupt Handling	4-11
4.3.2	Examples of Initialization and Enabling Sequences	4-12
4.3.3	Examples of Interrupt Instructions	4-13
4.4	INTERRUPT LATENCY	4-15
4.4.1	Interrupt Service	4-15
4.4.2	Priority Resolution	4-16
	Section 5. PROCESSOR OPTIONS	
5.1	TELETYPE	5-1
5.1.1	General Discussion	5-1
5.1.2	Half-Duplex Usage	5-1
5.1.3	Table of Half-Duplex Teletype Instructions	5-3
5.1.4	Full-Duplex Usage	5-6
5.1.5	Table of Full-Duplex Teletype Instructions	5-8
5.2	REAL-TIME CLOCK	5-12
5.2.1	Discussion of Usage	5-12
5.2.2	Summary Table	5-13

TABLE OF CONTENTS (Cont'd)

Section		Page
5.3	AUTOLOAD	5-14
5.4	POWER FAIL/RESTART	5-15
5.4.1	General	5-15
5.4.2	Power Fail	5-15
5.4.3	Restart	5-15
5.4.4	Interrupt Control Option	5-16
5.4.5	Programming Examples	5-16

Appendix A. HEXADECIMAL TABLES

Appendix B. RECOMMENDED DEVICE AND INTERRUPT ADDRESSES

Appednix C. INSTRUCTION SET BY CLASS

Appendix D. INSTRUCTION SET IN ALPHABETICAL ORDER

Appendix E. INSTRUCTION SET IN NUMERICAL ORDER

Appendix F. ALPHA LSI EXECUTION TIMES

F.1	GENERAL	F-1
F.2	MEMORY PARAMETERS	F-1
F.3	LSI-1 EXECUTION TIME ALGORITHMS	F-2
F.4	LSI-2 EXECUTION TIME ALGORITHMS	F-8
F.5	ALPHA LSI FAMILY INSTRUCTION EXECUTION TIME	F-16
F.6	MAXIMUM I/O TRANSFER RATES	F-24

TABLE OF CONTENTS (Cont'd)

Section		Page
	Appendix G. SOFTWARE SUMMARY	
G.1	INTRODUCTION	G-1
G.2	BOOTSTRAP	G-2
G.3	SOFTWARE	G-2
G.3.1	Autoload	G-2
G.3.2	Binary Load (BLD)	G-3
G.3.3	Binary Dump/Verify (BDP/VER)	G-3
G.3.4	Object Load	G-4
G.3.5	BETA-4 Assembler	G-4
G.3.6	BETA-8 Aseembler	G-4
G.3.7	OMEGA Conversational Assembler	G-5
G.3.8	Source Tape Preparation Program	G-6
G.3.9	Debug (DBG)	G-7
G.3.10	Concordance (CONC)	G-8
G.3.11	OS Command Summary	G-8

LIST OF ILLUSTRATIONS

Figure		Page
1-1	Data Word Bit Identification	1-10
1-2	Byte Storage, Two Bytes Per Word	1-10
1-3	Data in Memory, One Byte Per Word	1-12
1-4	Data in Memory, Two Bytes Per Word	1-13
1-5	Basic Word Address Format	1-14
1-6	Byte Address Format	1-14
1-7	Indirect Address Pointer Format	1-14
2-1	ALPHA 16 LSI Console	2-3
3-1	Instruction and Directive Classes	3-1
3-2	Source Statement Format	3-2
3-3	Arithmetic Overflow	3-5
3-4	Word Mode Memory Reference Instruction Format	3-6
3-5	Word Mode Addressing Summary	3-8
3-6	Byte Mode Memory Reference Instruction Format	3-9
3-7	Byte Mode Addressing Summary	3-11
3-8	Double Word Memory Reference Format	3-14
3-9	Divide	3-15
3-10	Multiply and Add	3-16
3-11	NRM Shift Path	3-17
3-12	Immediate Instruction Format	3-18
3-13	JOC Jump On Condition Format	3-19
3-14	JOC Microcode Bit Functions	3-20
3-15	Conditional Jump Format	3-20
3-16	Single Register Shift Format	3-21
3-17	Double Register (Long) Shift Format	3-21
3-18	Arithmetic Right Shift	3-22
3-19	Arithmetic Left Shift	3-22
3-20	Logical Right Shift	3-22
3-21	Logical Left Shift	3-22
3-22	Rotate Right	3-23
3-23	Rotate Left	3-23
3-24	Long Left Shift	3-24
3-25	Long Right Shift	3-24
3-26	Long Rotate Right	3-25
3-27	Long Rotate Left	3-25
3-28	Register Change Format	3-25
3-29	Control Format	3-29
3-30	Computer Status Word Format	3-30
3-31	Single Word Input/Output Instruction Format	3-32
3-32	Block Input/Output Instruction Format	3-38
3-33	Automatic Input/Output Instruction Format	3-39
3-34	In-line Auto I/O Instruction Sequence	3-40

LIST OF ILLUSTRATIONS (Cont'd)

Figure		Page
3-35	Interrupt Location Auto I/O Instruction Sequence	3-41
3-36	Begin Conditional Assembly Directives Format	3-42
3-37	End Conditional Assembly Directive Format	3-42
3-38	Location Control Directive Format	3-43
3-39	Data and Symbol Definition Directive Format	3-44
3-40	Program Linkage Directive Formats	3-45
3-41	Subroutine Definition Directive Formats	3-46
3-42	Title Directive Format	3-47
4-1	Initialization and Unconditional Output to Line Printer	4-7
4-2	Unconditional Character Read from Teletype Paper Tape Reader	4-7
4-3	Initialization and Conditional Control of Line Printer	4-7
4-4	Conditional Input from Teletype Keyboard with Auto-Echo	4-8
4-5	Uninterruptable Block Output to Line Printer	4-8
4-6	Automatic Byte Input from Card Reader	4-9
4-7	Line Printer Interrupt Initialization Sequence	4-12
4-8	Real-Time Clock Interrupt Initialization Sequence	4-13
4-9	Line Printer Interrupt Instructions	4-13
4-10	Real-Time Clock Interrupt Instructions	4-14
4-11	Standard Interrupt Priorities	4-17
5-1	Program-Controlled Data Output to Half-Duplex Teletype	5-2
5-2	Program-Controlled Data Input from TTY Paper Tape Reader	5-2
5-3	Program-Controlled Data Input from Full-Duplex Teletype	5-6
5-4	Automatic Interrupt Data Input from Full-Duplex Teletype	5-7
C.1	Class 1 - Single-Word Memory Reference Instruction Format	C-1
C.2	Class 2 - Double-Word Memory Reference Instruction Format	C-1
C.3	Class 3 - Byte Immediate Instruction Format	C-1
C.4	Class 4 - Conditional Jump Instruction Format	C-2
C.5	Class 5 - Register Shift Instruction Format	C-2
C.6	Class 6 - Register Change Insturction Format	C-2
C.7	Class 7 - Input/Output Instruction Format	C-2
C.8	Class 8 - JOC Jump-On-Condition Instruction Format	C-2
E.1	Single-Word Memory Reference Instruction Machine Code Format	E-1
E.2	Double-Word Memory Reference Instruction Machine Code Format	E-1
E.3	Byte Immediate Instruction Machine Code Format	E-2
E.4	Conditional Jump Instruction Machine Code Format	E-2
E.5	Single-Register Shift Instruction Machine Code Format	E-3
E.6	Double-Register Shift Instruction Machine Code Format	E-3
E.7	Register Change Instruction Machine Code Format	E-3
E.8	Control Instruction Machine Code Format	E-3
E.9	Input/Output Instruction Machine Code Format	E-4
E.10	Automatic Input/Output Instruction Machine Code Format	E-4
E.11	Block Input/Output Instruction Machine Code Format	E-4

LIST OF TABLES

Table		Page
2-1	Console Switches and Indicators	2-1
A-1	Hexadecimal-Decimal Conversions	A-2
A-2	8-BIT ASCII Teletype Codes	A-3
B-1	Recommended Device Addresses	B-2
B-2	Scratchpad/Page 0 Recommended Interrupt Address Map	B-3
B-3	Device Address 0 Command Summary	B-4
F-1	LSI Family Memory Parameters	F-1
F-2	LSI-1 Execution Time Algorithms	F-2
F-3	LSI-2 Execution Time Algorithms	F-9
F-4	LSI-1 Address Calculation Times	F-16
F-5	LSI-2 Address Calculation Times	F-17
F-6	ALPHA LSI Family Instruction Execution Times	F-18
F-7	ALPHA LSI Family Maximum Data Transfer Rates	F-24

Section 1
GENERAL DESCRIPTION

1.1 INTRODUCTION

The ALPHA LSI and NAKED MINI 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 The ALPHA LSI Family

The NAKED MINI/ALPHA LSI is not just one computer that can be packaged with or without a chassis, power supply and console. Instead, it is an integrated family of compatible components -- two central processors; three kinds of memories in fourteen sizes and three speeds; peripheral controllers; computer options, general purpose interfaces; etc. -- which can be combined in a multitude of configurations to match a wide range of needs.

The two central processors are referred to as the NAKED MINI LSI type 1 (LSI-1) and the NAKED MINI LSI type 2 (LSI-2). The LSI-1 and LSI-2 Processors feature the same basic architecture, instruction set and I/O capabilitites. They differ in terms of performance wherein the LSI-2 is faster than the LSI-1. Both Processors are plug-to-plug compatible and, except for timing differences, programs will execute properly in either Processor without change.

The memories that are available are: Core 980, Core 1200, Core 1600 (standard) and semiconductor - SC1200. The numbers define the full cycle time of the memory and each memory type can be interleaved.

The user can mix memories of varying speeds, sizes, and technologies with either Processor to obtain the best price/performance margin possible.

1.1.2 Upward Campatibility

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.

1.1.3 General Features

The ALPHA LSI computer family features a 16-bit word format and 168 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 LSI CONCEPT

The NAKED MINI LSI type 1 computer consists of the Processor and first memory module on one printed circuit board. The NAKED MINI LSI type 1 is a complete stand alone computer without a chassis, motherboard, power supply or operators console.

The NAKED MINI LSI type 1 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.

The NAKED MINI LSI type 2 computer consists of a full board Processor module and one or more memory modules, a motherboard and a chassis. Like the LSI-1, the LSI-2 Processor depends on the system power supply for power and a system control panel for operational control signals.

1.3 THE ALPHA LSI

Take a NAKED MINI LSI type 1 or 2 computer and add a power supply module, a motherboard, a chassis and an operator's console and you get the ALPHA LSI computer. The Motherboard interconnects the NAKED MINI 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 for LSI-1; 150 nanoseconds for LSI-2.

Direct Memory access (Standard) provides data transfer rates up to 1,020,000 words per second in a single memory bank or 1,666,667 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 168 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. Jump 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, multilevel 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.

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. 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.

Data transfer rates are discussed in Appendix F of this manual.

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 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 LSI computer printed circuit board. None of these options interface directly with the motherboard.

1. Teletype/CRT Modem Interface.

   Interfaces a modified ASR-33 or ASR-35 Teletype or CRT terminal or modem 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. Either half or full-duplex operation is selectable on command.

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 reads from the ROM when the AUTO switch is activated.

1.4.6 Plug-In Options

Locations are provided within the ALPHA 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: asynchronous 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.

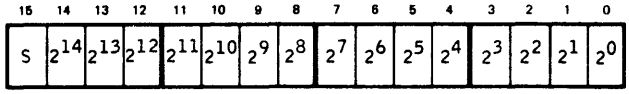


Figure 1-1. Data Word Bit Identification

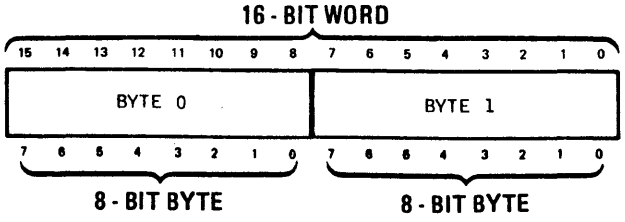


Figure 1-2. Byte Storage, Two Bytes Per Word

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 address and operand cycles 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.

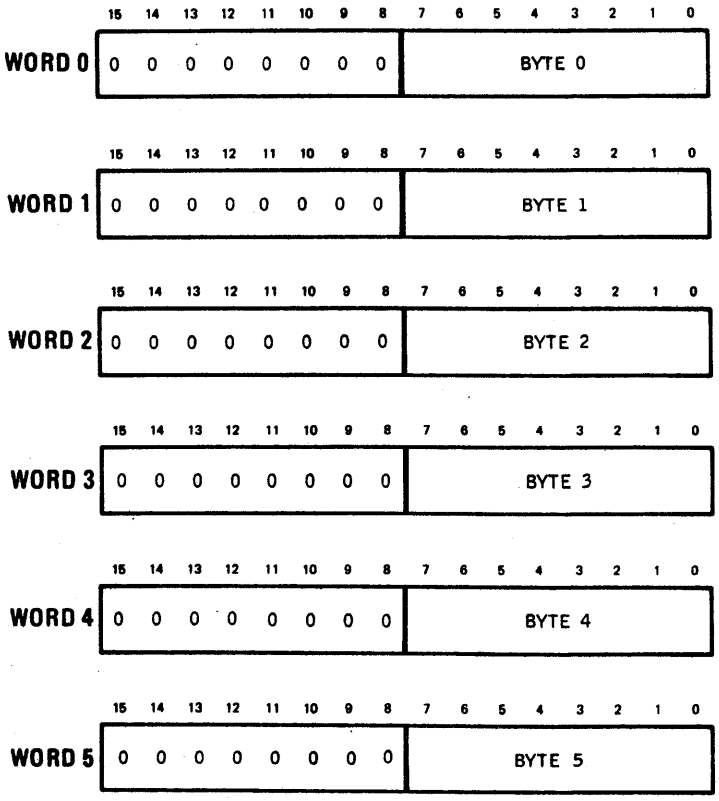


Figure 1-3. Data in Memory, One Byte Per Word

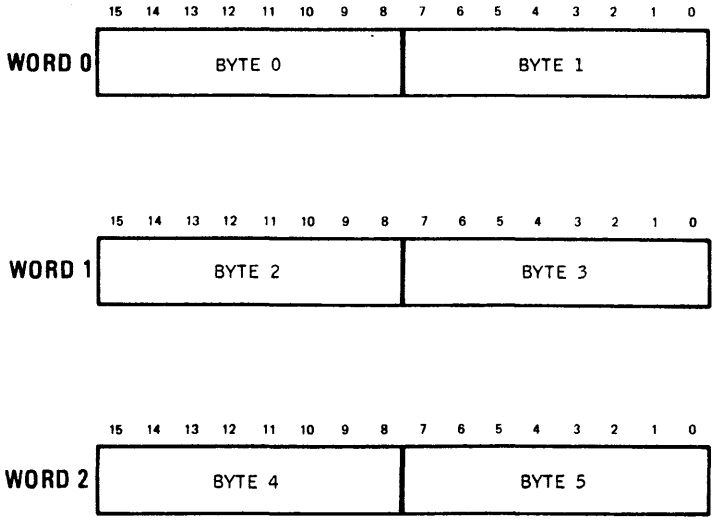


Figure 1-4. Data in Memory, Two Bytes Per Word

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.

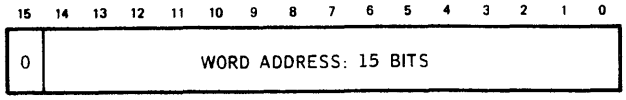


Figure 1-5. Basic Word Address Format

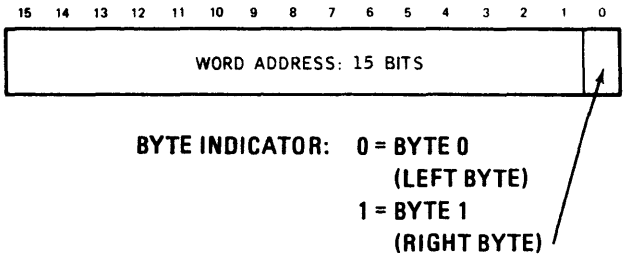


Figure 1-6. Byte Address Format

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 multi-level indicator. If bit 15 contains a 1 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.

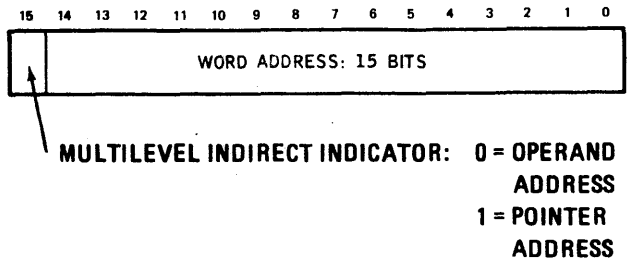


Figure 1-7. Indirect Address Pointer Format

Section 2
CONSOLE

2.1 INTRODUCTION

The ALPHA 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 LSI Console. All switches and indicators are listed and explained in table 2-1.

NOTE

All console switches, except the Console Enable switch, 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).

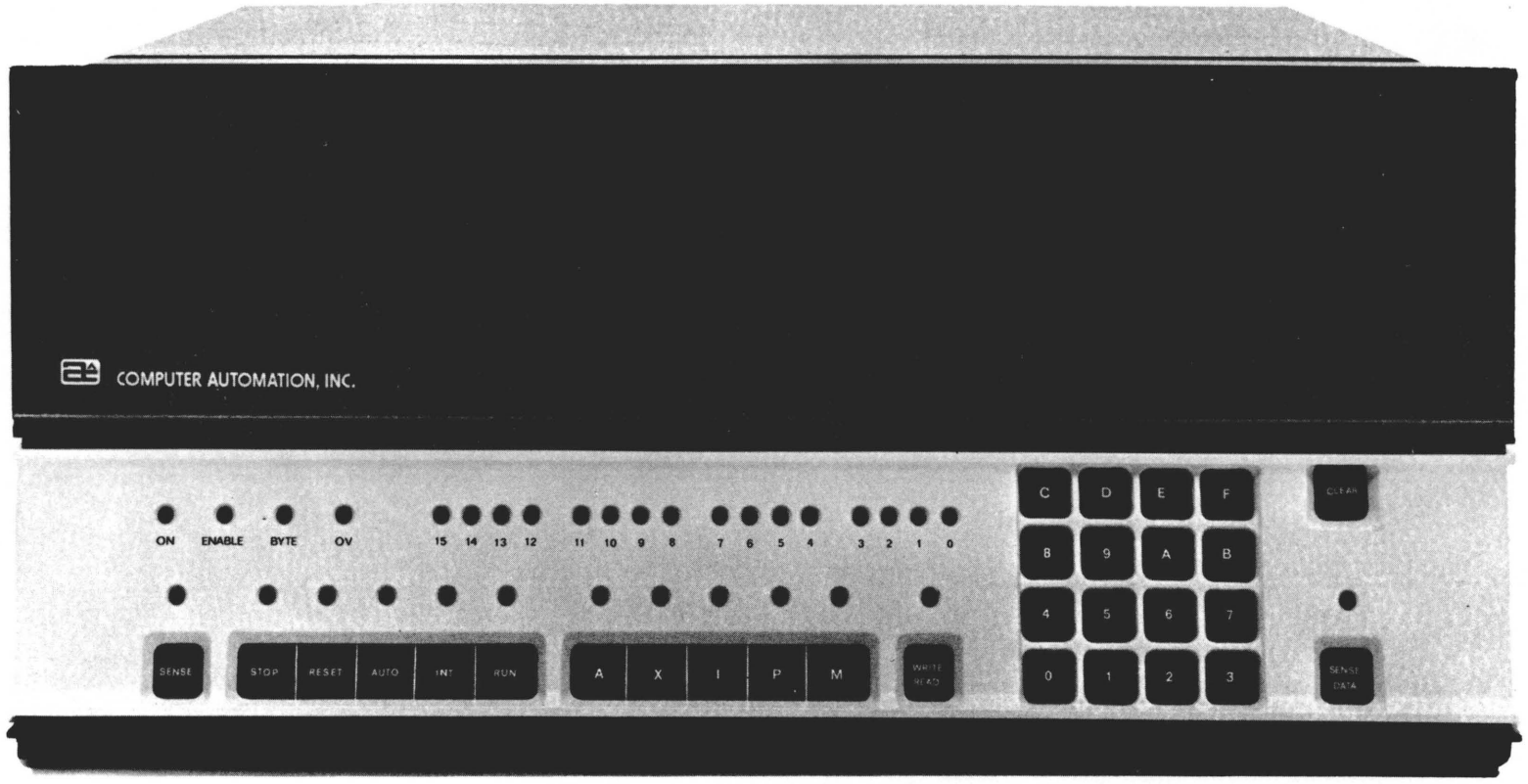


Figure 2-1. ALPHA 16 LSI Console

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. Interrupts are not serviced while in step mode.

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 LSI Console is used for initial start-up, program debug, and trouble-shooting. 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 chassis. 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.

The instructions and directives acceptable to the BETA Assembler are described in detail in the remainder of this section. The following conventions apply:

1. Square brackets [ ] enclose elements which are optional and may be included or omitted as required.

2. Two or more elements separated by a vertical bar | indicates a choice must be made from the enclosed elements.

3. A right square bracket followed by an elipses (]...), indicates that the enclosed element may be repeated an arbitrary number of times.

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 may 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.

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.

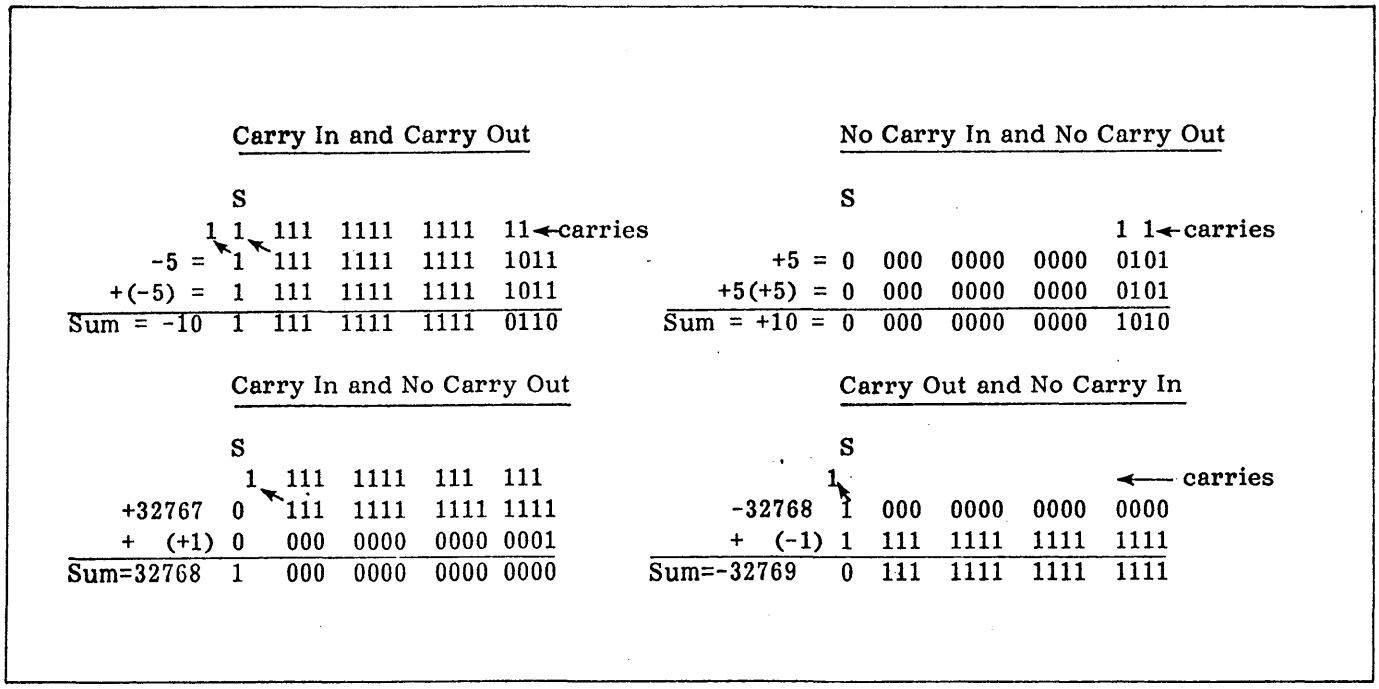


Figure 3-3. Arithmetic Overflow.

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.

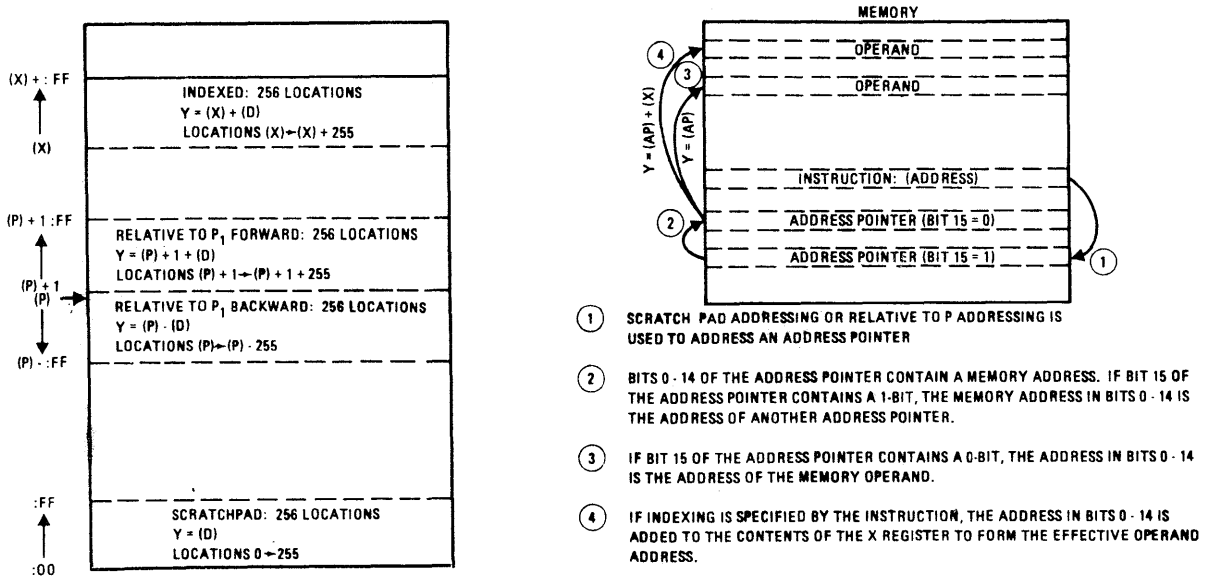
3.2.1.4 Word Mode Indirect Post-Indexed Addressing

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 is shown in Figure 3-5.



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 directive) 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.

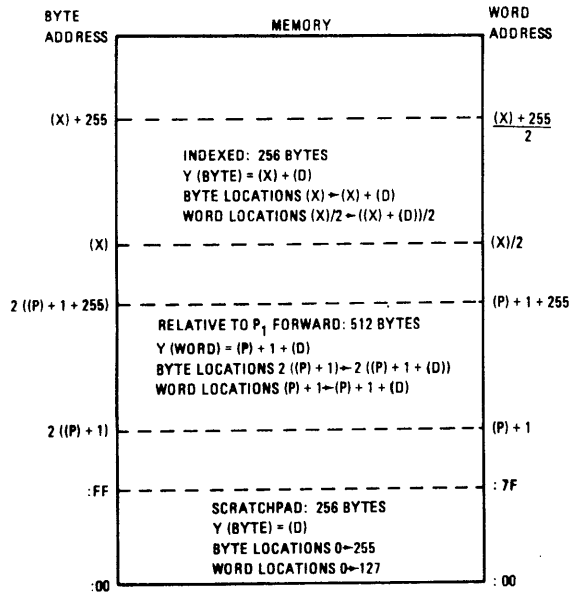
Because the Scan Memory (SCM) instruction always uses indirect post-indexed addressing, the assembler automatically generates the necessary machine code and does not allow @ or * operators on the associated operand expression. When performing byte scans, the operand expression for this instruction should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

3.2.2.5 Byte Mode Summary

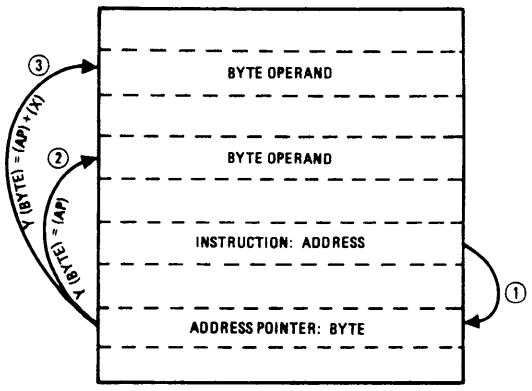
A summary of Byte Mode addressing is shown in Figure 3-7.

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.



Direct Addressing



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.

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

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 significant 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.

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.

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 in which case, when the increment causes a zero result, an Echo signal is generated and sent to the interrupting device. The interrupting device uses the Echo signal to develop an EOB (End-of-Block) interrupt. Under these conditions a skip does not occur and OV is unaffected. (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 value of the effective memory location plus one.

NOTE

JST is often used as an interrupt instruction. When used as such, all interrupts under EIN/DIN control are automatically disabled upon instruction execution. (See Sec. 4.3).

SCM SCAN MEMORY. Compares the A register with the area of memory (Buffer) 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. Thus, the data buffer is scanned in descending order, beginning with the highest memory location and ending with the lowest (Base Address). 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 SCM instruction again. If a match is not found by the time X reaches zero, a one place skip occurs and the instruction terminates.

3.2.5 Data Transfer Memory Reference Instructions

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 significant 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.

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.

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.

3.3.2 Instructions

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

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 fractional 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.

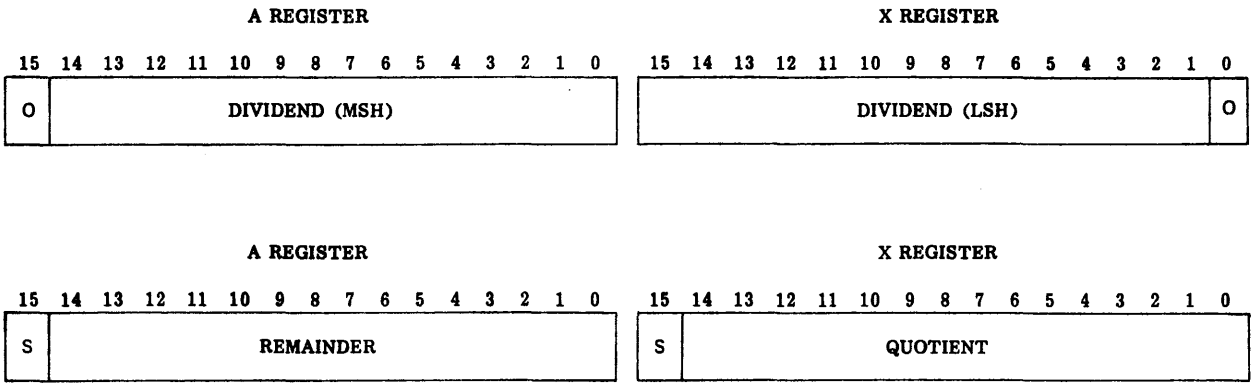


Figure 3-9. Divide

MPY MULTIPLY AND ADD. Multiplies the contents of the X register by the contents of the memory location addressed by Expression 1 and adds the contents of the A register to the product. The address pointer may be direct or indirect and occupies the second word of the instruction in memory.

Prior to execution of the MPY instruction, the X register contains the signed full-word multiplicand, the addressed memory location contains the full-word multiplier, and the A register contains the "offset" to be added. (Refer to Figure 3-10.) Multiplier and offset must be positive or zero, the multiplicand may be either positive, negative 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. The contents of OV prior to execution of MPY will be returned in the Least Significant Bit (Bit 0) of the X register.

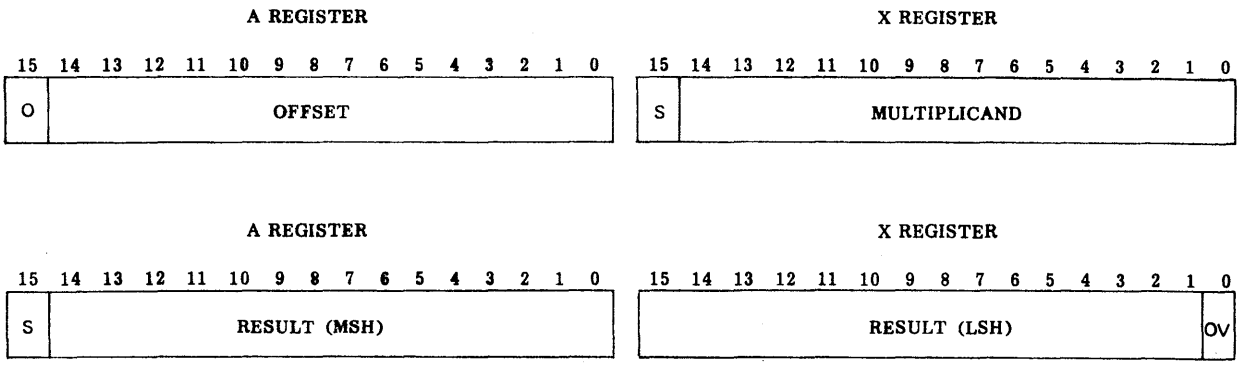


Figure 3-10. Multiply and Add

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 exhausted. The exponent (count cell), addressed by Expression 1, is decremented once for each shift until normalization occurs. The address of the exponent may be direct or indirect and occupies the second word of the instruction in memory.

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 set (=1). In either case, the exponent will be drecremented 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. A normalize operation with a count of zero (Expression 2) provides a test for normalization without effecting the contents of the registers.

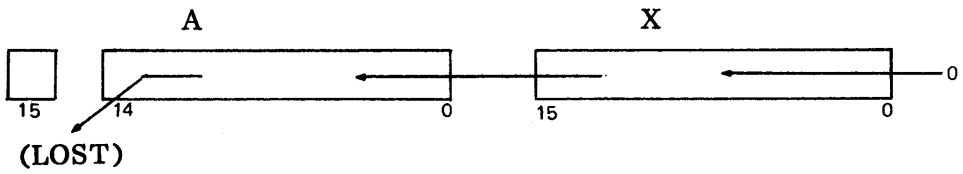


Figure 3-11. NRM Shift Path

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

AAI — ADD TO A IMMEDIATE. The operand is added to the contents of the A Register. OV is set if arithmetic overflow occurs.

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

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

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.

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.

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.

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.

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

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

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 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 within -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.

JOC :XX,ADR

B_5 B_4 B_3 B_2 B_1 B_0

	AND GROUP ($B_5 = 1$)	OR GROUP ($B_5 = 0$)
$B_4 = 1$	$X \neq 0$	$x = 0$
$B_3 = 1$	SENSE on	SENSE off
$B_2 = 1$	OV reset	OV set (resets OV)
$B_1 = 1$	$A \neq 0$	$A = 0$
$B_0 = 1$	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.

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

JAZ — JUMP IS A ZERO. Jump occurs if the A Register is zero.

JAN — JUMP IS A NOT ZERO. Jump occurs if the A Register is not zero.

JAL — JUMP IF A LESS THAN OR EQUAL TO ZERO. Jump occurs if the A Register is less than or equal to zero.

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

JXZ JUMP IS X ZERO. Jump occurs if the X Register is zero.

JXN JUMP IS X NOT ZERO. Jump occurs if the X Register is not zero.

3.5.4 Control Conditional Jump Instructions

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

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

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

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

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 thru 8.		

Figure 3-16. Single Register Shift Format

LABEL	OP-CODE	EXPRESSION	COMMENTS
	EXPRESSION: must be absolute and in the range 1 thru 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.

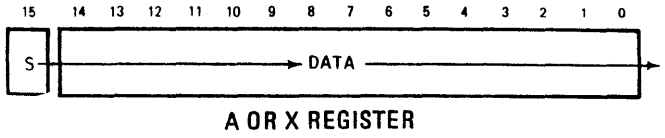


Figure 3-18. Arithmetic Right Shift

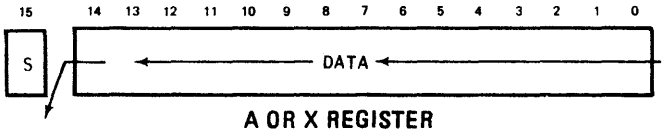


Figure 3-19. Arithmetic Left Shift

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.

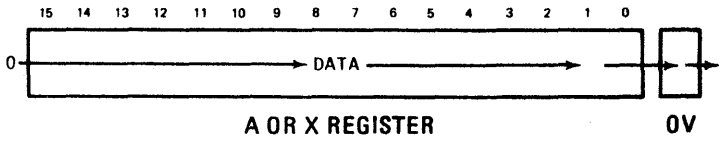


Figure 3-20. Logical Right Shift

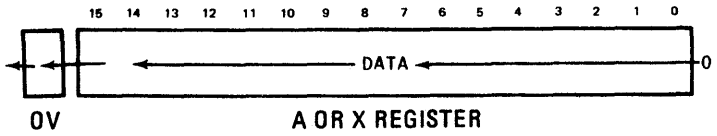


Figure 3-21. Logical Left Shift

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.

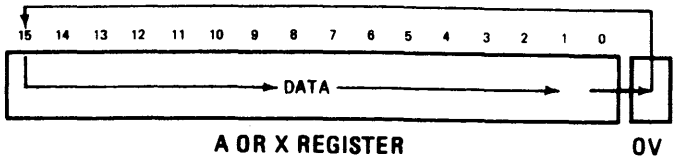


Figure 3-22. Rotate Right

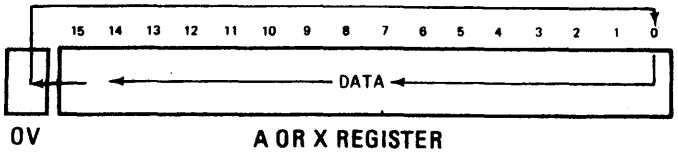


Figure 3-23. Rotate Left

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) Logical Shift Instructions

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

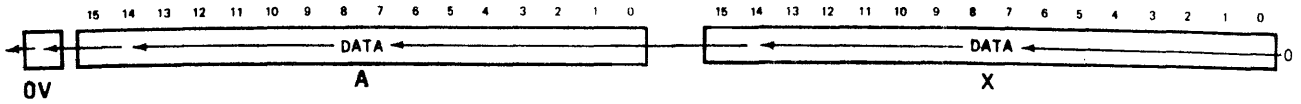


Figure 3-24. Long Left Shift

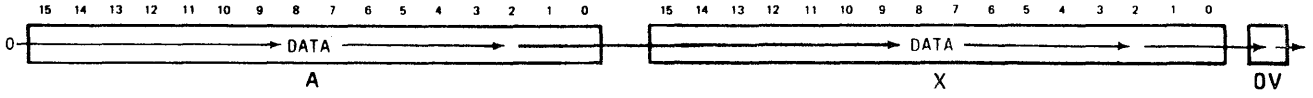


Figure 3-25. Long Right Shift

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 , X is shifted into A and A 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 , A is shifted into X and X 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) Rotate Shift Instructions

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

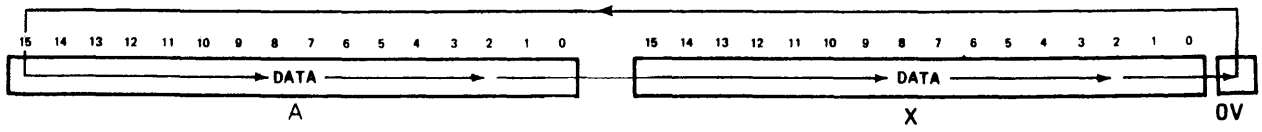


Figure 3-26. Long Rotate Right

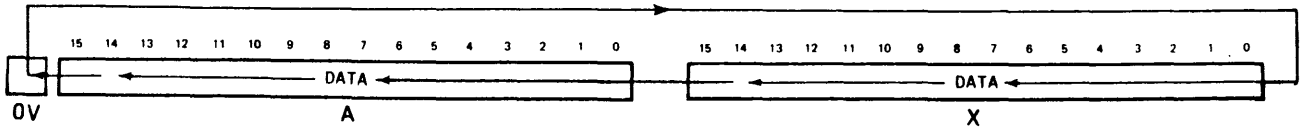


Figure 3-27. Long Rotate Left

LRL LONG ROTATE LEFT. Contents of A and X Registers are shifted left through OV n places. OV is shifted into X , X is shifted into A and A 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 , A is shifted into X and X 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. The Register Change Instruction format is shown in Figure 3-28.

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

EXPRESSION: there is no expression in the operand field, except for the BAO and BXO instructions, where it must be absolute and in the range 0 through 15.

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

ZXR — ZERO X REGISTER. Sets the X Register to zero.

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

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

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

NXR — NEGATE X REGISTER. Performs two's complement of 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.

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

3.7.4 OV Register Change Instructions

SOV — SET OVERFLOW. Sets OV (to one).

ROV — RESET OVERFLOW. Resets OV (to zero).

COV COMPLEMENT OVERFLOW. Complements OV.

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

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

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.

BAO BIT OF A TO OVERFLOW. Bit n of the A Register is copied into OV. The A Register is unchanged. Bit n is specified in the operand field.

BXO BIT OF X TO OVERFLOW. Bit n of the X Register is copied into OV. The X Register is unchanged. Bit n is specified in the operand field.

3.7.5 Multi-Register Change Instructions

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

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

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

TAX TRANSFER A TO X. Transfers the contents of the A Register to the X Register. The A Register is unchanged.

TXA TRANSFER X TO A. Transfers the contents of the X Register to the A Register. The X Register is unchanged.

EAX EXCHANGE A AND X. Enchange the contents of the A and X Registers.

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.

NRA NOR OF A AND X TO A. Performs logical NOR of the A and X ($\overline{A}$ AND $\overline{X}$) 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 ($\overline{A}$ AND $\overline{X}$) Registers and places the result in X. The A Register is unchanged.

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

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

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

IPX INCREMENT P TO X. Adds one to the value of the current program counter and places the result in X. The P Register is unchanged.

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

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.

NOTE

The Console Register instructions defined in this section are in fact "pre-defined" general I/O instructions requiring no operand.

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:

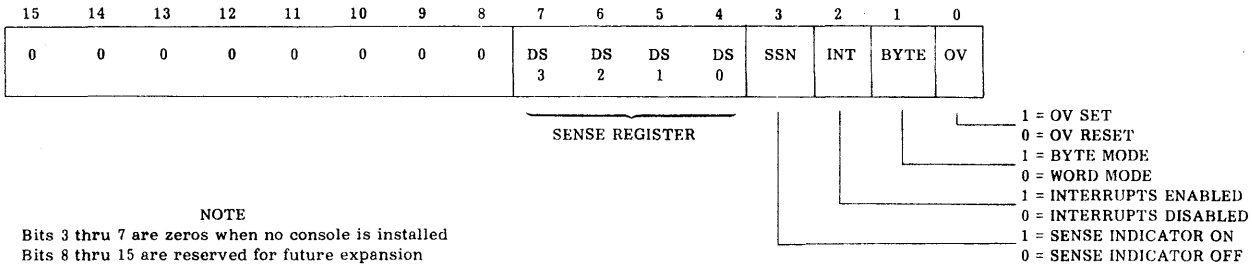


Figure 3-30. Computer Status Word Format

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.

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.

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.

NOTE

The Status Control instructions defined in this section are are in fact "pre-defined" general I/O instructions requiring no operand.

3.8.5 Interrupt Control Instructions

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.

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.

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. The Console interrupt is disabled when a Console interrupt or TRAP is serviced.

CID CONSOLE INTERRUPT DISABLE. Disables the Console interrupt.

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.

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.

TRP — TRAP. Generates an interrupt to the Console interrupt location if interrupts are enables 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. The Console interrupt is disabled when the TRAP is serviced.

NOTE

The Interrupt Control instructions described in this section (except EIN and DIN) are in fact "pre-defined" general I/O instructions requiring no operands.

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 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 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 affected 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.

NOTE

When A Select type instruction is used to turn off interrupts that may be pending, it should be preceded by a SIN 1 instruction to disable Processor recognition of the pending interrupt. This is necessary since the Processor examines interrupt requests prior to the Select taking effect and will therefore respond to the interrupt even though it is no longer pending.

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 X 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 X Register and logically ANDed with the previous contents of the X Register. The results are placed in the X 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 X 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]	DATA	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 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 base (the low-order memory location) of the data buffer.

BIN BLOCK IN. Sense 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. Sense 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 or DATA}	EXPRESSION 4		[COMMENTS]

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 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 (SEN) 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 (start 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	Address Pointer (start address -1)
	I+3	Unused
End-of-Block Interrupt Location	I+4	JST EOBSUB (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 unconditioanlly 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	[EXPRESSION]	[COMMENTS]

EXPRESSION: must be an absolute value of zero (False) or non-zero (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 skipped (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 discuessed 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	EXPRESSION	[COMMENTS]
[LABEL]	RES	EXPRESSION 1 [, EXPRESSION 2]	[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 bytes of sequential memory words. Trailing characters positions are filled with blanks (:A0) to complete full words. The expression must be a character string 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.

[LABEL]	NAM or EXTR	EXPRESSION 1 [, EXPRESSION 2,]...	[COMMENTS]
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 assigned to the external symbol 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 CHARACTERS NOT EXCEEDING 51 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. Comment lines may appear anywhere in 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:

```
                .
                .
                .
          LXP   COUNT
          BIN   DA,FC
          DATA  BUF - 1
                .
                .
                .
```

or:

```
                .
                .
                .
          LXP   COUNT
          BOT   DA,FC
          DATA  BUF - 1
                .
                .
BUF       RES   COUNT
                .
                .
```

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  BUF - 1 (Word)
          JMP   EOB
          JMP   SENSE
                .
                .
                .
```

or:

```
SENSE    SEN     DA,FC
         JMP     $-1
         AOB     DA,FC
         DATA    Negative Data Count (Byte)
         BAC     BUF - 1 (Byte)
         JMP     EOB
         JMP     SENSE
                 .
                 .
                 .
BUF      RES     COUNT
                 .
                 .
                 .
```

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-Blcok 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 (end-of-block) location is generated by the peripheral controller when the buffer is completely transferred.

4.1.2. Interrupt Operations

Interrupts constitute a means of reacting quickly 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. Although any of the instructions in the ALPHA's repertoire could be used in the reserved locations as interrupts instructions, only certain of them are generally useful - IMS, JMP, 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 (IMS) is ued 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 of 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 (JST) is ued 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    SEL             7,4             Initialize Teletype
            .
            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 Type 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
                .
                .
                .
                LXP             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             Buffer Byte Count
                BAC         BUF-1           Buffer 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).

NOTE

When a Select type instruction is used to turn off interrupts that may be pending, it should be preceded by a SIN 1 instruction to disable Processor recognition of the pending interrupt. This is necessary since the Processor examines interrupt requests prior to the Select taking effect and will therefore respond to the interrupt even though it is no longer pending.

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 affeect 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.

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 pointe (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 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 or during DMA operations.

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
.
.
.
```

Example 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 Examples 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,
                          .               Disable CPU Interrupts
                          .
                          .
Main memory   SUB         ENT
                          .
                          .
                          RTN     SUB
                          .
                          .
              BUF         RES     40
                          .
                          .
                          .
```

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,
                            .                 Diable 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     STATUS    Save Status
                            STX     XSAVE     Save X Register
                            LAM     100       Reset
                            STA     COUNT     RTC counter COUNT
                            .
                            .
                            .                 Perform specified Maintenance
                                              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 and the instruction address will be treated as a word address. 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 interrupts 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 Status Inhibit (SIN) instruction 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 on the following page in Figure 4-11.

ABSOLUTE PRIORITY		INTERRUPT ADDRESS
1	POWER FAIL	: 001C (: 011C)
2	TRAP INTERRUPT	: 001E (: 011E)
3	CONSOLE INTERRUPT	: 001E (: 011E)
4	MEMORY PARITY (IL1)	: 0012 (: 0112)
5	INTERRUPT LINE 1 (IL1)	: 0002 (: 0102)
6	INTERRUPT LINE 2 (IL2)	: 0006 (: 0106)
7	RTC SYNC INTERRUPT (IUR)	: 001A (: 011A)
8	CLOCK INTERRUPT (IUR)	: 0018 (: 0118)
9	TTY END-OF-BLOCK (IUR)	: 0006 (: 0106); OPTIONAL : 0026 (: 0126)
10	TTY WORD (IUR)	: 0002 (: 0102); OPTIONAL : 0022 (: 0122)
11	SLOT B200	
12	SLOT B100	
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	

IUR CHAIN

Slots B through E 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/CRT/MODEM CONTROLLER

5.1.1 General Discussion

The Teletype/CRT/Modem (TTY/CRT) option interfaces a CRT, Modem or 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 user terminal. 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. A CRT provides keyboard entry and display.

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

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

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

Input to the computer is accomplished via the TTY/CRT keyboard or the TTY 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/CRT for printing or display.

The Teletype or CRT can be operated in either half-duplex or full-duplex mode. The Initialize instruction (SEL7,4) puts the controller 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 controller in the full-duplex mode.

5.1.2 Half-Duplex Usage

Half-duplex controller operations involve either input from or output to the terminal device, but not simultaneously. Use of the Auto Echo feature causes input from the device to be automatically "echoed" back for printing or display, 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
        .
        .
        SWM                 Restore Word Addressing Mode
        .                   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
        .
        .
        SWM                 Restore Word Addressing Mode
        .
        .
```

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

The standard Word Interrupt Location for half-duplex controller operation is location : 0002. The controller interrupts to this location when the Word Transfer Mask is set, interrupts are enabled and the terminal device 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 terminal device is location : 0006. The controller 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 controller interrupts can be relocated to locations : 0102 and : 0106 or : 0122 and : 0126.

5.1.3 Table of Half-Duplex Controller Instructions

Instruction	Function
SEL 7,0	Enable Auto Echo. This instruction causes all input to be echoed back to the source terminal printing or display.
SEL 7,1	Select Keyboard. This instruction resets the Buffer Ready flip-flop and puts the controller in the read mode.
SEL 7,2	Step Read. This instruction causes the character under the read station on the TTY paper tape reader to be read. The tape is then advanced one character. The reader switch on the Teletype must be in the START position. The Buffer Ready flip-flop is reset.
SEL 7,3	Select Continuous Read. This instruction causes the TTY paper tape reader to read continuously until the reader is stopped or the tape runs out. The reader switch must be in the START position. The Buffer Ready flip-flop is reset. Execution of any other SEL instruction resets Continuous Read.
SEL 7,4	Initialize Controller. This instruction resets the control flip-flops, stops the oscillator and puts the controller in a static marking condition. The Buffer Ready flip-flop 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 controller to enable generation of interrupts upon establishment of a Buffer Ready condition.
SEL 7,6	Enable Echo/EOB Interrupt. This instruction sets an enable flip-flop in the controller 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 controller by resetting the enable flip-flops.
SEN 0,4	Sense Teletype Controller Installed. This instruction senses if the Teletype controller is installed on the Option Board. If the option is installed (true response) a one-word skip occurs.
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 Controller Not Busy. This instruction senses the state of the controller and generates a one-word skip if it is not processing a character.
SEN 7,4	Sense Clear to Send. This instruction senses the clear-to-send line from a CRT or Modem and generates a one-word skip if true. (This feature is available only with the EIA RS232C/ CCITT interface option.)
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 Parity Error. This instruction senses the occurrance of a parity error during the most recent input operation and generates a one-word skip if an error occurred. (This instruction requires prior strapping of parity option.)
SEN 7,7	Sense Full Duplex Mode Enabled. This instruction senses the state of the Full Duplex Enable flip-flop and generates a one-word skip if it is set.
OTZ 7,6	Turn Motor On. This instruction sets the Motor On flip-flop which turns the TTY motor on. This insutrction can be used only with Teletype units that have been modified for remote motor on/off control. Turning the motor on introduces a 600 millisecond delay for all controller sense responses to allow the motor to come up to speed. Clear Request-to-Send. When used with a CRT/Modem, this instruction turns off the Request-to-Send signal. (This feature is available only with the EIA RS232C/CCITT interface option.)

Instruction		Function
OTZ	7,7	Turn Motor Off. This instruction resets the Motor On flip-flop in the controller, which turns the Teletype motor off. Request-to-Send. This instruction generates a Request-to-Send signal for use by a CRT/Modem. (This feature is available only with the EIA RS232C/CCITT interface option.)
OTA OTX	7,0 7,0	Output A or X Register to Controller. These instructions unconditionally transfer the contents of the specified register to the controller interface, which causes the character to be output to the terminal device.
INA INX	7,0 7,0	Input Word from Controller to A or X Register. These instructions unconditionally transfer the character in the controller 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 AIB	7,0 7,0	Input Word/Byte from Controller to Memory Automatically. These instructions unconditionally transfer the character in the controller buffer register to memory, automatically. The AIN instruction causes the character to be loaded into the LSB byte of the memory location and forces the MSB byte to zero. The AIB instruction causes the data to be packed two bytes per word of memory.
BIN	7,1	Input Block from Controller to Memory. This instruction senses the state of the Buffer Ready flip-flop and transfers the character in the controller buffer 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 IBX	7,0 7,0	Input Byte from Controller to A or X Register. These instructions unconditionally transfer the character in the controller buffer register to the specified register. They do not affect the MSB byte of the specified register.
RBA RBX	7,1 7,1	Read Byte from Controller to A or X Register. These instructions sense the state of the Buffer Ready flip-flop and transfer the character in the controller buffer register to the specified register when it is set (true). They do not affect the MSB byte of the specified register.
WRA WRX	7,1 7,1	Write from A or X Register to Controller. These instructions sense the state of the Buffer Ready flip-flop and transfer the character in the specified register to the controller buffer register when it is set (true).

Instruction		Function
AOT	7,0	Output Word/Byte from Memory to Controller, Automatically.
AOB	7,0	These instructions unconditionally transfer a word or byte from memory to the controller buffer register, automatically. In the case of the AOT instruction, the controller 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 Controller. This instruction senses the state of the Buffer Ready flip-flop and transfers the full 16-bit memory word to the controller buffer 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 controller 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 device to be automatically "echoed" back for printing or display, 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 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
LAP     1       Set A Register to Plus One
SEA     7,4     Initialize TTY Interface to Full-Duplex
SEL     7,1     Select Keyboard Mode
RBA     7,0     Input Character When Ready
WRA     7,1     Output Character Just Input
        .
        .
SWM             Restore Word Addressing Mode
        .       Process Character
```

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

The previous 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             LAP     1         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
                        LAP     1
                        SEA     7,7       Disable TTY Interrupts
                        .
                        .
              BUF       RES               5
                        .
```

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 controller, the following standard and offset interrupt locations are provided:

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

The jumper option for relocation to locations : 0022 and : 0026 (or : 0122 and : 0126) in the half-duplex has no effect 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 Controller Instructions

Instruction	Function
SEL 7,0	Enable Auto Echo. This instruction causes all input to be echoed back to the source terminal for printing or display.
SEL 7,1	Select Keyboard. This instruction resets the input buffer full flip-flop and puts the controller in the read mode.
SEL 7,2	Step Read. This instruction causes the character under the read station on the TTY 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 START position. The input Buffer Full flip-flop is set.
SEL 7,3	Select Continuous Read. This instruction causes the TTY paper tape reader to read continuously until the reader is stopped or the tape runs out. The reader switch must be in the START position. The Input Buffer Full flip-flop is reset. Execution of any other Select instruction resets Continuous Read.
SEL 7,4	Initialize Controller to Half-Duplex. The instruction resets all controls and places the controller in the half-duplex mode. The Buffer Ready flip-flop is set.
SEA 7,4 SEX 7,4 (A or X = 1)	Initialize Controller to Full-Duplex. These instructions reset all controls and place the controller 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 Controller to Full-Duplex Diagnostic. These instructions reset all controls and place the controller 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 controller Input Buffer.
SEL 7,5	Enable Output Word Transfer Interrupt. This instruction sets an enable flip-flop in the controller to allow generation of interrupts upon establishment of an Output Buffer Empty condition.

Instruction	Function
SEA 7,5 SEX 7,5 (A or X = 1)	Enable Input Word Transfer Interrupt. These instructions set an enable flip-flop in the controller to allow generation of interrupts upon establishment of an Input Buffer Full condition.
SEL 7,6	Enable Output Echo/EOB Interrupt. This instruction sets an enable flip-flop in the controller to allow generation of the Output EOB interrrupt 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 controller 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 controller by resetting the corresponding enable flip-flops.
SEA 7,7 SEX 7,7 (A or X = 1)	Disable Input Word Transfer and Echo/EOB Interrupts. These instructions disable the two Input interrupts in the controller by resetting the corresponding enable flip-flops.
SEN 0,4	Sense Teletype Controller Installed. This instruction senses if the Teletype controller is installed on the Option Board. If the option is installed (true response) a one-word skip occurs.
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 Controller Not Busy. This instruction senses the state of the controller and generates a one-word skip if it is not processing a character.

Instruction	Function
SEN 7,4	Sense Clear to Send. This instruction senses the clear-to-send line from a CRT or Modem and generates a one-word skip if true. (This feature is available only with the EIA RS232C/CCITT interface option.)
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 Parity Error. This instruction senses the occurrance of a parity error during the most recent input operation and generates a one-word skip if an error occurred. (This instruction requires prior strapping of the parity option).
SEN 7,7	Sense Full Duplex Mode Enabled. This instruction senses the state of the Full Duplex Enable flip-flop and generates a one-word skip if it is set.
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 introduces a 600 millisecond delay for all controller sense responses and interrupts to enable the motor to come up to speed. Clear Request-to-Send. When used with a CRT/Modem, this instruction turns off the Request-to-Send signal. (This feature is available only with the EIA RS232C/CCITT interface option.)
OTZ 7,7	Turn Motor Off. This instruction resets the Motor On flip-flop in the controller which turns the Teletype motor off. Request-to-Send. This instruction generates a Request-to-Send signal for use by a CRT/Modem. (This feature is available only with the EIA RS232C/CCITT interface option.)
OTA 7,1 OTX 7,1	Output A or X Register to Controller. These instructions unconditionally transfer the contents of the specified register to the Ouput Data Buffer, which causes the character to be output to the terminal device.
INA 7,0 INX 7,0	Input Word from Controller to A or X Register. These instructions unconditionally transfer the character in the controller 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 AIB	7,0 7,0	Input Word/Byte from Controller to Memory Automatically. These instructions unconditionally transfer the character in the controller buffer to memory, automatically. The AIN instruction causes the character to be loaded into the LSB byte of the memory location and forces the MSB byte to zero. The AIB instruction causes the data to be packed two bytes per word of memory.
BIN	7,0	Input Block from Controller to Memory. This instruction senses the state of the Input Buffer Full flip-flop and transfers the character in the controller buffer 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 IBX	7,0 7,0	Input Byte from Controller 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 RBX	7,0 7,0	Read Byte from Controller 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 WRX	7,1 7,1	Write from A or X Register to Controller. 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).
AOT AOB	7,0 7,0	Output Word/Byte from Memory to Controller, Automatically. These instructions unconditionally transfer a word or byte from memory to the controller buffer register, automatically. In the case of the AOT instruction, the controller 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 Controller. This instruction senses the state of the Output Buffer Empty flip-flop and transfers the full 16-bit memory word to the controller buffer 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
                .
                .
                .
                .
                .
                .
                .
        EIN             Enable interrupts.
        RTN     SYNC    Return to mainline program.
                .
                .
COUNT   DATA    0
```

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.

Instruction		Function
SEL	8,7	Disarm Sync. Prevents Sync interrupts from being generated without disabling Time interrupts.
SEN	0,2	Sense RTC Installed. A one-word skip occurs if the option is installed.

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 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	Disc
Load Absolute	: 0	: 1	: 2	: 3	: 4
Load Relocatable	: 8	: 9	: A	: B	: C

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 installed.
```

A true response causes a skip to occur (when Autoload is 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 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 failed.

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 un-
                                                conditional Jump to the Power
                                                Up subroutine.

Interrupt Location : 001C      JST   DOWN     This is the Power Down in-
        (or : 011C if offset)                   terrupt location. It contains
                                                a Jump and Store to the Power
                                                Down subroutine. Using a JST
                                                automatically saves the contents
                                                of the P register and disables
                                                interrupts.

              DOWN             ENT            Reserved location for storage of
                                                the P register when the JST
                                                instruction at the Power Fail
                                                interrupt locations is executed.

                               PFD            Disable further Power Fail
                                                interrupts.

                               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.

                               ICA   CSAVE    Save the contents of the Console
                               STA            Data Register.

                               STX   XSAVE    Save the X Register.

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

```
                WAIT                    Wait for Power Down to
                .                       complete.
                .
                .
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, and reset it.

                JAN   $+2

                HLT                     No - do not restart.

                LDX   XSAVE             This instruction restores the
                                        X Register.

                LDA   CSAVE             Restore the contents of the
                OCA                     Console Data 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.

                PFE                     Enable Power Fail

                EIN                     Enable interrupts.

                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.

CSAVE           DATA  0                 Console Register save location.

XSAVE           DATA  0                 X Register save location.

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

PSTP	DATA0	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 outside EIN/DIN control in this example. If the Power Fail interrupt were inside EIN/DIN control, the UP routine would not have to include a PFE instruction and the DOWN routine would not have to include a PFD 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.

HEXADECIMAL	DECIMAL EQUIVALENTS							
	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. 8-BIT ASCII Teletype Codes

Symbol	Hexadecimal Code
@	C0
A	C1
B	C2
C	C3
D	C4
E	C5
F	C6
G	C7
H	C8
I	C9
J	CA
K	CB
L	CC
M	CD
N	CE
O	CF
P	D0
Q	D1
R	D2
S	D3
T	D4
U	D5
V	D6
W	D7
X	D8
Y	D9
Z	DA
[	DB
\	DC
]	DD
↑	DE
←	DF
NULL	00
BELL	87

Symbol	Hexadecimal Code
↓	A0
!	A1
"	A2
#	A3
$	A4
%	A5
&	A6
'	A7
(	A8
)	A9
*	AA
+	AB
,	AC
-	AD
.	AE
/	AF
0	B0
1	B1
2	B2
3	B3
4	B4
5	B5
6	B6
7	B7
8	B8
9	B9
:	BA
;	BB
<	BC
=	BD
>	BE
?	BF
CR	8D
LF	8A
RUBOUT	FF

Appendix B

RECOMMENDED DEVICE AND INTERRUPT ADDRESSES

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
Refer to Table B-3	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	
Synchronous Modem Controller (SCM)	0D	
Asynchronous Modem Multiplexer (AMM)	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 Address 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)		:C0 AMM RCV1		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)		:C4 AMM XMIT1		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)		:C8 AMM RCV2		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)		:CC AMM XMIT2		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)		:D0 AMM RCV3		0
1									1
2	:12 MEMORY PARITY	:32** 103 DSC INPUT WORD	:52 CASSETTE WORD	:72 16-BIT I/O WORD	:92 PIM(9)	:B2 AMM EXCEPTION			2
3									3
4					:94 PIM(10)	:B4 AMM RECEIVE EOB	:D4 AMM XMIT3		4
5									5
6		:36** 103 DSC IN EOB	:56 CASSETTE EOP	:76 16-BIT I/O EOB	:96 PIM(11)	:B6 AMM XMIT EOB			6
7									7
8	:18 RTC CLOCK	:38** 103 DSC PAR ERR/FLT			:98 PIM(12)		:D8 AMM RCV4		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)		:DC AMM XMIT4		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

Table B-3. Device Address 0 Command Summary

FUNCTION CODE	SELECT COMMANDS	SENSE COMMANDS	INPUT COMMANDS	OUTPUT COMMANDS
0	Terminate Autoload Sequence	Autoload Option Installed	SIA (: 5800), SIX (: 5A00)	SOA (: 6C00), SOX (: 6E00)
1	Initiate Autoload Sequence	Real-Time Clock Option Installed	ISA (: 5801) ISX (: 5A01)	SIN 0 (: 6801)
2	PFE (: 4002)			SIN 1 (: 6802)
3	PFD (: 4003)			SIN 2 (: 6803)
4	OCA (: 4404) OCX (: 4604)	TTY/CRT/Modem Option Installed	ICA (: 5804) ICX (: 5A04)	SIN 3 (: 6804)
5	CIE (: 4005)			SIN 4 (: 6805)
6	CID (: 4006)			SIN 5 (; 6806)
7	TRP (: 4007)			SIN 6 (: 6807)

Appendix C

INSTRUCTION SET BY CLASS

This appendix contains the ALPHA LSI instruction set in class 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]

No Operator = Direct Addressing
* = Indirect Addressing (multi-level)
EXPRESSION 1 represents an address to be stored in the second word of the instruction. EXPRESSION 2 is an optional absolute instruction count in the range 1 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	[EXPRESSION]	[COMMENTS]
		EXPRESSION: there are no expressions in the operand field, except for BAO and BXO instructions, where it must be value in the range 1 through 16.	

Figure C.6. Class 6 - Register Change 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 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 Skeleton 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	Enchange 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
SCMB	Scan Memory Byte	CD00	3-14
CMS	Compare A with Memory, Skip (low, high, equal test)	D000	3-12
CMSB	Compare A with Memory Byte, skip (low, high, equal test)	D000	3-13

DOUBLE WORD MEMORY REFERENCE (Class 2)

		Instruction Skeleton in Hex	Reference Page
DVD	Divide	1970	3-15
MPY	Multiply and Add	1960	3-16
NRM	Normalize A and X	1940	3-17

BYTE IMMEDIATE (Class 3)

AAI	Add to A Register Immediate	0B00	3-18
AXI	Add to X Register Immediate	C200	3-18
SAI	Subtract from A Register Immediate	0D00	3-18
SXI	Subtract from X Register Immediate	C300	3-18
CAI	Campare to A Immediate, skip if not equal	C000	3-18
CXI	Compare to X Immediate, skip if not equal	C100	3-18
LAP	Load A Positive Immediate	C600	3-19
LXP	Load X Positive Immediate	C400	3-19
LAM	Load A Minus Immediate	C700	3-19
LXM	Load X Minus Immediate	C500	3-19

CONDITIONAL JUMP (Class 4 or 8)

Microcoded (Class 8)

JOC	Jump on Condition Specified	2000	3-20

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 Zero	2180	3-20
JAM	Jump if A Minus	2080	3-21
JXZ	Jump if X Zero	2800	3-21
JXN	Jump if X Not Zero	3800	3-21

Control (Class 4)

JSS	Jump if SENSE Indicator on	3400	3-21
JSR	Jump if SENSE Indicator off	2400	3-21
JOS	Jump if OV Set	2200	3-21
JOR	Jump if OV Reset	3200	3-21

SHIFT CLASS (Class 5)

		Instruction Skeleton in Hex	Reference Page
Single Register			
Arithmetic			
ARA	Arithmetic Right A	10D0	3-22
ARX	Arithmetic Right X	10A8	3-22
ALA	Arithmetic Left A	1050	3-22
ALX	Arithmetic Left X	1028	3-22
Logical			
LRA	Logical Right A	13D0	3-23
LRX	Logical Right X	13A8	3-23
LLA	Logical Left A	1350	3-23
LLX	Logical Left X	1328	3-23
Rotate			
RRA	Rotate Right A with OV	11D0	3-24
RRX	Rotate Right X with OV	11A8	3-24
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	3-25
LRR	Long Rotate Right with OV	1980	3-25

REGISTER CHANGE (Class 6)

Accumulator			
ZAR	Zero A Register	0110	3-26
ARP	Set A Register to Positive 1	0350	3-26
ARM	Set A Register to Minus 1	0010	3-26
CAR	Complement (1's) A Register	0210	3-26
NAR	Negate A Register	0310	3-26
IAR	Increment A Register	0150	3-26
DAR	Decrement A Register	00D0	3-26

		Instruction Skeleton 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	3-27
SAO	Sign of A to OV	1340	3-27
SXO	Sign of X to OV	1320	3-27
LAO	Least significant bit of A to OV	13C0	3-27
LXO	Least significant bit of X to OV	13A0	3-27
BAO	Bit of A to OV	1340	3-27
BXO	Bit of X to OV	1320	3-27
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 Z	0030	3-27
EAX	Exchange A and X	0428	3-27
ANA	AND of A and X to A	0070	3-27
ANX	AND of A and X to X	0068	3-27
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	3-28
CXA	1's Complement (X) and put in A	0410	3-28
NAX	Negate (A) and put in X	0308	3-28
NXA	Negate (X) and put in A	0510	3-28
IAX	Increment (A) and put in X	0148	3-28
IXA	Increment (X) and put in A	0130	3-28
IPX	Increment (P) and put in X	0090	3-28
DAX	Decrement (A) and put in X	00C8	3-28
DXA	Decrement (X) and put in A	00B0	3-28
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	3-29
OCX	Output X to Console Data Register	4604	3-29

CONTROL (Class 6)

		Instruction Skeleton 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	3-30
SWM	Set Word Operand Mode	0F00	3-30
Status			
SIN	Status Inhibit	6800	3-30
SIA	Status Input to A	5800	3-30
SIX	Status Input to X	5A00	3-31
SOA	Status Output from A	6C00	3-31
SOX	Status Output from X	6E00	3-31
Interrupts			
EIN	Enable Interrupts	0A00	3-31
DIN	Disable Interrupts	0C00	3-31
CIE	Console Interrupt Enable	4005	3-31
CID	Console Interrupt Disable	4006	3-31
PFE	Power Fail Interrupt Enable	4002	3-31
PFD	Power Fail Interrupt Disable	4003	3-32
TRP	Trap	4007	3-32

INPUT/OUTPUT (Class 7)

Control			
SEL	Select	4000	3-33
SEA	Select and Present A	4400	3-33
SEX	Select and Present X	4600	3-33
SEN	Sense and Skip on Response	4900	3-33
SSN	Sense and Skip on no Response	4800	3-33

		Instruction Skeleton in Hex	Reference Page
Unconditional Word			
INA	Input Word to A	5800	3-34
INAM	Input Word to A Masked	5C00	3-34
INX	Input Word to X	5A00	3-34
INXM	Input Word to X Masked	5E00	3-34
OTA	Output A	6C00	3-34
OTX	Output X	6E00	3-34
OTZ	Output Zero's	6800	3-34
Conditional Word			
RDA	Read Word to A	5900	3-35
RDAM	Read Word to A Masked	5D00	3-35
RDX	Read Word to X	5B00	3-35
RDXM	Read Word to X Masked	5F00	3-35
WRA	Write A	6D00	3-35
WRX	Write X	6F00	3-35
WRZ	Write Zero's	6900	3-35
Unconditional Byte			
IBA	Input Byte to A	7800	3-36
IBAM	Input Byte to A Masked	7C00	3-36
IBX	Input Byte to X	7A00	3-36
IBXM	Input Byte to X Masked	7E00	3-36
Conditional Byte			
RBA	Read Byte to A	7900	3-36
RBAM	Read Byte to A Masked	7D00	3-37
RBX	Read Byte to X	7B00	3-37
RBXM	Read Byte to X Masked	7F00	3-37
Block			
BIN	Input Block to Memory	7100	3-38
BOT	Output Block from Memory	7500	3-38
Automatic			
AIN	Automatic Input to Memory--Word	5000	3-41
AOT	Automatic Output to Memory--Word	6000	3-41
AIB	Automatic Input to Memory--Byte	5400	3-41
AOB	Automatic Output from Memory--Byte	6400	3-41

Appendix D

INSTRUCTION SET IN ALPHABETICAL ORDER

This appendix contains the ALPHA LSI instruction set in alphabetical order by instruction mnemonic. Instructions with variable fields have been appended with astericks (*).

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
AAI	0B00*	Add to A direct immediate	3-18
ADD	8800*	Add to A direct, scratchpad	3-10
ADD	8900*	Add to A indirect, AP in scratchpad	3-10
ADD	8A00*	Add to A relative to P forward, direct	3-10
ADD	8B00*	Add to A relative to P forward, indirect	3-10
ADD	8C00*	Add to A indexed, direct	3-10
ADD	8D00*	Add to A indexed, indirect	3-10
ADD	8E00*	Add to A relative to P backward, direct	3-10
ADD	8F00*	Add to A relative to P backward, indirect	3-10
ADDB	8800*	Add Byte, direct, scratchpad	3-10
ADDB	8900*	Add Byte, indirect, AP in scratchpad	3-10
ADDB	8A00*	Add Byte 0, relative to P forward, direct	3-10
ADDB	8B00*	Add Byte, indirect, AP relative to P, forward	3-10
ADDB	8C00*	Add Byte, direct, indexed	3-10
ADDB	8D00*	Add Byte, indirect, indexed, AP in scratchpad	3-10
ADDB	8E00*	Add Byte 1, relative to P forward, direct	3-10
ADDB	8F00*	Add Byte, indirect, relative to P, backward	3-10

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
AIB	5400*	Automatic Input: Byte	3-41
AIN	5000*	Automatic Input: Word	3-41
ALA	1050*	Arithmetic shift A left	3-22
ALX	1028*	Arithmetic shift X left	3-22
ANA	0070	AND of A and X to A	3-27
AND	8000*	AND to A direct, scratchpad	3-11
AND	8100*	AND to A indirect, AP in scratchpad	3-11
AND	8200*	AND to A relative to P forward, direct	3-11
AND	8300*	AND to A relative to P forward, indirect	3-11
AND	8400*	AND to A indexed, direct	3-11
AND	8500*	AND to A indexed, indirect	3-11
AND	8600*	AND to A relative to P backward, direct	3-11
AND	8700*	AND to A relative to P backward, indirect	3-11
ANDB	8000*	AND to A Byte, direct, sratchpad	3-11
ANDB	8100*	AND to A Byte, indirect, AP in scratchpad	3-11
ANDB	8200*	AND to A Byte 0, direct, relative to P forward	3-11
ANDB	8300*	AND to A Byte, indirect, AP relative to P forward	3-11
ANDB	8400*	AND to A Byte, indexed, direct	3-11
ANDB	8500*	AND to A Byte, indexed, indirect, AP in scratchpad	3-11
ANDB	8600*	AND to A Byte 1, direct, relative to P forward	3-11
ANDB	8700*	AND to A Byte, indirect, AP relative to P backward	3-11

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
ANX	0068	AND of A and X to X	3-27
AOB	6400*	Automatic Output: Byte	3-41
AOT	6000*	Automatic Output: Word	3-41
ARA	10D0*	Arithmetic shift A right	3-22
ARM	0010	Set A to minus 1	3-26
ARP	0350	Set A to plus 1	3-26
ARX	10A8*	Arithmetic shift X right	3-22
AXI	C200*	Add to X immediate	3-18
AXM	0018	Set A and X to minus 1	3-27
AXP	0358	Set A and X to plus 1	3-28
BAO	1340*	Bit of A to overflow	3-27
BIN	7100*	Block input to memory	3-38
BOT	7500*	Block output from memory	3-38
BXO	1320*	Bit of X to overflow	3-27
CAI	C000*	Compare to A immediate, skip if unequal	3-18
CAR	0210	Complement A	3-26
CAX	0208	Complement A and put in X	3-28
CID	4006	Console interrupt disable	3-31
CIE	4005	Console interrupt enable	3-31
CMS	D000*	Compare memory to A and skip if high or equal; direct, scratchpad	3-12
CMS	D100*	Compare memory to A and skip if high or equal; indirect, AP in scratchpad	3-12

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
CMS	D200*	Compare memory to A and skip if high or equal; relative to P forward, direct	3-12
CMS	D300*	Compare memory to A and skip if high or equal; relative to P forward, indirect	3-12
CMS	D400*	Compare memory to A and skip if high or equal; indexed, direct	3-12
CMS	D500*	Compare memory to A and skip if high or equal; indexed, indirect	3-12
CMS	D600*	Compare memory to A and skip if high or equal; relative to P backward, direct	3-12
CMS	D700*	Compare memory to A and skip if high or equal; relative to P backward, indirect	3-12
CMSB	D000*	Compare Byte and skip if high or equal, direct, scratchpad	3-13
CMSB	D100*	Compare Byte and skip if high or equal, indirect, AP in scratchpad	3-13
CMSB	D200*	Compare Byte 0 and skip if high or equal, direct, relative to P forward	3-13
CMSB	D300*	Compare Byte and skip if high or equal, indirect, AP relative to P forward	3-13
CMSB	D400*	Compare Byte and skip if high or equal, indexed, direct	3-13
CMSB	D500*	Compare Byte and skip if high or equal, indexed, indirect, AP in scratchpad	3-13
CMSB	D600*	Compare Byte 1 and skip if high or equal, direct, relative to P forward	3-12
CMSB	D700*	Compare Byte and skip if high or equal, indirect, AP relative to P backward	3-13

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
COV	1600	Complement overflow	3-27
CXA	0410	Complement X and put in A	3-28
CXI	C100*	Compare to X immediate, skip if unequal	3-18
CXR	0408	Complement X	3-26
DAR	00D0	Decrement A	3-26
DAX	00C8	Decrement A and put in X	3-28
DIN	0C00	Disable interrupts	3-31
DVD	1970*	Divide	3-15
DXA	00B0	Decrement X and put in A	3-28
DXR	00A8	Decrement X	3-26
EAX	0428	Exchange A and X	3-27
EIN	0A00	Enable interrupts	3-31
EMA	B800*	Exchange memory and A; direct, scratchpad	3-12
EMA	B900*	Exchange memory and A; indirect, AP in scratchpad	3-12
EMA	BA00*	Exchange memory and A; relative to P forward, direct	3-12
EMA	BB00*	Exchange memory and A; relative to P forward, indirect	3-12
EMA	BC00*	Exchange memory and A; indexed, direct	3-12
EMA	BD00*	Exchange memory and A; indexed, indirect	3-12
EMA	BE00*	Exchange memory and A; relative to P backward, direct	3-12
EMA	BF00*	Exchange memory and A; relative to P backward, indirect	3-12

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
EMAB	B800*	Exchange Memory and A Byte, direct scratchpad	3-12
EMAB	B900*	Exchange Memory and A Byte, indirect, AP in scratchpad	3-12
EMAB	BA00*	Exchange Memory and A Byte 0, direct, relative to P forward	3-12
EMAB	BB00*	Exchange Memory and A Byte, indirect, AP relative to P forward	3-12
EMAB	BC00*	Exchange Memory and A Byte, indexed, direct	3-12
EMAB	BD00*	Exchange Memory and A Byte, indexed, indirect, AP in scratchpad	3-12
EMAB	BE00*	Exchange Memory and A Byte 1, direct, relative to P forward	3-12
EMAB	BF00*	Exchange Memory and A, indirect, AP relative to P backward	3-12
HLT	0800	Halt	3-29
IAR	0150	Increment A	3-26
IAX	0148	Increment A and put in X	3-28
IBA	7800*	Input byte to A (unconditionally)	3-36
IBAM	7C00*	Input byte to A, masked (unconditionally)	3-36
IBX	7A00*	Input byte to X (unconditionally)	3-36
IBXM	7E00*	Input byte to X, masked (unconditionally)	3-36
ICA	5804	Input Console Data Register to A	3-28
ICX	5A04	Input Console Data Register to X	3-28
IMS	D800*	Increment memory and skip on zero result; direct, scratchpad	3-13
IMS	D900*	Increment memory and skip on zero result; indirect, AP in scratchpad	3-13

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
IMS	DA00*	Increment memory and skip on zero result; relative to P forward, direct	3-13
IMS	DB00*	Increment memory and skip on zero result; relative to P forward, indirect	3-13
IMS	DC00*	Increment memory and skip on zero result; indexed, direct	3-13
IMS	DD00*	Increment memory and skip on zero result; indexed, indirect	3-13
IMS	DE00*	Increment memory and skip on zero result; relative to P backward, direct	3-13
IMS	DF00*	Increment memory and skip on zero result; relative to P backward, indirect	3-13
INA	5800*	Input word to A (unconditionally)	3-34
INAM	5C00*	Input word to A, masked (uncondtionally)	3-34
INX	5A00*	Input word to X (unconditionally)	3-34
INXM	5E00*	Input word to X, masked (uncondtionally)	3-34
IOR	A000*	Inclusive OR to A; direct, scratchpad	3-11
IOR	A100*	Inclusive OR to A; indirect, AP in scratchpad	3-11
IOR	A200*	Inclusive OR to A; relative to P forward, direct	3-11
IOR	A300*	Inclusive OR to A; relative to P forward, indirect	3-11
IOR	A400*	Inclusive OR to A; indexed, direct	3-11
IOR	A500*	Inclusive OR to A; indexed, indirect	3-11
IOR	A600*	Inclusive OR to A; relative to P backward, direct	3-11
IOR	A700*	Inclusive OR to A; relative to P backward, indirect	3-11

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
IORB	A000*	Inclusive OR Byte, direct, scratchpad	3-11
IORB	A100*	Inclusive OR Byte, indirect, AP in scratchpad	3-11
IORB	A200*	Inclusive OR Byte 0, direct, relative to P forward	3-11
IORB	A300*	Inclusive OR Byte, indirect, AP relative to P forward	3-11
IORB	A400*	Inclusive OR Byte, indexed, direct	3-11
IORB	A500*	Inclusive OR Byte, indexed, indirect, AP in scratchpad	3-11
IORB	A600*	Inclusive OR Byte 0, direct, relative to P forward	3-11
IORB	A700*	Inclusive OR Byte, indirect, AP relative to P backward	3-11
IPX	0090	Increment P and put in X	3-28
ISA	5801	Input data switches to A	3-28
ISX	5A01	Input data switches to X	3-28
IXA	0130	Increment X and put in A	3-28
IXR	0128	Increment X	3-26
JAG		Jump if A positive and not equal to zero: (A)>0	3-20
	3180*	Forward jump	
	31C0*	Backward jump	
JAL		Jump if A negative or equal to zero: (A)≤ 0	3-20
	2180*	Forward jump	
	21C0*	Backward jump	
JAM		Jump if A negative: (A)< 0	3-21
	2080*	Forward jump	
	20C0*	Backward jump	

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
JAN		Jump if A not zero: (A)≠0	3-20
	3100*	Forward jump	
	3140*	Backward jump	
JAP		Jump if A positive or equal to zero: (A)≥ 0	3-20
	3080*	Forward jump	
	30C0*	Backward jump	
JAZ		Jump if A zero: (A)=0	3-20
	2100*	Forward jump	
	2140*	Backward jump	
JMP	F000*	Jump unconditionally; direct, scratchpad	3-13
JMP	F100*	Jump unconditionally; indirect, AP in scratchpad	3-13
JMP	F200*	Jump unconditionally; relative to P forward, direct	3-13
JMP	F300*	Jump unconditionally; relative to P forward, indirect	3-13
JMP	F400*	Jump unconditionally; indexed, direct	3-13
JMP	F500*	Jump unconditionally; indexed, indirect	3-13
JMP	F600*	Jump unconditionally; relative to P backward, direct	3-13
JMP	F700*	Jump unconditionally; relative to P backward, indirect	3-13
JOC	2000	Jump on condition specified	3-20
JOR		Jump if overflow reset: OV=0	3-21
	3200*	Forward jump	
	3240*	Backward jump	
JOS		Jump if overflow set: OV=1	3-21
	2200*	Forward jump	
	2240*	Backward jump	

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
JSR		Jump if Sense Switch off: SS=0	3-21
	2400*	Forward jump	
	2440*	Backward jump	
JSS		Jump if Sense Switch on: SS=1	3-21
	3400*	Forward jump	
	3440*	Backward jump	
JST	F800*	Jump and Store; direct, scratchpad	3-13
JST	F900*	Jump and Store; indirect, AP in scratchpad	3-13
JST	FA00*	Jump and Store; relative to P forward, direct	3-13
JST	FB00*	Jump and Store; relative to P forward, indirect	3-13
JST	FC00*	Jump and Store; indexed, direct	3-13
JST	FD00*	Jump and Store; indexed, indirect	3-13
JST	FE00*	Jump and Store; relative to P backward, direct	3-13
JST	FF00*	Jump and Store; relative to P backward, indirect	3-13
JXN		Jump if X non-zero: (X)≠0	3-21
	3800*	Forward jump	
	3840*	Backward jump	
JXZ		Jump if X equal to zero: (X)=0	3-21
	2800*	Forward jump	
	2840*	Backward jump	
LAM	C700*	Load A minus immediate	3-19
LAO	13C0	LSB of A to OV	3-27
LAP	C600*	Load A positive immediate	3-19
LDA	B000*	Load A; direct, scratchpad	3-12
LDA	B100*	Load A; indirect, AP in scratchpad	3-12

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
LDA	B200*	Load A; relative to P forward, direct	3-12
LDA	B300*	Load A; relative to P forward, indirect	3-12
LDA	B400*	Load A; indexed, direct	3-12
LDA	B500*	Load A; indexed, indirect	3-12
LDA	B600*	Load A; relative to P backward, direct	3-12
LDA	B700*	Load A; relative to P backward, indirect	3-12
LDAB	B000*	Load A Byte, direct, scratchpad	3-12
LDAB	B100*	Load A Byte, indirect, AP in scratchpad	3-12
LDAB	B200*	Load A Byte 0, direct, relative to P forward	3-12
LDAB	B300*	Load A Byte, indirect, AP relative to P forward	3-12
LDAB	B400*	Load A Byte, indexed, direct	3-12
LDAB	B500*	Load A Byte, indexed, indirect, AP in scratchpad	3-12
LDAB	B600*	Load A Byte 1, direct, relative to P forward	3-12
LDAB	B700*	Load A Byte, indirect, AP relative to P backward	3-12
LDX	E000*	Load X; direct, scratchpad	3-12
LDX	E100*	Load X; indirect, AP in scratchpad	3-12
LDX	E200*	Load X; relative to P forward, direct	3-12
LDX	E300*	Load X; relative to P forward, indirect	3-12
LDX	E400*	Load X; indexed, direct	3-12
LDX	E500*	Load X; indexed, indirect	3-12
LDX	E600*	Load X; relative to P backward, direct	3-12

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
LDX	E700*	Load X; relative to P backward, indirect	3-12
LDXB	E000*	Load X Byte, direct, scratchpad	3-12
LDXB	E100*	Load X Byte, indirect, AP in scratchpad	3-12
LDXB	E200*	Load X Byte 0, direct, relative to P forward	3-12
LDXB	E300*	Load X Byte, indirect, relative to P forward	3-12
LDXB	E400*	Load X Byte, indexed, direct	3-12
LDXB	E500*	Load X Byte, indexed, indirect, AP in scratchpad	3-12
LDXB	E600*	Load X Byte 1, direct, relative to P forward	3-12
LDXB	E700*	Load X Byte, indirect, relative to P backward	3-12
LLA	1350*	Logical shift A left	3-23
LLL	1B00*	Long logical left shift	3-24
LLR	1B80*	Long logical right shift	3-24
LLX	1328*	Logical shift X left	3-23
LRA	13D0*	Logical shift A right	3-23
LRL	1900*	Long rotate left	3-25
LRR	1980*	Long rotate right	3-25
LRX	13A8*	Logical shift X right	3-23
LXM	C500*	Load X minus immediate	3-19
LXO	13A0	LSB of X to OV	3-27
LXP	C400*	Load X positive immediate	3-19
MPY	1960*	Multiply and Add	3-16

INSTUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
NAR	0310	Negate A register	3-26
NAX	0308	Negate A and put in X	3-28
NOP	0000	No operation	3-29
NRM	1940*	Normalize A and X	3-17
NRA	0610	NOR if (A and X) to A: $\overline{(A) \vee (X)} \rightarrow A$	3-27
NRX	0608	NOR if (A and X) to X: $\overline{(A) \vee (X)} \rightarrow A$	3-27
NXA	0510	Negate X and put in A	3-28
NXR	0508	Negate X register	3-26
OCA	4404	Output A to Console Data Register	3-29
OCX	4604	Output X to Console Data Register	3-29
OTA	6C00*	Output A register (unconditionally)	3-34
OTX	6E00*	Output X register (unconditionally)	3-34
OTZ	6800*	Output zero (unconditionally)	3-34
PFD	4003	Power Fail interrupt disable	3-32
PFE	4002	Power Fail interrupt enable	3-31
RBA	7900*	Read byte to A	3-37
RBAM	7D00*	Read byte to A, masked	3-37
RBX	7B00*	Read byte to X	3-37
RBXM	7F00*	Read byte to X, masked	3-37
RDA	5900*	Read word to A	3-35
RDAM	5D00*	Read word to A, masked	3-35

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
RDX	5B00*	Read word to X	3-35
RDXM	5F00*	Read word to X, masked	3-35
RLA	1150*	Rotate A left with OV	3-23
RLX	1128*	Rotate X left with OV	3-23
ROV	1200	Reset overflow	3-26
RRA	11D0*	Rotate A right with OV	3-24
RRX	11A8*	Rotate X right with OV	3-24
SAI	00D0*	Subtract from A immediate	3-18
SAO	1340	Sign of A to OV	3-27
SBM	0E00	Set byte mode	3-30
SCM	CD00*	Scan memory, indexed, indirect	3-13
SCMB	CD00*	Scan memory Byte, indexed, indirect	3-14
SEA	4400*	Select and present A	3-33
SEL	4000*	Select function	3-33
SEN	4900*	Sense and skip on response	3-33
SEX	4600*	Select and present X	3-33
SIA	5800	Status input to A	3-30
SIN	6800	Status inhibit	3-30
SIX	5A00	Status input to X	3-30
SOA	6C00	Status output from A	3-30
SOX	6E00	Status output from X	3-30
SOV	1400	Set overflow	3-26

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
SSN	4800*	Sense and skip on no response	3-33
STA	9800*	Store A; direct, scratchpad	3-12
STA	9900*	Store A; indirect, AP in scratchpad	3-12
STA	9A00*	Store A; relative to P forward, direct	3-12
STA	9B00*	Store A; relative to P forward, indirect	3-12
STA	9C00*	Store A; indexed, direct	3-12
STA	9D00*	Store A; indexed, indirect	3-12
STA	9E00*	Store A; relative to P backward, direct	3-12
STA	9F00*	Store A; relative to P backward, indirect	3-12
STAB	9800*	Store A Byte, direct, scratchpad	3-12
STAB	9900*	Store A Byte, indirect, AP in scratchpad	3-12
STAB	9A00*	Store A Byte 0, direct, relative to P forward	3-12
STAB	9B00*	Store A Byte, indirect, AP relative to P forward	3-12
STAB	9C00*	Store A Byte, indexed, direct	3-12
STAB	9D00*	Store A Byte, indexed, indirect, AP in scratchpad	3-12
STAB	9E00*	Store A Byte 1, direct, relative to P forward	3-12
STAB	9F00*	Store A Byte, indirect, AP relative to P backward	3-12
STOP	0800*	Halt with operand	3-29
STX	E800*	Store X; direct, scratchpad	3-12
STX	E900*	Store A; indirect, AP in scratchpad	3-12
STX	EA00*	Store X; relative to P forward, direct	3-12

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
STX	EB00*	Store X; relative to P forward, indirect	3-12
STX	EC00*	Store X; indexed, direct	3-12
STX	ED00*	Store X; indexed, indirect	3-12
STX	EE00*	Store X; relative to P backward, direct	3-12
STX	EF00*	Store X; relative to P backward, indirect	3-12
STXB	E800*	Store X Byte, direct, scratchpad	3-12
STXB	E900*	Store X Byte, indirect, AP in scratchpad	3-12
STXB	EA00*	Store X Byte 0, direct, relative to P forward	3-12
STXB	EB00*	Store X Byte, indirect, relative to P forward	3-12
STXB	EC00*	Store X Byte, indexed, direct	3-12
STXB	ED00*	Store X Byte, indexed, indirect, AP	3-12
STXB	EE00*	Store X Byte 1, direct, relative to P forward	3-12
STXB	EF00*	Store X Byte, indirect, relative to P backward	3-12
SUB	9000*	Subtract from A; direct, scratchpad	3-10
SUB	9100*	Subtract from A; indirect, AP in scratchpad	3-10
SUB	9200*	Subtract from A; relative to P forward, direct	3-10
SUB	9300*	Subtract from A; relative to P forward, indirect	3-10
SUB	9400*	Subtract from A; indexed, direct	3-10
SUB	9500*	Subtract from A; indexed, indirect	3-10
SUB	9600*	Subtract from A; relative to P backward, direct	3-10
SUB	9700*	Subtract from A; relative to P backward, indirect	3-10

INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
SUBB	9000*	Subtract Byte, direct, scratchpad	3-10
SUBB	9100*	Subtract Byte, indirect, AP in scratchpad	3-10
SUBB	9200*	Subtract Byte 0, direct, relative to P forward	3-10
SUBB	9300*	Subtract Byte, indirect, AP relative to P forward	3-10
SUBB	9400*	Subtract Byte, indexed, direct	3-10
SUBB	9500*	Subtract Byte, indirect, indexed, AP in scratchpad	3-10
SUBB	9600*	Subtract Byte 1, direct, relative to P forward	3-10
SUBB	9700*	Subtract Byte, indirect, relative to P backward	3-10
SWM	0F00	Set word Mode	3-30
SXI	C300*	Subtract from X immediate	3-18
SXO	1320	Sign of X to OV	3-27
TAX	0048	Transfer A to X	3-27
TRP	4007	Trap	3-32
TXA	0030	Transfer X to A	3-27
WAIT	F600	Wait for interrupts	3-29
WRA	6D00*	Write from A	3-35
WRX	6F00*	Write from X	3-35
WRZ	6900*	Write zeros	3-35
XOR	A800*	Exclusive OR to A; direct, scratchpad	3-11
XOR	A900*	Exclusive OR to A; indirect, AP in scratchpad	3-11
XOR	AA00*	Exclusive OR to A; relative to P forward, direct	3-11

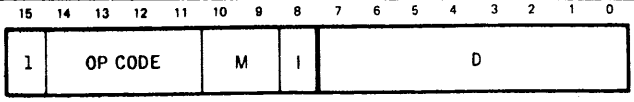
INSTRUCTION SET IN ALPHABETICAL ORDER

Instruction Mnemonic	Instruction Skeleton in Hex	Description	Page
XOR	AB00*	Exclusive OR to A; relative to P forward, indirect	3-11
XOR	AC00*	Exclusive OR to A; indexed, direct	3-11
XOR	AD00*	Exclusive OR to A; indexed, indirect	3-11
XOR	AE00*	Exclusive OR to A; relative to P backward, direct	3-11
XOR	AF00*	Exclusive OR to A; relative to P backward, indirect	3-11
XORB	A800*	Exclusive OR Byte, direct, scratchpad	3-11
XORB	A900*	Exclusive OR Byte, indirect, AP in scratchpad	3-11
XORB	AA00*	Exclusive OR Byte 0, direct, relative to P forward	3-11
XORB	AB00*	Exclusive OR Byte, indirect, AP relative to P forward	3-11
XORB	AC00*	Exclusive OR Byte, indexed, direct	3-11
XORB	AD00*	Exclusive OR Byte, indexed, indirect, AP in scratchpad	3-11
XORB	AE00*	Exclusive OR Byte 1, direct, relative to P forward	3-11
XORB	AF00*	Exclusive OR Byte, indirect, AP relative to P backward	3-11
XRM	0008	Set X to minus 1	3-26
XRP	0528	Set X to plus 1	3-26
ZAR	0110	Zero A register	3-26
ZAX	0118	Zero A and X register	3-27
ZXR	0108	Zero X register	3-26

Appendix E
INSTRUCTION SET IN NUMERICAL ORDER

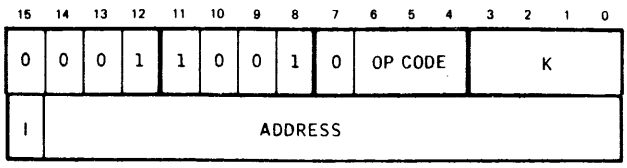
This appendix contains the ALPHA LSI instruction set in machine code in numerical order. For each instruction, reference is made to one of the machine code formats listed below. Instructions with variable fields (D, K, etc.) have been appended with asterisks (*).

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



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 E-1. Single-Word Memory Reference Instruction Machine Code Format



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

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

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

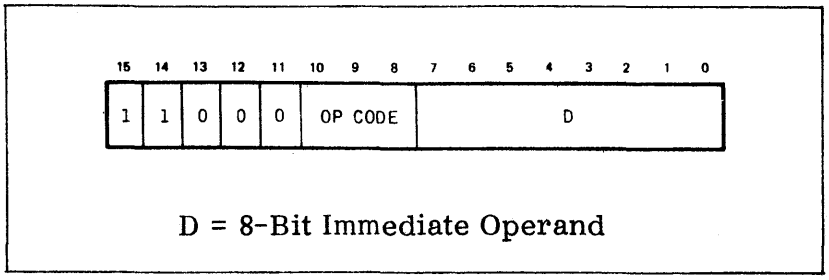
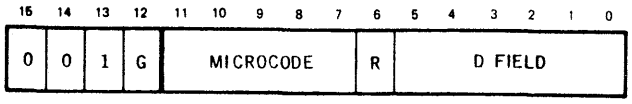


Figure E-3 . Byte Immediate Instruction Machine Code Format

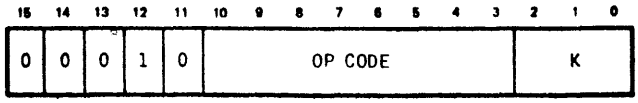


Bits	Field	Definition
12	G	Test Group Indicator: G = 1 for AND Group G = 0 for OR Group
7-11	Conditions	Microcode of Test Conditions:

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

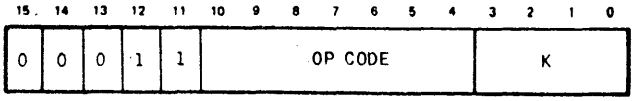
6	R	Jump Direction: R = 0 for Forward Jump R = 1 for Backward Jump
0-5	D Field	Jump Distance (-63 to +64)

Figure E-4 . Conditional Jump Instruction Machine Code Format



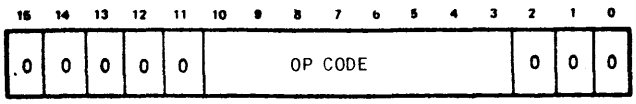
K = Shift Control Count, Shift Will Move 1 + K Bit Positions.
Op Code = Shift Control Code Which Selects Source, Type of Shift, and Location of Results

Figure E-5. Single-Register Shift Instruction Machine Code Format



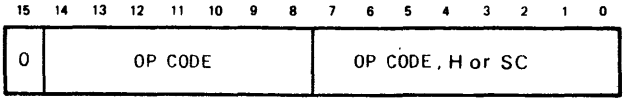
Op Code = Shift Control Code Which Selects the Type of Long Shift to be Executed
K = Shift Count. Shift Will Move 1 + K Bit Positions

Figure E-6. Double-Register Shift Instruction Machine Code Format



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

Figure E-7. Register Change Instruction Machine Code Format



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

Figure E-8. Control Instruction Machine Code Format

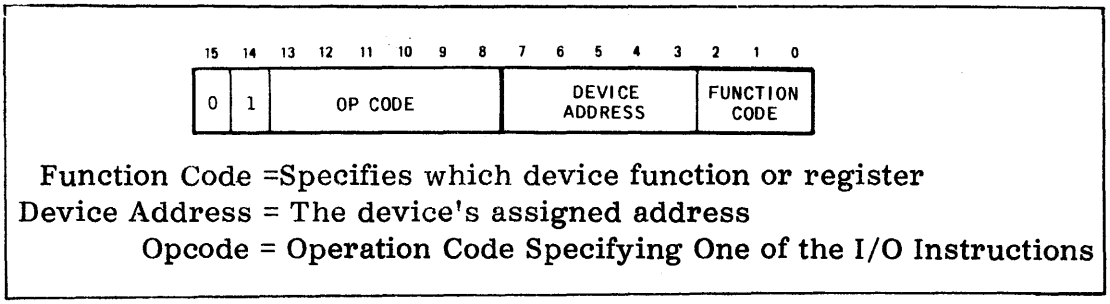


Figure E-9. Input/Output Instruction Machine Code Format

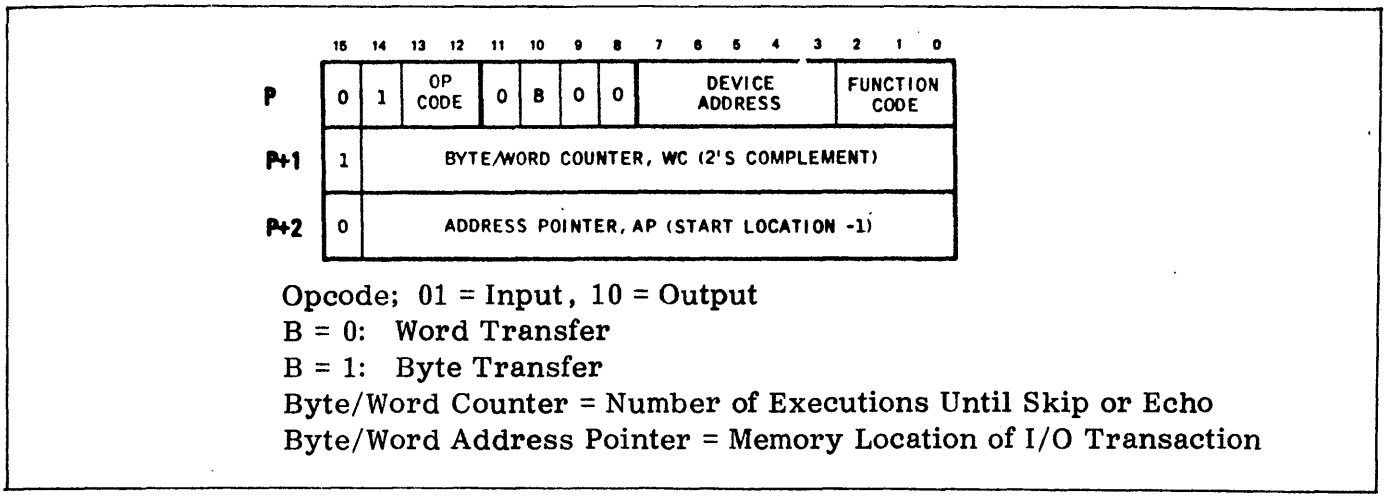


Figure E-10. Automatic Input/Output Instruction Machine Code Format

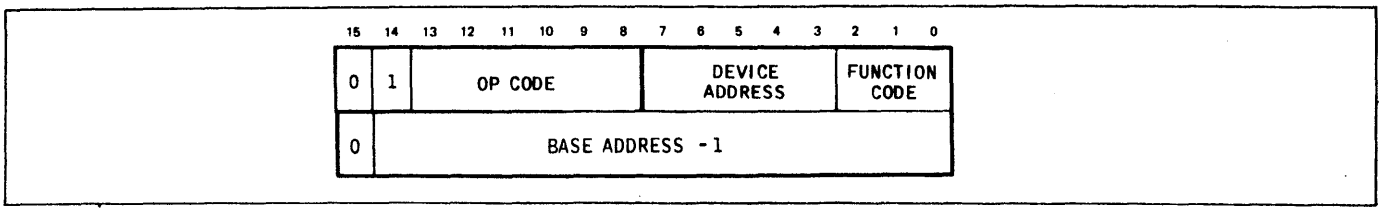


Figure E-11. Block Input/Output Instruction Machine Code Format

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
0000	NOP	No Operation	7	3-29
0008	XRM	X Register to Minus One	7	3-26
0010	ARM	A Register to Minus One	7	3-26
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	3-27
0070	ANA	AND of A and X to A	7	3-27
0090	IPX	Increment P to X	7	3-28
00A8	DXR	Decrement X Register	7	3-26
00B0	DXA	Decrement X to A	7	3-28
00C8	DAX	Decrement A to X	7	3-28
00D0	DAR	Decrement A Register	7	3-26
0108	ZXR	Zero X Register	7	3-26
0110	ZAR	Zero A Register	7	3-26
0118	ZAX	Zero A and X	7	3-27
0128	IXR	Increment X Register	7	3-26
0130	IXA	Increment X to A	7	3-28
0148	IAX	Increment A to X	7	3-28
0150	IAR	Increment A Register	7	3-26
0208	CAX	Complement of A to X	7	3-28
0210	CAR	Complement A Register	7	3-26
0308	NAX	Negate A to X	7	3-28
0310	NAR	Negate A Register	7	3-26

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
0350	ARP	A Register to Plus One	7	3-26
0358	AXP	A and X Registers to Plus One	7	3-28
0408	CXR	Complement X Register	7	3-26
0410	CXA	Complement of X to A	7	3-28
0428	EAX	Exchange A and X	7	3-27
0508	NXR	Negate X Regster	7	3-26
0510	NXA	Negate X to A	7	3-28
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	3-31
0B00	AAI*	Add to A Immediate	3	3-18
0C00	DIN	Disable Interrupts	8	3-31
0D00	SAI*	Subtract from A Immediate	3	3-18
0E00	SBM	Set Byte Mode	8	3-30
0F00	SWM	Set Word Mode	8	3-30
1028	ALX*	Arithmetic Shift X Left	5	3-22
1050	ALA*	Arithmetic Shift A Left	5	3-22
10A8*	ARX*	Arithmetic Shift X Right	5	3-22
10D0	ARA*	Arithmetic Shift A Right	5	3-22
1128	RLX*	Rotate X Left with Overflow	5	3-23

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
1150	RLA*	Rotate A Left with Overflow	5	3-23
11A8	RRX*	Rotate X Right with Overflow	5	3-24
11D0	RRA*	Rotate A Right with Overflow	5	3-24
1200	ROV	Reset Overflow	5	3-26
1320	BXO*	Bit of X to Overflow	5	3-27
1320	SXO	Sign of X to Overflow	5	3-27
1328	LLX*	Logical Shift X Left	5	3-23
1340	BAO*	Bit of A to Overflow	5	3-27
1340	SAO	Sign of A to Overflow	5	3-27
1350	LLA*	Logical Shift A Left	5	3-23
13A0	LXO	LSB of X to Overflow	5	3-27
13A8	LRX*	Logical Shift X Right	5	3-23
13C0	LAO	LSB of A to Overflow	5	3-27
13D0	LRA*	Logical Shift A Right	5	3-23
1400	SOV	Set Overflow	5	3-26
1600	COV	Complement Overflow	5	3-27
1900	LRL*	Long Rotate Left	6	3-25
1940	NRM	Normalize A and X	2	3-17
1960	MPY	Multiply and Add	2	3-16
1970	DVD	Divide	2	3-15
1980	LRR*	Long Rotate Right	6	3-25
1B00	LLL*	Long Logical Shift Left	6	3-24
1B80	LLR*	Long Logical Shift Right	6	3-24

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
2080-3F80 Forward 20C0-3FC0 Backward	JOC*	Jump on Condition	4	3-20
2080 Forward 20C0 Backward	JAM*	Jump if A Minus	4	3-21
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
2200 Forward 2240 Backward	JOS*	Jump if Overflow Set	4	3-21
2400 Forward 2440 Backward	JSR*	Jump if Sense Switch Reset	4	3-21
2800 Forward 2840 Backward	JXZ*	Jump if X Zero	4	3-21
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	3-21
3400 Forward 3440 Backward	JSS*	Jump if Sense Switch Set	4	3-21
3800 Forward 3840 Backward	JXN*	Jump if X Not Zero	4	3-21
4000	SEL*	Select Function	9	3-23
4002	PFE	Power Fail Enable	9	3-31
4003	PFD	Power Fail Disable	9	3-32

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
4005	CIE	Console Interrupt Enable	9	3-31
4006	CID	Console Interrupt Disable	9	3-31
4007	TRP	Trap	9	3-32
4400	SEA*	Select and Present A	9	3-33
4404	OCA	Output A to Console Register	9	3-29
4600	SEX*	Select and Present X	9	3-33
4604	OCX	Output X to Console Register	9	3-29
4800	SSN*	Sense and Skip On No Response	9	3-33
4900	SEN*	Sense and Skip On Response	9	3-33
5000	AIN*	Automatic Input Word to Memory	10	3-41
5400	AIB*	Automatic Input Byte to Memory	10	3-41
5800	INA*	Input to A Register	9	3-34
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	3-35
5A00	INX*	Input to X Register	9	3-34
5A00	SIX	Status Input to X	9	3-30
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	3-35
5C00	INAM*	Input to A Register Masked	9	3-34
5D00	RDAM*	Read Word to A Register Masked	9	3-35

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
5E00	INXM*	Input to X Register Masked	9	3-34
5F00	RDXM*	Read Word to X Register Masked	10	3-35
6000	AOT*	Automatic Output Word From Memory	10	3-41
6400	AOB*	Automatic Output Byte From Memory	10	3-41
6800	OTZ*	Output Zero	9	3-34
6800	SIN*	Status Inhibit	8	3-30
6900	WRZ*	Write Zero	9	3-35
6C00	OTA*	Output A Register	9	3-34
6C00	SOA	Status Output From A	9	3-30
6D00	WRA*	Write From A Register	9	3-35
6E00	OTX*	Ouptut X Register	3	3-34
6E00	SOX	Status Output From X	9	3-30
6F00	WRX*	Write From X REgister	3	3-35
7100	BIN*	Block In	11	3-38
7500	BOT*	Block Out	11	3-38
7800	IBA*	Input Byte to A Register	9	3-36
7900	RBA*	Read Byte to A Register	9	3-37
7A00	IBX*	Input Byte to X Register	9	3-36
7B00	RBX*	Read Byte to X Register	9	3-37
7C00	IBAM*	Input Byte to A Register Masked	9	3-36
7D00	RBAM*	Read Byte to A Register Masked	9	3-37
7E00	IBXM*	Input Byte to X Register Masked	9	3-36
7F00	RBXM*	Read Byte to X Register Masked	9	3-37

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
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 to Byte to A	1	3-10
9000	SUB*	Subtract From A	1	3-10
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	3-18
C100	CXI*	Compare to X Immediate	3	3-18
C200	AXI*	Add to X Immediate	3	3-18
C300	SXI*	Subtract From X Immediate	3	3-18
C400	LXP*	Load X Positive Immediate	3	3-19
C500	LXM*	Load X Minus Immediate	3	3-19
C600	LAP*	Load A Positive Immediate	3	3-19
C700	LAM*	Load A Minus Immediate	3	3-19

Instruction Skeleton in Hex	Instruction Mnemonic	Description	Machine Code Format	Page
CD00	SCM*	Scan Memory	1	3-13
CD00	SCMB*	Scan Memory Byte	1	3-14
D000	CMS*	Compare and Skip if High or Equal	1	3-12
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 F
ALPHA LSI EXECUTION TIMES

F.1 GENERAL

This Appendix defines the execution time of each instruction in the ALPHA LSI instruction set. Two Processors and 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 memory parameter substitution.

F.2 MEMORY PARAMETERS

Currently, four memories are offered in the ALPHA LSI family, three of these are core memories, while the fourth is a semiconductor memory. Table F-1 lists the parameters of these memories. All times listed are in nanoseconds.

Table F-1. LSI Family Memory Parameters

Memory Type	Configuration	C	RA	RO	WA	WO	M	M'	ROI	WOI
Core 980	Add on 4K, 8K	980	380	600	180	800	600	400	220	420
Core 1200	Add on 16K	1200	400	800	200	1000	600	400	300	500
Core 1600	Add on or integral 4K, 8K	1600	450	1150	250	1350	600	400	0	0
SC 1200	Add on 2K, 4K, 8K Integral 2K, 4K	1200	500	700	200	1000	600	400	0	0

Parameters in nanoseconds are:

C = Cycle Time
RA = Read Access
RO = Read Overhead
WA = Write Access
WO = Write Overhead
M = LSI-1 Effective Read Access
M' = LSI-1 Effective Write Access
ROI = Interleaved Effective Read Overhead
WOI = Interleaved Effective Write Overhead

F.3 LSI-1 EXECUTION TIME ALGORITHMS

The LSI-1 execution time algorithms are listed in table F-2. The algorithms are partitioned by class and subclass. Numerous instructions have two times listed with the reason for the dual listing given in parenthesis. All numeric values are in microseconds. The value of A (address calculation time) is derived from the list of addressing modes at the beginning of the table. The variables m and m' are derived from table F-1 and are in nanoseconds.

The letter i stands for indirect address levels. Where indirect addressing is used, the value (3.2 + m)i must be added for each level of indirect addressing that is employed.

The letter n denotes a shift. The value 1.6n or 3.2n must be added to the basic execution time of shift instructions for each bit shifted.

The letter w is used by the SCM and Block I/O instructions. The parenthetical expression which precedes the w is the time calculation on a per word basis.

Table F-2. LSI-1 Execution Time Algorithms

MEMORY REFERENCE CLASS	
A = Address Calculation Time for Memory Reference Instructions:	
DIRECT SCRATCHPAD	1.6 + m
DIRECT RELATIVE	1.6 + m
DIRECT INDEXED	3.2 + m
INDIRECT SCRATCHPAD	(3.2 + m) i
INDIRECT REALTIVE	1.6 + (3.2 + m) i
INDIRECT INDEXED	1.6 + (3.2 + m) i
ARITHMETIC	
ADD	6.4 + m + A
SUB	6.4 + m + A
LOGICAL	
AND	6.4 + m + A
IOR	6.4 + m + A
XOR	6.4 + m + A
DATA TRANSFER	
LDA	4.8 + m + A
LDX	4.8 + m + A
STA	4.8 + m' + A
STX	4.8 + m' + A
EMA	8.0 + m + m' + A

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

PROGRAM TRANSFER	
CMS	4.8 + A
JST (Non-Interrupt)	8.0 + m' + A
JST (Interrupt)	6.4 + m' + A
IMS	9.6 + m + m' + A
SCM	(12.8 + m + A) w
CMS	12.8 + m + A
DOUBLE WORD MEMORY REFERENCE CLASS	
DVD	118.4 + 3m + (3.2 + m) i
MPY	110.4 + 3m + (3.2 + m) i
NRM (count expires)	17.6 + 3m + m' + 9.6n + (3.2 + m) i
NRM (count does not expire)	20.8 + 3m + m' + 9.6n + (3.2 + m) i
BYTE IMMEDIATE CLASS	
AAI	4.8 + m
AXI	4.8 + m
SAI	4.8 + m
SXI	4.8 + m
CAI	6.4 + m
CXI	6.4 + m
LAP	4.8 + m
LXP	4.8 + m
LAM	4.8 + m
LXM	4.8 + m
CONDITIONAL JUMP CLASS	
MICROCODED	
(JOC)	
ALL Double Register Tests	14.4 + m
ALL Others	6.4 + m
ARITHMETIC	
JAG	6.4 + m
JAP	
JAZ	
JAN	
JAL	
JAM	
JXZ	
JXN	

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

CONTROL	
JSS, JSR, JOS, JOR	6.4 + m
SHIFT CLASS	
ARITHMETIC SHIFTS	
ARA, ARX, ALA, ALX	3.2 + m + 1.6n
LOGICAL SHIFTS	
LRA, LRX, LLA, LLX	3.2 + m + 1.6n
ROTATE SHIFTS	
RRA, RRX, RLA, RLX	3.2 + m + 1.6n
DOUBLE REGISTER LOGICAL SHIFTS	
LLL, LLR	3.2 + m + 3.2n
DOUBLE REGISTER ROTATE SHIFTS	
LRL, LRR	3.2 + m + 3.2n
REGISTER CHANGE CLASS	
A REGISTER CHANGE	
ZAR, ARP, ARM, CAR, NAR, IAR, DAR	4.8 + m

Table F-2. LSI-1 Executive Time Algorithms (Cont'd)

X REGISTER CHANGE	
ZXR	4.8 + m
XRP	
XRM	
CXR	
NXR	
IXR	
DXR	
OVERFLOW REGISTER CHANGE	
SOV	4.8 + m
ROV	4.8 + m
COV	4.8 + m
SAO	6.4 + m
SXO	6.4 + m
LAO	6.4 + m
LXO	6.4 + m
BAO	6.4 + m + 1.6n
BXO	6.4 + m + 1.6n
MULTI-REGISTER CHANGE	
ZAX	6.4 + m
AXP	6.4 + m
AXM	6.4 + m
TAX	4.8 + m
TXA	4.8 + m
EAX	8.0 + m
ANA	4.8 + m
ANX	4.8 + m
NRA	6.4 + m
NRX	6.4 + m
CAX	4.8 + m
CXA	4.8 + m
NAX	4.8 + m
NXA	4.8 + m
IAX	4.8 + m
IXA	4.8 + m
IPX	4.8 + m
DAX	4.8 + m
DXA	4.8 + m
CONSOLE REGISTER	
ICA	5.6 + m
ICX	
ISA	
ISX	
OCA	
OCX	

Table F-2. LSI-1 Executive Time Algorithms (Cont'd)

CONTROL CLASS	
PROCESSOR CONTROLS	
NOP	4.8 + m
HLT (STOP)	4.8 + m
MODE CONTROLS	
SBM	4.8 + m
SWM	4.8 + m
STATUS CONTROLS	
SIN	5.6 + m
SIA	5.6 + m
SIX	5.6 + m
SOA	5.6 + m
SOX	5.6 + m
INTERRUPT CONTROLS	
EIN	4.8 + m
DIN	6.4 + m
CIE	5.6 + m
CID	5.6 + m
PFE	5.6 + m
PFD	5.6 + m
TRP	5.6 + m
INPUT/OUTPUT CLASS	
CONTROL	
SEL	5.6 + m
SEA	5.6 + m
SEX	5.6 + m
SEN	7.2 + m
SSN	7.2 + m
UNCONDITIONAL WORD	
INA	5.6 + m
INAM	7.2 + m
INX	5.6 + m
INXM	7.2 + m
OTA	5.6 + m
OTX	5.6 + m
OTZ	5.6 + m

Table F-2. LSI-1 Execution Time Algorithms (Cont'd)

CONDITIONAL WORD	
RDA	7.2 + m
RDAM	10.4 + m
RDX	7.2 + m
RDXM	10.4 + m
WRA	7.2 + m
WRX	7.2 + m
WRZ	7.2 + m
UNCONDITIONAL BYTE	
IBA	7.2 + m
IBAM	8.8 + m
IBX	7.2 + m
IBXM	8.8 + m
CONDITIONAL BYTE	
RBA	10.4 + m
RBAM	12.0 + m
RBX	10.4 + m
RBXM	12.0 + m
BLOCK	
BIN	11.2 + 2m + (7.2 + m) w
BOT	11.2 + 2m + (7.2 + m) w
AUTOMATIC	
AIN	23.2 + 5m - 2m + 3m'
AIN (Under Interrupts)	20.0 + 5m - 2m + 3m'
AOT	23.2 + 5m - 3m + 2m'
AOT (Under Interrupts)	20.0 + 5m - 3m + 2m'
AIB	23.2 + 5m - 2m + 3m'
AIB (Under Interrupts)	20.0 + 5m - 2m + 3m'
AOB	23.2 + 5m - 3m + 2m'
AOB (under Interrupts)	20.0 + 5m - 3m + 2m'

F.4 LSI-2 EXECUTION TIME ALGORITHMS

The LSI-2 execution time algorithms are listed in table F-3. The algorithms are partitioned by class and subclass as in table F-2. The memory reference instruction address calculation times precede the instruction execution algorithms. Note that four different sets of address calculations are provided. The list of memory reference instructions have algorithms which list A_1, A_2, A_3, or A_4. The appropriate address calculation variable should be used as indicated.

All memories may be overlapped to achieve higher transfer rates. Core 1600 and SC1200 may be overlapped 100 percent to achieve twice the data transfer rate of a single memory module. Core 1200 and Core 980 may be overlapped to achieve a maximum transfer rate of 171 and 163 percent, respectively, of a single memory. Overlapping is always effective for DMA operation.

Overlapping is effective for LSI-2 as indicated by the execution time equations. Terms of the form n/RO or m/WO mean that the larger of the two times indicated are to be used. When overlapping is achieved by alternate memory accesses in different memory modules, the overhead times are masked and the effective RO and WO become zero except for Core 980 and Core 1200 which have an overhead time even when interleaved. This overhead time prevents these high current memories from overloading the power supply.

As in table F-2, numerous instructions have several times listed to define variations of an instruction. The symbols i, n, and w are described in paragraph F.3.

Table F-3. LSI-2 Execution Time Algorithms

MEMORY REFERENCE CLASS

PROCESSOR MODE	ADDRESSING MODE	A_1	A_2
WORD	direct scratchpad	RA + 700/RO	RA + 800/RO
	direct relative forward	RA + 700/RO	RA + 800/RO
	direct relative backward	RA + 850/RO	RA + 950/RO
	direct indexed	RA + 700/RO	RA + 800/RO
	indirect scratchpad	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 400/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 500/RO + (RA + 400/RO) (i-1)
	indirect relative indexed	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RP + (RA + 400/RO) (i-1)
BYTE	direct scratchpad	RA + 1000/RO	RA + 1100/RO
	direct relative	RA + 700/RO	RA + 800/RO
	direct indexed	RA + 1000/RO	RA + 1100/RO
	indirect scratchpad	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 800/RO + (RA + 400/RO) (i-1)
	indirect indexed	2RA + 700/RO + 900/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)

PROCESSOR MODE	ADDRESSING MODE	A_3	A_4
WORD	direct scratchpad	RA + 1000/RO	RA + 1300/RO
	direct relative forward	RA + 1000/RO	RA + 1300/RO
	direct relative backward	RA + 1150/RO	RA + 1450/RO
	direct indexed	RA + 1000/RO	RA + 1300/RO
	indirect scratchpad	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 700/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)
	indirect relative indexed	2RA + 700.RO + 1200/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1500/RO + (RA + 400/RO) (i-1)
BYTE	direct scratchpad	RA + 1300/RO	RA + 1600/RO
	direct relative	RA + 1000/RO	RA + 1300/RO
	direct indexed	RA + 1300/RO	RA + 1600/RO
	indirect scratchpad	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect relative forward	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect relative backward	2RA + 700/RO + 1000/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1300/RO + (RA + 400/RO) (i-1)
	indirect indexed	2RA + 700/RO + 1200/RO + (RA + 400/RO) (i-1)	2RA + 700/RO + 1500/RO + (RA + 400/RO) (i-1)

A_1 is used with ADD, SUB, AND, IOR, XOR, EMA, LDA, LDX, CMS and IMS.

A_2 is used with STA, STX and JST.

A_3 is used by JMP only.

A_4 is used by SCM only.

ARITHMETIC
ADD
SUB

LOGICAL
AND
IOR
XOR

A_1 + RA + (400/RO)

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

DATA TRANSFER	
LDA	A_1 + RA + 400/RO
LDX	A_1 + RA + 400/RO
STA	A_2 + WA + 250/RO
STX	A_2 + WA + 250/RO
EMA	A_1 + RA + 500/RO* + WA + 550/WO
PROGRAM TRANSFER	
JMP	A_3
JST (Non-Interrupt)	A_2 + WA + 550/WO
JST (Interrupt)	A_2 + WA + 700/WO
IMS	A_1 + RA + (500/RO)* + WA + 700/RO ≠ 0 in line no skip or + 1450/RO = 0 in line skip or + 850/RO ≠ 0 interrupt no echo or + 1600/RO = 0 interrupt echo
SCM	A_4 + RA + 550/RO + (RA + 1600/RO) (w-1)
CMS	A_1 + RA + 550/RO for A < Y or + 850/RO for A = Y or +1150/RO for A > Y
DOUBLE WORD MEMORY REFERENCE CLASS	
DVD	2RA + 1000/RO + (RA + 400/RO) i + (2950 + 450n)/RO
MPY	2RA + 1000/RO + (RA + 400/RO) i + (3100** + 600n)/RO
NRM	2RA + 1000/RO + (RA + 400/RO) i + (1400 + 600n)/RO + WA + 1750/WO
BYTE IMMEDIATE CLASS	
AAI	RA + 1000/RO
AXI	RA + 700/RO
SAI	RA + 1000/RO
SXI	RA + 700/RO
CAI, CXI	RA + 1000/RO skip RA + 850/RO no skip
LAP	RA + 700/RO
LXP	RA + 700/RO
LAM	RA + 700/RO
LXM	RA + 700/RO

* Not Affected By Overlap

** +300 for Negative Multiplier

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

CONDITIONAL JUMP CLASS	
MICROCODED JOC	
ARITHMETIC JAG JAL JAM JAP JAZ JXN JXZ	RA + 700/RO = No Jump RA + 1000/RO = Jump
CONTROL JOR JOS JSR JSS	

SHIFT CLASS	
ARITHMETIC SHIFTS ALA ALX ARA ARX	
LOGICAL SHIFTS LLA LLX LRA LRX	RA + 1150 + 150n/RO
ROTATE SHIFTS RLA RLX RRA RRX	

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

DOUBLE REGISTER LOGICAL SHIFTS LLL LLR DOUBLE REGISTER ROTATE SHIFTS LRL LRR	RA + 2350 + 150n/RO
REGISTER CHANGE CLASS	
A REGISTER CHANGE ZAR ARP ARM CAR NAR IAR DAR	RA + 1000/RO
X REGISTER CHANGE ZXR XRP XRM CXR NXR IXR DXR	RA + 1000/RO
OVERFLOW REGISTER CHANGE SOV ROV COV	RA + 850/RO
SAO SXO LAO LXO BAO BXO	RA + 1300 + 150n/RO n is number of bits away from 0 to 15

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

REGISTER CHANGE CLASS (Cont'd)	
MULTI-REGISTER CHANGE	
ZAX	RA + 1300/RO
AXP	RA + 1300/RO
AXM	RA + 1300/RO
TAX	RA + 1000/RO
TXA	RA + 1000/RO
EAX	RA + 1300/RO
ANA	RA + 1000/RO
ANX	RA + 1000/RO
NRA	RA + 1000/RO
NRX	RA + 1000/RO
CAX	RA + 1000/RO
CXA	RA + 1000/RO
NAX	RA + 1300/RO
NXA	RA + 1300/RO
IAX	RA + 1000/RO
IXA	RA + 1000/RO
IPX	RA + 1000/RO
DAX	RA + 1000/RO
DXA	RA + 1000/RO
CONSOLE REGISTER	
ICA, ICX, ISA, ISX, OCA, OCX	RA + 1600/RO

CONTROL CLASS	
PROCESSOR CONTROLS	
HLT (STOP), NOP	RA + 1150/RO
MODE CONTROLS	
SBM, SWM	RA + 1000/RO
STATUS CONTROLS	
SIA, SIX, SIN, SOA, SOX	RA + 1600/RO

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

INTERRUPT CONTROLS	
CID, CIE	RA + 1600/RO
DIN, EIN	RA + 850/RO
PFE, PFE, TRP	RA + 1600/RO
INPUT/OUTPUT CLASS	
CONTROL	
SEN	RA + 1550/RO no skip RA + 1900/RO skip
SEA	RA + 1600/RO
SEL	RA + 1600/RO
SEX	RA + 1600/RO
SSN	RA + 1900/RO no skip RA + 1700/RO skip
UNCONDITIONAL WORD	
INA, INAM, INX, INXM, OTA, OTX, OTZ	RA + 1600/RO
CONDITIONAL WORD	
RDA, RDAM, RDX, RDXM, WRA, WRX, WRZ	RA + 2050/RO successful RA + 2000/RO unsuccessful repeat period
UNCONDITIONAL BYTE	
IBA, IBAM, IBX, IBXM	RA + 1600/RO

Table F-3. LSI-2 Execution Time Algorithms (Cont'd)

CONDITIONAL BYTE	
RBA, RBAM, RBX, RBXM	RA + 2050/RO successful RA + 2000/RO unsuccessful repeat period
BLOCK (1)	
BIN	2RA + 400/RO + 1550/RO + WA + 850/WO + (WA + 2000/WO) (W-1)
BOT	3RA + 2 (400/RO) + 1300/RO + + (RA + 2050/RO) (W-1)
AUTOMATIC	
AIN/AIB	3RA + 3WA + 400/RO + 800/RO + 500/RO* + 550/WO + 1700/WO + 550/WO if inline, or + 400/WO if interrupt
AOT/AOB	4RA + 2WA + 400/RO + 800/RO + 500/RO* + 2(550/WO) + 1750/RO inline, or + 1600/RO if interrupt

Note: (1) Time given assuming device is always ready. If not ready, BIN and BOT retest for ready every 850 ns.

* Not Affected By Overlap
** (1050/WO) if WC = 0

F.5 ALPHA LSI FAMILY INSTRUCTION EXECUTION TIMES

The execution times of the ALPHA LSI instruction set is listed in table F-6. The address calculation times for the LSI-1 and LSI-2 are listed in tables F-4 and F-5, respectively.

Table F-4. LSI-1 Address Calculation Times

DIRECT SCRATCHPAD	2.2
DIRECT RELATIVE	2.2
DIRECT INDEXED	3.8
INDIRECT SCRACTHPAD	3.8i
INDIRECT RELATIVE	1.6 + 3.8i
INDIRECT INDEXED	1.6 + 3.8i

Table F-5. LSI-2 Address Calculation Times

MEMORY TYPE	PROCESSOR MODE	ADDRESSING MODE	A_1	A_2	A_3	A_4
CORE 1600	WORD	direct scratchpad	1.6	1.6	1.6	1.75
		direct relative forward	1.6	1.6	1.6	1.75
		direct relative backward	1.6	1.6	1.6	1.9
		direct indexed	1.6	1.6	1.6	1.75
		indirect scratchpad	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect relative forward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect relative backward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)
		indirect indexed	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
	BYTE	direct scratchpad	1.6	1.6	1.75	2.05
		direct relative	1.6	1.6	1.6	1.75
		direct indexed	1.6	1.6	1.75	2.05
		indirect scratchpad	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect relative forward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect relative backward	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.35 + 1.6 (i-1)
		indirect indexed	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.2 + 1.6 (i-1)	3.55 + 1.6 (i-1)
CORE 1200	WORD	direct scratchpad	1.2	1.2	1.4	1.7
		direct relative forward	1.2	1.2	1.4	1.7
		direct relative backward	1.25	1.35	1.55	1.85
		direct indexed	1.2	1.2	1.4	1.7
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)
		indirect indexed	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
	BYTE	direct scratchpad	1.4	1.5	1.7	2.0
		direct relative	1.2	1.2	1.4	1.7
		direct indexed	1.4	1.5	1.7	2.0
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.9 + 1.2 (i-1)
		indirect indexed	2.5 + 1.2 (i-1)	2.6 + 1.2 (i-1)	2.8 + 1.2 (i-1)	3.1 + 1.2 (i-1)
CORE 980	WORD	direct scratchpad	1.08	1.18	1.38	1.68
		direct relative forward	1.08	1.18	1.38	1.68
		direct relative backward	1.23	1.33	1.53	1.83
		direct indexed	1.08	1.18	1.38	1.68
		indirect scratchpad	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect relative forward	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect relative backward	2.06 + .98 (i-1)	2.06 + .98 (i-1)	2.16 + .98 (i-1)	2.46 + .98 (i-1)
		indirect indexed	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
	BYTE	direct scratchpad	1.38	1.48	1.68	1.98
		direct relative	1.08	1.18	1.38	1.68
		direct indexed	1.38	1.48	1.68	1.98
		indirect scratchpad	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect relative forward	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect relative backward	2.16 + .98 (i-1)	2.26 + .98 (i-1)	2.46 + .98 (i-1)	2.76 + .98 (i-1)
		indirect indexed	2.36 + .98 (i-1)	2.46 + .98 (i-1)	2.66 + .98 (i-1)	2.96 + .98 (i-1)
SC1200	WORD	direct scratchpad	1.2	1.3	1.5	1.8
		direct relative forward	1.2	1.3	1.5	1.8
		direct relative backward	1.35	1.45	1.65	1.95
		direct indexed	1.2	1.3	1.5	1.8
		indirect scratchpad	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.4 + 1.2 (i-1)	2.7 + 1.2 (i-1)
		indirect indexed	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-10	3.0 + 1.2 (i-1)
	BYTE	direct scratchpad	1.5	1.6	1.8	2.1
		direct relative	1.2	1.3	1.5	1.8
		direct indexed	1.5	1.6	1.8	2.1
		indirect scratchpad	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect relative forward	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect relative backward	2.4 + 1.2 (i-1)	2.5 + 1.2 (i-1)	2.7 + 1.2 (i-1)	3.0 + 1.2 (i-1)
		indirect indexed	2.6 + 1.2 (i-1)	2.7 + 1.2 (i-1)	2.9 + 1.2 (i-1)	3.2 + 1.2 (i-1)

A_1 is used with ADD, SUB, AND, IOR, XOR, EMA, LDA, LDX, CMS and IMS.

A_2 is used with STA, STX and JST.

A_3 is used by JMP only.

A_4 is used by SCM only.

Table F-6. ALPHA LSI Family Instruction Execution Times

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
MEMORY REFERENCE					
Arithmetic					
ADD	A + 7	A_1 + 1.6	A_1 + 1.2	A_1 + 0.98	A_1 + 1.2
ADDB	A + 7				
SUB	A + 7				
SUBB	A + 7				
Logic					
AND	A + 7				
ANDB	A + 7				
IOR	A + 7				
IORB	A + 7				
XOR	A + 7				
XORB	A + 7				
Data Transfer					
LDA	A + 5.4				
LDAB	A + 5.4				
LDX	A + 5.4				
LDXB	A + 5.4				
STA	A + 5.2	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
STAB	A + 5.2				
STX	A + 5.2				
STXB	A + 5.2				
EMA	A + 9	A_1 + 3.2	A_1 + 2.4	A_1 + 1.96	A_1 + 2.4
EMAB	A + 9	A_1 + 3.2	A_1 + 2.4	A_1 + 1.96	A_1 + 2.4
Program Transfer					
CMS	A + 13.4	A_1 + 1.6	A_1 + (1.2 or 1.55)	A_1 + (0.98 or 1.53)	A_1 + (1.2 or 1.65)
CMSB	A + 13.4	A_1 + 1.6	A_1 + (1.2 or 1.55)	A_1 + (0.98 or 1.53)	A_1 + (1.2 or 1.65)
IMSN	A + 10.6	A_1 + (3.0 or 3.3)	A_1 + (2.2 or 2.85)	A_1 + (1.86 or 2.61)	A_1 + (2.1 or 2.85)
IMSI	A + 10.6	A_1 + (3.0 or 3.45)	A_1 + (2.25 or 3.0)	A_1 + (2.01 or 2.76)	A_1 + (2.25 or 3.0)
JMP	A + 4.8	A_3	A_3	A_3	A_3
JSTN	A + 8.4	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
JSTI	A + 6.8	A_2 + 1.6	A_2 + 1.2	A_2 + 0.98	A_2 + 1.2
SCM	A + 13.4	A_4 + (1.6 + 2.05W)	A_4 + (1.2 + 2.0W)	A_4 + (0.98 + 1.98W)	A_4 + (1.2 + 2.1W)
SCMB	A + 13.4	A_4 + (1.6 + 2.05W)	A_4 + (1.2 + 2.0W)	A_4 + (.098 + 1.98W)	A_4 + (1.2 + 2.1W)
DOUBLE WORD MEMORY REFERENCE					
DVD	3.8i + 120.2	13.35	12.74	12.44	12.9
MPY	3.8i + 112.2	15.75	15.1	14.84	15.3
NRM1	3.8i + 19.8 + 9.6n	7.05 + .6n	6.35 + .6n	6.07 + .6n	6.55 + .6n
NRM2	3.8i + 23 + 9.6n				

Table F-6. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
BYTE IMMEDIATE					
AAI	5.4	1.6	1.4	1.38	1.5
AXI	5.4	1.6	1.2	1.08	1.2
SAI	5.4	1.6	1.4	1.38	1.5
SXI	5.4	1.6	1.2	1.08	1.2
CAI	7	1.6	1.25 or 1.4	1.23 or 1.38	1.35 or 1.5
CXI	7	1.6	1.25 or 1.4	1.23 or 1.38	1.35 or 1.5
LAP	5.4	1.6	1.2	1.08	1.2
LXP	5.4	1.6	1.2	1.08	1.2
LAM	5.4	1.6	1.2	1.08	1.2
LXM	5.4	1.6	1.2	1.08	1.2
CONDITIONAL JUMP					
Microcoded					
JOC1	15				
JOC2	7				
Arithmetic					
JAG	7				
JAP	7				
JAZ	7				
JAN	7				
JAL	7	1.6	1.2 or 1.4	1.08 or 1.38	1.2 or 1.5
JAM	7				
JXZ	7				
JXN	7				
Control					
JOR	7				
JOS	7				
JSR	7				
JSS	7				
SHIFT					
Single Register					
Arithmetic Shifts					
ALA	3.8 + 1.6n	1.6 + .15n	1.55 + .15n	1.53 + .15n	1.65 + .15n
ALX					
ARA					
ARX					

Table F-6. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
SHIFTS (Cont'd)					
Logical Shifts LLA LLX LRA LRX Rotate Shifts RLA RLX RRA RRX	3.8 + 1.6n	1.6 + .15n	1.55 + .15n	1.53 + .15n	1.65 + .15n
Double Register Logical LLL LLR LRL LRR	3.8 + 3.2n	2.8 + .15n	2.75 + .15n	2.73 + .15n	2.85 + .15n
REGISTER CHANGE					
Accumulator ARM ARP CAR DAR IAR NAR ZAR Index ZXR XRP XRM CXR NXR IXR DXR	5.4	1.6	1.4	1.38	1.5

Table F-6. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
		REGISTER CHANGE			
Overflow					
SOV, ROV, COV	5.4	1.6	1.25	1.23	1.35
SAO, SXO, LAO, LXO	7	1.75	1.7	1.68	1.8
BAO, BXO	5.4 + 1.6n	1.75 + 15n	1.7 + 1.5n	1.68 + 15n	1.8 + 15n
Multi-Register					
ZAX, AXP, AXM	7	1.75	1.7	1.68	1.8
TAX	5.4	1.6	1.4	1.38	1.5
TXA	5.4	1.6	1.4	1.38	1.5
EAX	8.6	1.75	1.7	1.68	1.8
ANA	5.4				
ANX	5.4				
NRA	7.0				
NRX	7 0	1.6	1.4	1.38	1.5
CAX	5.4				
CXA	5.4				
NAX	5.4	1.75	1.7	1.68	1.8
NXA	5.4	1.75	1.7	1.68	1.8
IAX, IXA, IPX, DAX, DXA	5.4	1.6	1.4	1.38	1.5
Console Register					
ICA, ICX, ISA, ISX, OCA, OCX	6.2	2.05	2	1.98	2.1

Table F-6. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
CONTROL					
Processor					
NOP, HLT, STOP	5.4	1.6	1.55	1.53	1.65
WAIT	Indefinite	Indefinite	Indefinite	Indefinite	Indefinite
Mode Control					
SBM	5.4	1.6	1.4	1.38	1.5
SWM	5.4	1.6	1.4	1.38	1.5
Status					
SIA, SIN, SIX, SOA, SOX	6.2	2.05	2	1.98	2.1
Interrupts					
EIN	5.4	1.6	1.25	1.23	1.35
DIN	7	1.6	1.25	1.23	1.35
CIE, CID, PFE, PFD, TRP	6.2	2.05	2	1.98	2.1
INPUT/OUTPUT					
Control					
SEN	7.8	2 or 2.35	1.95 or 2.3	1.93 or 2.28	2.05 or 2.4
SSN	7.8	2.15 or 2.35	2.1 or 2.3	2.08 or 2.28	2.2 or 2.4
SEL, SEA, SEX	6.2	2.05	2	1.98	2.1
Unconditional Word					
INA	6.2				
INAM	7.8				
INX	6.2				
INXM	7.8	2.05	2	1.98	2.1
OTA, OTX, OTZ	6.2				

Table F-6. ALPHA LSI Family Instruction Execution Times (Cont'd)

MNEMONIC	LSI-1 C1600 C1200 C980 SC1200	LSI-2 C1600	LSI-2 C1200	LSI-2 C980	LSI-2 SC1200
INPUT/OUTPUT (Cont'd)					
Conditional Word					
RDA	7.8	2.45 or 2.5	2.4 or 2.45	2.38 or 2.43	2.5 or 2.55
RDAM	11				
RDX	7.8				
RDAM	11				
WRA	7.8				
WRX					
WRZ					
Unconditional Byte					
IBA	7.8	2.05	2	1.98	2.1
IBAM	9.4				
IBX	7.8				
IBXM	9.4				
Conditonal Byte					
RBA	11	2.45 or 2.5	2.4 or 2.45	2.38 or 2.43	2.5 or 2.55
RBAM	12.6				
RBX	11				
RBXM	12.6				
Block					
BIN	12.4 + 7.6W	5 + 2.25W	4.2 + 2.2W	3.94 + 2.18W	4.3 + 2.2W
BOT	12.4 + 7.8W	4.95 + 2.5W	4.1 + 2.45W	3.64 + 2.43W	4.2 + 2.55W
Automatic					
AIB	25.6	9.95	7.9	6.98	8
AIBI	22.4				
AIN	25.6				
AINI	22.4				
AOB	25.8	10.2	8.15	7.23	8.35
AOBI	22.6	10.05	8	7.08	8.2
AOT	25.8	10.2	8.15	7.23	8.35
AOTI	22.6	10.05	8	7.08	8.2

F.6 MAXIMUM I/O TRANSFER RATES

The maximum I/O transfer rates for the LSI-1 and LSI-2 are listed in table F-7.

Table F-7. ALPHA LSI Family Maximum Data Transfer Rates

I/O MODE	LSI-1	LSI-2			
	C1600 C1200 C980 SC1200	C1600	C1200	C980	SC1200
DMA (Non Interleaved)	same as LSI-2	625,000 w/s	833,333 w/s	1,020,000 w/s	833,333 w/s
DMA (Interleaved)	same as LSI-2	1,250,000 w/s	1,409,000 w/s	1,666,666 w/s	1,666,666 w/s
Block In	131,579 w/s	444,444 w/s	454,545 w/s	458,711 w/s	454,545 w/s
Block Out	131,579 w/s	400,000 w/s	408,163 w/s	411,522 w/s	392,156 w/s
Programmed In (Cond) Word	34,247 w/s	112,369 w/s	130,718 w/s	136,040 w/s	124,223 w/s
Byte	34,247 b/s	112,369 b/s	125,896 b/s	130,718 b/s	119,760 b/s
Programmed Out (Cond) Word	34,247 w/s	112,994 w/s	131,578 w/s	135,135 w/s	126,582 w/s
Byte	34,247 b/s	112,994 b/s	126,582 b/s	129,870 b/s	122,222 b/s
Programmed In (Memory)	24,631 w/b/s	71,942 w/b/s	85,106 w/b/s	92,678 w/b/s	82,987 w/b/s
Programmed Out (Memory)	24,631 w/b/s	72,727 w/b/s	82,440 w/b/s	90,570 w/b/s	80,645 w/b/s
DMC In	26,738 w/b/s	63,091 w/b/s	74,627 w/b/s	82,101 w/b/s	73,529 w/b/s
DMC Out	26,738 w/b/s	62,111 w/b/s	73,260 w/b/s	81,766 w/b/s	71,684 w/b/s

w/s = words per seconds
b/s = bytes per seconds
w/b/s = words or bytes per seconds

Appendix G

SOFTWARE SUMMARY

G.1 INTRODUCTION

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

Table G-1. Assembler Directives

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

G.2 BOOTSTRAP

To Enter:

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

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
: FFE	3145	3145
: nFFF	0800	0800

G.3 SOFTWARE OPERATION SUMMARY

G.3.1 Autoload

RESET
Enter option control number in Sense Register:

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

To relocate, set X = load address
For Load and Go, set SENSE Switch
Ready Device
AUTO

G.3.2 Binary Loader (BLD)

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

G.3.3 Binary Dump/Verify (BLD/VER)

Load BDP; set P = BDP zero
Set A = Initial location
Set X = Last location
Enter option control number in Sense register:

Mode \ Device		Include EOF		Suppress EOF	
		TTY	HSPT	TTY	HSPT
Punch	Abs	: 0	: 1	: 2	: 3
	Rel	: 8	: 9	: A	: B
Verify	Abs	: 4	: 5	: 6	: 7
	Rel	: C	: D	: E	: F

For transfer address, set SENSE switch
RUN
If Halt (I = : 0802), set A = transfer address, RUN

G.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 option control number in Sense register:

Print Symbols / Load Mode	Defined and Undefined		Defined Only		Undefined Only		Neither
	TTY	LP	TTY	LP	TTY	LP	
Library	:0	:1	:2	:3	:4	:5	:6
Unconditional	:8	:9	:A	:B	:C	:D	:E

Ready tape in reader (TTY or HSPT)
RUN

G.3.5 BETA-4 Assembler

Load BETA-4; set P = :0100; RUN
Enter option control number in Sense Register:

Listing Device / Punch Device	TTY		Line Printer	
	Complete Listing	Error Only	Complete Listing	Error Only
TTY	:0	:1	:2	:3
HSP	:4	:5	:6	:7

To repeat Pass 2, add :8
To flag out-of-range memory reference instruction, set SENSE switch.
Ready source in reader (TTY or HSPT)
RUN

G.3.6 BETA-8 Assembler

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

For / Enter	SI=	LO=	BO=	SD=	P#=
B	BATCH	N/A	N/A	N/A	N/A
L	N/A	N/A	Library	N/A	N/A
X	N/A	Error Only	N/A	N/A	N/A
0	Punch EOF	No Listing	No Binary	No Save	1
1	Keyboard	TTY	TTY	Memory	1
2	TTY	D.P.	Error	Unit 0	2
3	HSPT	Cent.	HSPT	Unit 1	1
4	Card Rdr.	Cent.	TTY	Unit 2	1
5	Card Rdr.	Cent.	TTY	Unit 3	1

G.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.
> Q.	Set ADD function terminator character to n.
> 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.
> XLA. or XLE.	Suppress EOF for current assembly.

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)		

G.3.8 Source Tape Preparation Program

Load STP; set P = STP zero; 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.

G.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.
> La@b.	List contents of locations a through b.
> 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 relcation 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.

G.3.10 Concordance (CONC)

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

SI=

R	Equal Listing
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.

G.3.11 OS Command Summary (DOS, MTOS and COS)

	COMMAND	RESPONSE
1.	/ASSIGN	unit=device [, unit=device ...]
2.	/BATCH	device
3.	/BEGIN	address [, parameters ...]
4.	/CANCEL	
5.	/COMMENT	
6.	/DATE *date	[mm/dd/yy]
7.	/EXEC	program-name [,parameters ...]
8.	/JOB *date, time	
9.	/LOAD	program-name
10.	/LIST *date, time *lu pu	
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 Status

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