LSI-11 WCS
user's guide

EK-KUV11-TM-001

digital equipment corporation • maynard, massachusetts

1st Edition, June 1978

Printed in U.S.A.

PREFACE

This manual provides the user with all the information required to write, assemble, debug and execute microprograms on the LSI-11 utilizing the Writable Control Store (WCS) option (KUV11-AA) in conjunction with microprogramming support software.

Chapter 1 provides an introduction to microprogramming the LSI-11. It discusses machine-micromachine relationships and supplies an introductory glossary of important definitions.

Chapter 2 is divided into two parts: LSI-11 machine architecture and LSI-11 machine operation. The first part is an LSI-11 family hardware overview and is included for completeness. The new LSI-11 user can gain a basic system understanding from this overview, but should refer to the Microcomputer Handbook for a more complete description. The second part is recommended reading for all LSI-11 microprogrammers because it introduces control flow diagrams which explain the transfer of control between the LSI-11 machine and micromachine.

Chapter 3 follows the same organization as Chapter 2, but concentrates on the LSI-11 micromachine. The first part covers the LSI-11 micro processor as implemented in the Control and Data chips and details their internal organization. The second part describes the micromachine operation and its relationship to higher level machine operation.

Chapter 4 presents the LSI-11 microinstruction set, organized on a functional basis. The chapter also contains a detailed explanation of each microinstruction along with an example containing assembly mnemonics and assembled octal equivalents.

Chapter 5 concentrates on the Data Access group of microinstructions and provides details sufficient to accurately determine the execution times for microprogrammed I/O transactions.

Chapter 6 presents the LSI-11 Writable Control Store hardware and its relationship to the LSI-11 micromachine. The discussion includes a description of the data buffer and control/status registers.

Chapter 7 details the microprogramming support software consisting of (1) the microassembler (MICRO), (2) the WCS loader dumper (WCSLOD) and (3) the WCS Microprogram Octal Debug Tool (MODT).

Chapter 8 discusses techniques which may be used to write microprograms.

Chapter 9 contains information on the WCS module installation and checkout.

Chapter 10 contains information on WCS maintenance.

TABLE OF CONTENTS

Chapter 2 THE LSI-11 MACHINE STRUCTURE

CHAPTER 1

INTRODUCTION TO LSI-11 USER MICROPROGRAMMING

1.1 GENERAL

This chapter provides an introduction to LSI-11 microprogramming. Through microprogramming, the user has access to nearly all the resources and techniques used to implement the LSI-11 architecture in four LSI (Large Scale Integration) chips. Section 1.6 lists other documents referenced by this manual.

1.1.1 Introductory Microprogramming Glossary

This section contains a glossary of selected terms which provide an introduction to the the terminology used to describe LSI-11 microprogramming concepts.

Microprocessor	The microprocessor is implemented as a two-chip set of large scale integrated (LSI) circuits called the Data and Control chips. The Data chip contains the Arithmetic Logic Unit (ALU), the Register File and interconnection to the LSI-11 system bus data/address lines (DAL). The Control chip provides the system bus control lines and contains the translation array.
Micromachine	The micromachine consists of a two-chip microprocessor, two or three MICROMs (Control Store chips), and the LSI-11 system bus interface logic. The first two MICROMS contain the PDP-11 emulation microprogram as well as the console ODT microprogram. The optional KEV11 MICROM contains microprograms which execute the extended and floating point machine instructions.
Machine	The LSI-11 machine encompasses the LSI-11 Micromachine, main or program memory and input/output devices.
Machine Cycle	Machine cycle refers to the smallest complete cycle of operations performed by the LSI-11 machine. The three fundamental operations of the

	machine cycle are (1) Fetch machine instruction, (2) Execute machine instruction (3) Trap and interrupt service. The number and type of micromachine operations which must be performed during a given machine cycle are a function of the fetched machine instruction and whether a machine interrupt or system fault is present. Consequently, the time required to complete the machine cycle is variable.
Microcycle (Micromachine Cycle)	Microcycle refers to the smallest complete cycle of operation performed by the LSI-11 micromachine. A machine cycle is composed of one or more microcycles. A microcycle consists of four equal phases (PH1-PH4).
Machine Instruction (Machine Code)	A machine instruction is a 16-bit word stored in main memory. Machine instructions are commonly referred to as instructions.
Microinstruction (Microcode)	A microinstruction is a 22-bit word stored in Control Store.
Microassembler	The microassembler is a microprogram development tool which accepts as input a source file containing micromachine assembly language statements with optional comments. The microassembler output is an object file which may be loaded into the Writable Control Store.
Fetch	Fetch is the operation of presenting an address to memory and subsequently moving the contents of the addressed location to the instruction register contained in the processor.
Execute	The action of performing all operations required by a machine instruction.
Interrupt	The action of diverting control from the normal (uninterrupted) machine operating cycle of repeated Fetch-Execute events. The simplest complete operating cycle is expressed as Fetch-Execute-Interrupt.
Program Counter	The Program Counter (PC) stores the main memory address of the next machine instruction to be feched during a machine cycle. It is automatically incremented by 2 as the last part of the Fetch Machine Instruction phase.
Microprogram Counter	The Microprogram Location Counter (LC) stores the address of the next microinstruction to be fetched during a microcycle. The Location Counter may be loaded from various sources, depending upon the type of microinstruction being executed.
Program Store	Program Store refers to the LSI-11 machine main}

oth program and data in-

formation. Program store is accessed via the LSI-11 system bus. Program store may alternately be referred to as Main Store, Memory or sometimes as core.

Control Store — Control store refers to the storage area for microcode and consists, potentially, of 2048 locations. Control store for the LSI-11 machine is provided by two or more MICROMs and the WCS module.

Writable Control Store — Writable Control Store (WCS) refers to a control store which may be altered by the user. Alterable or writable control store is implemented by read/write random access memory (RAM) devices which can be accessed by both the LSI-11 System Bus and the microinstruction bus.

Data Chip — All Data processing occurs within the Data chip. It contains the ALU and the internal registers. It also provides bi-directional connection to the LSI-11 system bus data/address lines (DAL).

Control Chip — The Control chip functions as a controller/sequencer. It provides all control signals for the system bus and also determines which microinstructions are executed. An important feature of the ConTrol chip is the translation Array. This is a large combinatorial logic network which accepts the machine instruction as input and outputs the addresses of the microprogrammed routine appropriate to that instruction.

MICROM Chip — A MICROM (MICrocode Read Only Memory) chip is the unalterable, non-volatile read only control store memory. One or more MICROMs are connected to the Data and Control chips via the microinstruction bus (MIB).

System Bus — The LSI-11 system bus is the common, bidirectional path which passes address, data and control information between the LSI-11 machine and all other modules which make up a given machine configuration.

Microinstruction Bus — The Microinstruction Bus (MIB) is the common bi-directional data and control path between all elements of the micromachine. It is through connection to this bus that the WCS module makes its control store accessible to the microprocessor.

Instruction Register — When an LSI-11 machine instruction is fetched from main memory, it is placed in the Instruction Register (RIR) in the Data chip as well as in the translation register in the Control chip.

Microinstruction Register — A microinstruction which has been fetched from either MICROM or user control store is placed in the microinstruction register. This register is

implemented on both the Data and Control chips. The microinstruction register on each chip contains only the portion of the microinstruction relevant to the chip's function.

Translation Register — The Translation Register and the instruction register (RIR) are loaded simultaneously during a machine instruction fetch. The translation register holds the machine instruction for input to the translation array.

Translation Array — The Translation Array is located in the microprocessor Control chip and receives as input either the upper or lower byte of the Translation Register, the microprogram Location Counter and the interrupt register contents. It consists of a large combinatorial logic network which determines the starting location for microcode routines appropriate to fetched machine instructions.

Return Register — The Return Register (RR) is located in the microprocessor Control chip and provides storage for a single microprogram subroutine return address.

Register File — The register file is located in the Data chip and contains 26 8-bit registers which are accessed by a combination of direct and indirect addressing. Adjacent 8-bit registers may be sequentially addressed to form word operands.

1.1.2 LSI-11 Microprogramming Features

1. LSI-11 microprogramming is provided by a Writable Control Store module in conjunction with microprogramming support software.

2. The WCS module allows one half (1024) of the total (2048) control store locations to be user microprogrammed.

3. The microprogramming support software includes a microassembler, writable control store loader and dumper, and a Micro Octal Debugging Tool which allows simultaneous debugging at the machine and micromachine levels.

4. The microprogramming support software also includes the micro code for the EIS/FIS machine instructions. The user may include this microcode in the final writable control store load module, thus combining the existing advantages of the familiar extended and floating point machine instructions with user designed instructions.

5. LSI-11 microprogramming combines the simplicity of vertical microinstructions with the individual bit-control of horizontal microinstructions. The existing bit-control field supported by the micromachine is expanded by two additional bits

with the addition of the WCS module. This new bit-control facility is also timed with respect to the micromachine cycle, thus enabling higher control rates and more accurate control timing than can be achieved with normal LSI-11 system bus I/O control.

6. The WCS module is equipped with a 16-word recirculating microlocation Trace RAM which aids in microprogram debugging. It contains the last 16 microlocations placed on the microinstruction bus. This hardware feature is utilized by MODT to display the last 16 microlocations accessed prior to a user-specified dump point.

1.1.3 Basic LSI-11 Machine-Micromachine Structure

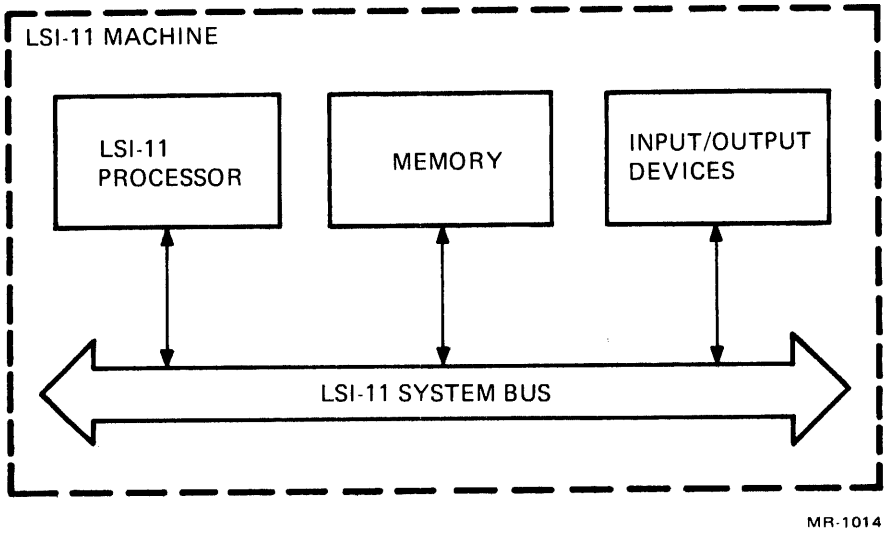
Figure 1-1 contains a simplified illustration of the LSI-11 machine. The LSI-11 machine encompasses the processor, main store or memory, and the input/output devices. These three main components are interconnected by the LSI-11 system bus. The LSI-11 processor is realized by a micromachine which contains a (micro)processor element, a storage element (control store), and external I/O capability. The microprocessor is implemented with two chips (called the Data chip and the Control chip). The Data chip contains the Arithmetic Logic Unit (ALU), the register file, and provides connection to the 16 time multiplexed data/address lines (DAL). The Control chip arbitrates all system bus transactions and determines the microinstruction execution sequence. The structure of the microprocessor chip set is presented in detail in Chapter 3.

The microinstructions are stored in a control store consisting of two or more MICROMs which provide non-alterable, non-volatile memory. The term MICROM is an acronym for MICrocode Read Only Memory. In the basic LSI-11 processor, the MICROM control store contains microcode which performs 2 functions: (1) PDP-11 emulation and (2) console ODT. When included on the processor, The KEV11 EIS/FIS option expands the control store to implement the extended and floating point machine instructions.

The Writable Control Store is connected to the micromachine via a cable/plug assembly (which replaces the KEV11 option in the third MICROM socket). This gives the WCS module access to the microinstruction bus which is the internal bus interconnecting all components of the micromachine. The WCS then may be accessed by the microprocessor in the same fashion as a MICROM. The total number of control store locations addressable by the microprocessor is 2048. The PDP-11 emulation and console ODT microcode require only 1024 locations, or two MICROMs. The WCS module supplies the remaining 1024 control store locations. The WCS memory is loaded and dumped via its LSI-11 system bus interface, which consists of normal control/status and data buffer registers.

LSI-11 Machine-Micromachine Structure

Figure 1-1



1.2 THE BENEFITS OF USER MICROPROGRAMMING

User microprogramming affords greater control over machine operations. The user can create new machine instructions to be used in the same manner as members of the LSI-11, PDP-11/03 machine instruction set. The following paragraphs identify specific areas in which user microprogramming may be beneficial.

1.2.1 Arithmetic Calculations

Many arithmetic calculations are characterized by a concisely-defined algorithm which is often repetitive in nature. The execution of the routine for such an algorithm normally requires many machine instruction fetches and operand address calculations. If the algorithm is suitable for microprogramming it can be implemented in microcode which is then executed in response to a single, user-defined machine instruction. This approach eliminates the multiple machine instruction fetch operations because control remains entirely within the micromachine until the routine is completly executed. A good example of the improvement available in this area is the KEV11 EIS/FIS option. The EIS/FIS option contains microcoded routines for the extended and floating point machine instructions. Upon execution of a single machine instruction, FDIV for example, control is transfered to the appropriate starting point in the FIS microcode. When the routine has terminated, control is returned to the LSI-11 emulation microcode contained in the 2 standard microms. The speed advantage realized by the EIS/FIS microcode ranges from a 3 to 10 times improvement over comparable PDP-11 macroinstruction routines, depending upon the instruction executed and the operand values.

1.2.2 Critically-Timed Input/Output and Control Operations

The rate at which real-time input/output (I/O) operations can be performed depends on three factors: (1) the speed of machine instruction execution, (2) the time delays associated with the LSI-11 system bus and (3) the speed of the device or memory being accessed. The microprogramming facility allows the user to write specialized I/O routines for execution in microcode. In this context, the user employs the "data access" group of microinstructions which are discussed generally in Chapter 4 with detailed explanations in Chapter 5. Sufficient information is provided to allow identification of where these bus delays occur so non I/O microinstructions can be inserted to utilize the delay time.

User-written microcode can make use of a special field of 4 TTL control bits within the microinstruction. Of the 16 possible states encoded in this field, 8 are presently used to control the LSI-11 system bus logic which interfaces between the system bus and the microprocessor chip set. The other 8 states are available for user-defined functions. The 4 bits from this microinstruction field appear on the LSI-11 module fingers as does the third phase of the microcycle clock (PH3 H). These TTL control bits enable the microprogrammer to produce high rate control signals which are timed with respect to the micromachine cycle.

In addition to the 4 control bits (MI<21:18>), the WCS module stores 2 extra bits (MI<23:22>) in each control store location. These two bits are available as signals directly on the LSI-11 system backplane and may be used for any user-defined purpose. The high-order bit (MI<23>) is also used to control the microlocation trace buffer.

1.2.3 Data Manipulation and Relocation

A block move microprogram example is discussed in Chapter 8 in which the arguments required are (1) the starting address of the block to be moved, (2) the starting address of the new block location, and (3) the block length. The execution time saved is proportional to the block length, because machine instruction fetches are eliminated for each word moved.

1.3 SYSTEM IMPLICATIONS OF USER MICROPROGRAMMING

With the microprogramming facility, all the resources used to emulate the LSI-11 architecture and operation are at the user's disposal. The only resource that cannot be accessed is the Control chip translation array which is specifically configured to implement the opcodes of the standard and extended machine instruction set. However, the microprocessor instruction set does provide a means for accomplishing translations in only a few additional microcycles.

1.3.1 Control Flow Integrity

The first responsibility of the microprogrammer is to maintain the integrity of the control flow which implements the machine operating cycle. Control is transferred to user microcode when the microprocessor control chip, via the translation array, determines that a the fetched instruction is a user-defined machine opcode. User microcode then maintains control until the microroutine has executed, whereupon control must be returned to the trap interrupt service routine, thus maintaining the normal Fetch-Execute-Interrupt machine cycle.

An additional requirement arises when the user-microprogrammed routine performs I/O operations. In executing an I/O transfer the LSI-11 system bus transaction requires that the addressed I/O device return a reply signal to the processor, acknowledging its role in the transfer. If no reply is received by the CPU within 10 microseconds, the processor executes a bus error trap through LSI-11 memory location 4. Because a bus error can occur in a number of contexts, the microprogrammer must prepare for the proper response by setting an internal flag (see Section 8.6.4.1).

1.3.2 Interrupt Response Latency

Interrupts are recognized by the LSI-11 processor only during the final phase of the normal machine operating cycle (Fetch-Execute-Interrupt). The delay in acknowledging a pending in-

terrupt is directly related to the length of time of the Execute phase. The microprogrammer is provided with a means of testing for external interrupts and the Event Line interrupt during user-microprogram execution. If such an interrupt is pending, control can be transferred to a user microcode routine. This routine could save the interrupted machine state, decrement the PC and then return control to the last part of the machine cycle.

Once the normal LSI-11 interrupt service has been completed, the user-defined machine instruction is fetched again and the user microprogram can then execute to completion. The external interrupt test facility allows potentially lengthy microcoded routines to operate in a time critical environment. Interrupt testing is unnecessary when the microcode to be executed is of known short duration. See Section 8.6.7.

1.3.3 Register Content Security

The microprogrammer has access to the LSI-11 processor registers as well as to the internal registers in the micromachine. Several of the internal registers have predefined uses (e.g., PSW, bus error flags) and should only be modified in accordance with those uses. Manipulation of the standard processor registers (R0-R7) should only occur as part of the intended function of the user-defined machine instruction.

1.3.4 Processor Status Word Updating

The processor status word (PSW) in the LSI-11 is a composite of (1) the 4 PDP-11 Status Flags (N,Z,V,C) and (2) the Trace Bit and (3) the Interrupt Enable bit. Internally, the PDP-11 Status Flags are explicitly accessed by 2 microinstructions (LCF, CCF) and implicitly altered by executing a microinstruction capable of affecting these flags (e.g., AWF). The Trace Bit and Interrupt Enable Bit are altered by executing a Set Interrupt (SI) or Reset Interrupt (RI) for the microprocessor control flags I4 and I5 respectively. Since these flags cannot be read, a copy of these flags is kept in an internal register (RPSWH) in the bit positions that correspond to the Trace bit (bit <4>) and the Interrupt Enable bit (bit <7>) of the LSI-11 PSW.

1.3.5 Dedicated Control Store Locations

The 1024 Writable Control Store locations are normally configured at microaddresses 2000 through 3777 (octal). The WCS module therefore replaces the EIS/FIS MICROM which responds to addresses 2000-2777. The LSI-11 emulation microcode, in conjunction with the translation array, performs a partial decode of the extended and floating point machine instructions and transfers control to control store locations in the 2000-2777 range. The microprogrammer has the responsibility of handling such a transfer as a reserved instruction trap.

1.3.6 Machine Instruction Support

A newly defined machine instruction must be explicitly documented as to function, execution characteristics, and operand requirements. It should also be assigned an assembly mnemonic which is supported by a macro definition to enable the instruction to be programmed and assembled along with the standard instruction set (See Section 8.2).

1.4 MICROPROGRAM CHARACTERISTICS

1.4.1 Vertical and Horizontal Microinstructions

The user will discover that microprogramming the LSI-11 micromachine requires techniques nearly identical to those used in LSI-11 assembly language programming. This strong similarity arises because the micromachine executes vertical as opposed to horizontal microcode. One characteristic of vertical microinstructions is that microinstructions are executed out of sequential locations in control store, just as machine instruction are executed sequentially out of main or program store.

Another characteristic shared between vertical microinstructions and machine instructions is that both perform recognized complete operations upon execution. For example, the COMPLEMENT BYTE microinstruction reads the contents of the specified source operand register into the ALU, forms the complement and places the result in the specified destination register. A horizontal microinstruction, by contrast, would affect (micro) processor control at a much more detailed level, often with direct control of register read and write circuitry, data path bus drivers, ALU operating modes, and so forth. In addition, the horizontal microcode usually contains a field which holds the address of the next microinstruction to be executed.

1.4.2 Logic Control Features

The LSI-11 microprogramming option offers a repetoire of 149 microinstructions which support both byte (8-bit) and word (16-bit) operations. Further, the basic 16-bit vertical microinstruction word is extended by 6 control bits within the micromachine and by an additional 2 control bits in the WCS module. The 6-bit extension contains 4 bits which may be encoded for direct control of user logic. The 2-bits supplied by the WCS module enhance this logic control capability. See Section 6.7.5.

1.5 THE MICROPROGRAMMING ENVIRONMENT

1.5.1 Creating the Source File

The RT-11 Operating System environment in which the LSI-11 microprogrammer works is familiar to the experienced PDP-11 assembly language programmer. A fundamental reference is the microinstruction set description, which is contained in Chapter 4 and Appendix A of this manual. Chapter 7 describes the WCS Software Tools available.

The source file for the microprogram is created with either text editor, EDIT or TECO. The recommended extension in the file specification is .MIC signifiying a microprogram source file.

1.5.2 The Microassembler

The Microassembler is described in detail in Chapter 7. It consists of a stand alone 2 pass assembler for the LSI-11 microinstruction mnemonics with several assembler directives available. The listing shows the assembled octal for each microlocation and includes a bitmap of the utilized microaddresses. The microassembler output (.OBJ extension) is then loaded into the WCS module as described in the next section.

1.5.3 Writable Control Store Loader

The microprogramming support software provides a WCS loader program which can initially clear the WCS RAM memory and then load from one to six specified .OBJ files produced by the Microassembler. The WCS loader is described in Chapter 7.

1.5.4 Micro Octal Debugging Tool (MODT)

Microprogram debugging is done using the Microprogram Octal Debugging Tool program (MODT). This program expands the familiar PDP-11 ODT so that user-written microprograms can be examined, modified, and executed. Since all microprograms are executed in response to machine instructions, MODT allows the user to write, modify and execute short programs. Breakpoints may be set in the machine instruction flow to examine program progress at the machine level. At the micromachine level, the user may establish "dump points" which display the contents of all internal registers at a selected point. Once the dump results are analyzed, the dump points can be moved and subsequent executions allow the user to examine additional portions of microcode. This dump-examine process continues until the microprogram is completely debugged and verified. MODT is described in detail in Chapter 7.

1.5.5 Microprogram Trace Facility

The microaddress trace hardware on the WCS module is normally controlled and examined by the Micro Octal Debugging Tool (MODT) program. It consists of a 16-word buffer which stores a sequence of 16 microinstruction locations placed on the microinstruction bus (MIB) during the execution of user-written microcode. The last location to be stored is designated by the microprogrammer (using MODT). Once the 16 addresses have been stored, MODT may display the contents of the addressed locations in both octal and symbolic forms. See Chapter 7 for a complete description.

1.6 REFERENCES

The following documents provide background information for this manual:

1. RT-11 System User's Guide, DEC-11-ORGDA-A-D
2. PDP-11 TECO User's Guide, DEC-11-UTECA-A-D
3. The Microcomputer Handbook, EB-07948-53
4. KD11-H processor schematic diagram, D-CS-M7264-0-1 (revision Y or greater)
5. WCS schematic diagram, D-CS-M8018-0-1

Additional copies of all items above can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attn: Communications Services (NR2/M15)
Customer Services Sections

CHAPTER 2

THE LSI-11 MACHINE STRUCTURE

2.1 GENERAL

This discussion of the LSI-11 machine structure covers both the architecture and operation of the LSI-11. Architecture relates to the system resources and their configuration. Operation indicates how data and address information is moved between and manipulated within the system resources.

This chapter is a subset of the Microcomputer Handbook, which should be consulted for additional detail, specifically in the areas of LSI-11 options and hardware. This chapter emphasizes selected topics which are essential to the understanding of the microprocessor contained within the LSI-11 processor module. Chapter 3 enhances the information to include the additional information required to microprogram the LSI-11 micromachine.

2.2 MACHINE ARCHITECTURE

The LSI-11 machine consists of three general component areas connected by a common system bus. Any specific machine configuration may be accurately represented by using specialized components in these three areas:

1. The LSI-11 Processor
2. The Memory
3. The Input/Output Devices

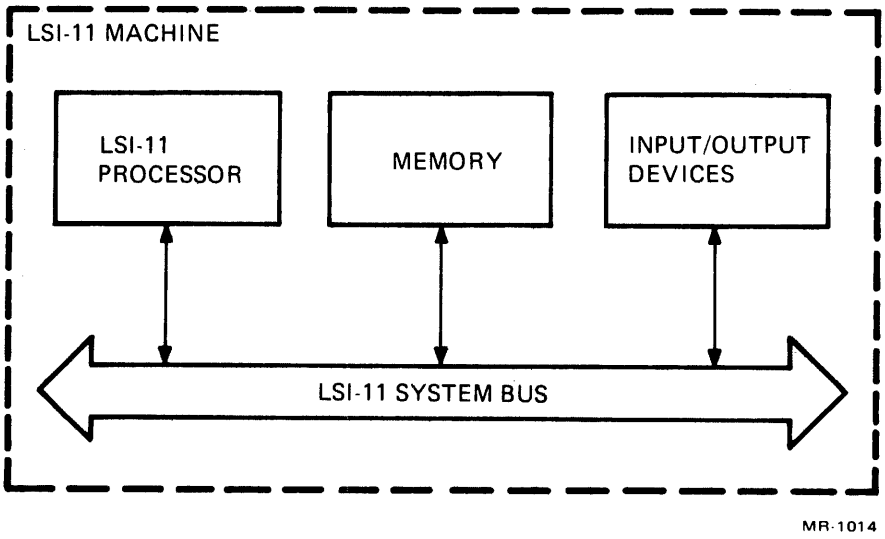
These components and their common, bidirectional access to the system bus are illustrated in Figure 2-1.

2.2.1 System Bus

The system bus is characterized by the number of memory or device locations addressable, the type of information transfers supported, and by the auxiliary system control signals it contains.

LSI-11 Machine Structure

Figure 2-1



2.2.1.1 System Bus Address Space - The virtual (and physical) addressing capability of the system bus is determined by the 16-bit width of the binary addressing word. The system bus supports both 8 bit byte and 16 bit word addressing. Figure 2-2 illustrates the address space with which the LSI-11 machine operates. The bottom 28K (28,672) word addresses constitute the memory space. The top 4K (4096) word addresses are normally dedicated to the input/output devices. The LSI-11 processor does not occupy any address locations.

2.2.1.2 System Bus Data Transfer - All data transfer operations supported by the system bus are under the control of the LSI-11 processor. A portion of the processor is dedicated to the control of the system bus and administrates the transfer of data and address information between the machine components. There are three types of data transfers possible within the LSI-11 machine:

1. Programmed Input/Output Transactions (Programmed I/O)

2. Interrupt-Driven Input/Output Transactions (Interrupt I/O)

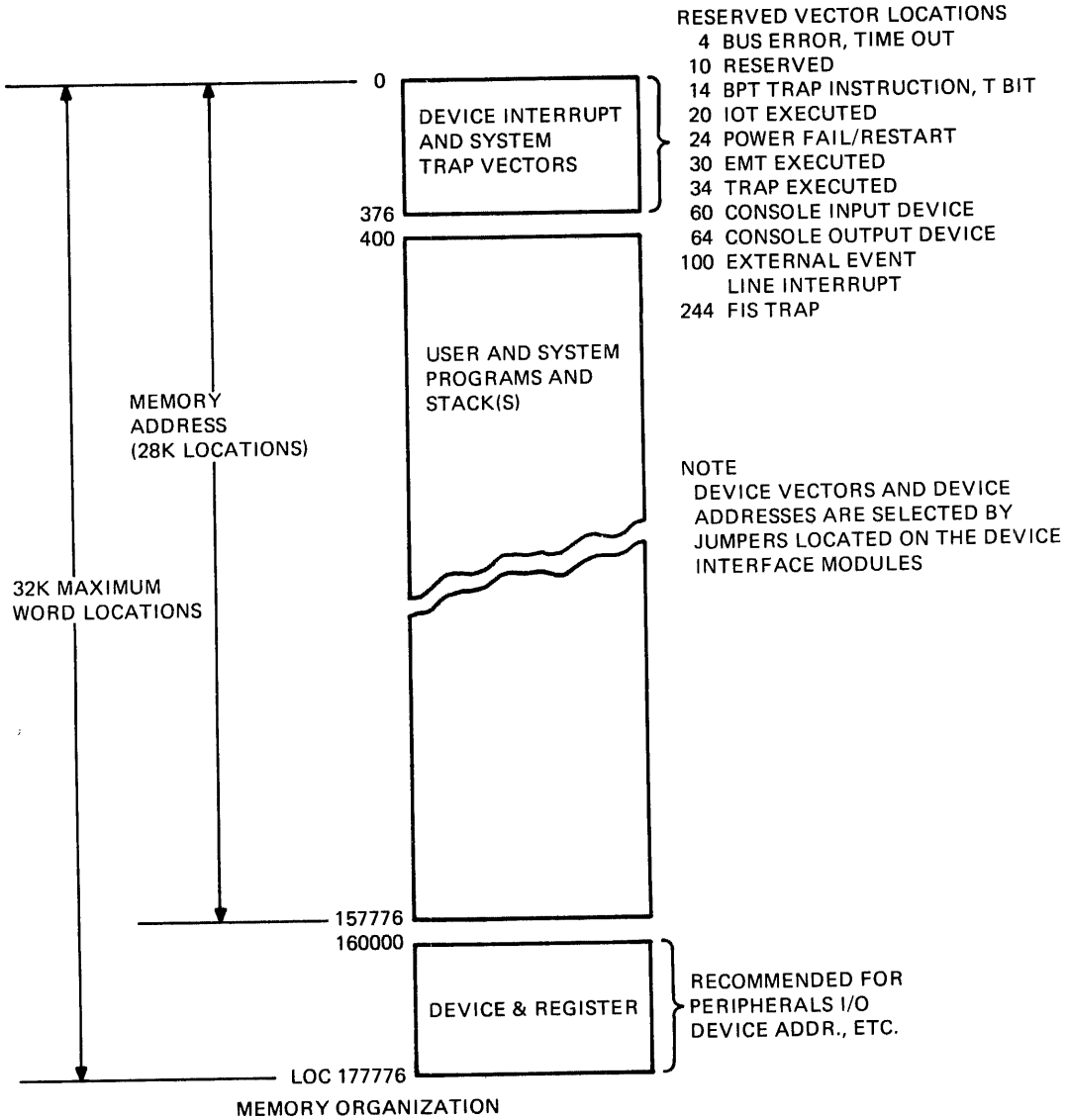
3. Direct Memory Access Input/Output Transactions (DMA I/O)

Programmed I/O occurs in response to programmed machine instructions. An example of Programmed I/O is the execution of a MOV instruction where at least one of the source or destination addresses is in memory or in a device register. If both source and destination operands are in the processor registers, no bus I/O transfer is required.

The simplest example of a Programmed I/O bus transaction is the DATI (Data-In) bus operation. This operation illustrates how the 16 Data/Address Lines (BDAL L <15:00>) are time-multiplexed between address and data information. The first event of the DATI cycle is placing the memory or device address on BDAL L <15:00>. After the address information settles (or becomes valid), the BSYNC L signal is asserted. This signal causes each memory and I/O DEVICE module connected to the bus to check whether the address corresponds to its own address(es). Recognition by the addressed device is represented by a signal internal to that device which is latched (or stored) by the assertion of BSYNC L for the duration of the bus operation. Storing this signal constitutes the second event. The third event of the DATI cycle removes the address information from BDAL L<15:00> and asserts BDIN L. This signal informs the memory or selected device that a DATA-IN cycle is to be performed. The fourth event is in response to BDIN L; the selected device places its data on BDAL L<15:00> and returns BRPLY L to the processor. The assertion of BRPLY L is the first indication to the processor that the addressed component exists and is putting data on the bus. Upon receipt of BRPLY L, the processor accepts the information on BDAL L<15:00> as valid and stores it internally. If BRPLY L is not received within 10 microseconds after BDIN L was asserted, a processor trap occurs to memory lcoation 10. By this means, the processor avoids waiting for a reply from a memory location or device which does not exist or is malfunctioning. The bus cycle fifth event is the accepting and storing of the input data and the subsequent negation of BDIN L. In the sixth event, the selected device responds to the negation of BDIN L by terminating BRPLY L. In the seventh and final event, the processor terminates the bus cycle by negating BSYNC L.

LSI-11 Machine Address Space

Figure 2-2



NOTE
THERE IS 32K OF USERS MEMORY SPACE AVAILABLE; HOWEVER 0–28K IS RECOMMENDED FOR MEMORY ADDRESS LOCATIONS, AND 28K–32K FOR PERIPHERALS I/O DEVICE ADDRESSES, ETC.

MR-1202

The DATI bus cycle is described in detail in the Microcomputer Handbook along with the DATO (Data-Out), DATOB, DATIO (Data Input-Output) and DATIOB bus cycles. The "IO" (Input-Output) bus cycles enable the processor to execute a Read-Modify-Write operation. This allows data to be retrieved from an addressed location, manipulated within the processor, and re-deposited in the same location while asserting only one bus address. The "B" suffix on the DATOB and DATIOB bus cycles indicates that the output portion of the bus cycle is a byte rather than a word transfer.

Interrupt I/O is initiated by an Input/Output device. The interrupting device requests service from the LSI-11 processor by asserting the BIRQ L signal on the system bus. When the processor acknowledges a device request by asserting the BIACK H signal, it causes the interrupting device that is electrically closest to the processor on the LSI-11 bus to return a vector address. The vector address tells the processor where the address of the device service program is located in memory. Once execution of the service program has been completed, the LSI-11 processor resumes execution of the interrupted program.

Priority among multiple external devices having Interrupt I/O capability is established via electrical position relative to the processor. The device closer to the processor posesses the higher priority. The interrupt grant chain is broken by the device being serviced, thus making further interrupts from lower priority devices impossible. Further interrupts from higher priority (electrically closer) devices may still be acknowledged and serviced. Additional details regarding Interrupt I/O transactions are available in the Microcomputer Handbook.

Although the LSI-11 processor remains master of the system bus during both Programmed I/O and Interrupt I/O, DMA I/O requires that bus mastership be granted to an Input/Output device for high speed data transfer to or from memory. During a DMA I/O transaction, the Programmed I/O and Interrupt I/O operations of the LSI-11 processor are suspended. The device functioning as bus master can now perform any of the possible bus cycles (DATI, DATO, DATOB, DATIO, DATIOB). When the DMA transfer is completed, bus mastership is relinquished to the processor. Note that during DMA I/O, the CPU module waits to perform Programmed I/O or Interrupt I/O only. Any other processor activity (such as operations between registers) can continue.

2.2.1.3 System Bus Control Signals - In addition to the bus signals required to support Programmed I/O, Interrupt I/O, and DMA I/O several auxilliary control signals are contained within the bus system.
System Initialization Signal

The common system initialization signal is BINIT L. It is asserted by the processor whenever BDCOK H is passive and whenever the BINIT command is issued by the processor. Examples of the latter are during the Power-up routine and in execution of the RESET machine instruction. All peripheral devices should use the BINIT L signal to initializae and clear internal flip-flops and registers.

Power Supply Signals

Two signals which indicate the status of the system power supply are

part of the bus system. These signals are generated by the power supply itself and the processor monitors them to take appropriate action during Power-Up and Power-Fail sequences. At the beginning of the Power-Up sequence, BDCOK H and BPOK H are both passive. BDCOK H is asserted first, whereupon the processor initializes itself and all system components and waits for BPOK H to be asserted. When BPOK H is asserted, the processor executes the jumper selected Power-Up routine. During a Power-Down or Power-Fail sequence, BPOK H is negated first, causing a Power-Fail trap. The processor then executes the program located at the trap vector address (24). When BDCOK H goes passive, the processor asserts BINIT L. It should be noted that a proper Power-Fail sequence will negate BDCOK H before restoring BPOK H. BDCOK H must go passive to re-initiate the Power-Up sequence.

Processor Control Signal

The only signal in the bus system directly controlled by the operator of the processor is BHALT L. This signal is asserted by a front panel (PDP-11/03) switch or alternately by the BREAK key on the console terminal. Pressing the BREAK key causes a framing error which asserts BHALT L via the console serial line interface. When the front panel Run/Halt switch is used to halt the processor, it must be reset to RUN before the processor can proceed. However, pressing the console BREAK key asserts BHALT L only as long as the key is depressed.

Processor Monitor Signal

The ability to monitor the Run/Halt state of the processor is provided by the SRUN signal. The SRUN signal is asserted once each time the processor performs a Machine Instruction Fetch. This signal is the input to a circuit on the PDP-11/03 console that drives a RUN indicator light.

Memory Refresh Control

All dynamic MOS memory modules which do not have self-contained refresh capability must be refreshed via the system bus. A signal called BREF L is asserted during the addressing portion of the BSYNC/BDIN transaction to differentiate between memory refresh and the standard DATI bus cycle.

2.2.2 Memory

The memory component of Figure 2.1 may be implemented with either semiconductor or magnetic devices. Each memory device has operating characteristics which determine how it is to be used in a particular system.

Most memory devices function as Read/Write memory and may be accessed by either the processor or a DMA I/O device.

2.2.2.1 Semiconductor Memory - Semiconductor memory may be classified as either dynamic or static memory. The most familiar example of dynamic semiconducor memory is the 4K MOS Read/Write memory located on the KD11-F LSI-11 processor module. Dynamic memory must be periodically refreshed in order for the memory to retain its contents.

Static semiconductor memory is also available as Read/Write memory and doese refreshing. An example of this type is the 256 16-bit words of RAM found on the MRV11-BA UVPROM module. An additional static semiconductor memory type is Read Only Memory. Read Only Memory types are found on the MRV11-AA PROM module and on the MRV11-BA UVPROM module. Both the PROM and UVPROM are non-volatile types; they retain their contents even after power has been removed. The MRV11-AA PROM module contains fusible link semiconductor memory devices whose contents are established by special programming equipment. Once programmed, the fusible link PROM contents can not be altered. The MRV11-BA UVPROM is also programmed by special equipment. but its contents can subsequently be erased by exposure to UV light, effectively clearing the contents in preparation for re-programming with new data.

2.2.2.2 Dynamic Memory Refreshing - Three techniques are available to satisfy the requirement for dynamic memory refreshing. These three techniques are:

1. Processor Controlled Refresh
2. Direct Memory Access Refresh
3. Distributed Refresh

Processor Controlled Refresh is a feature available on the M7264 LSI-11 processor module. This refresh mode is normally disabled, but can be enabled by removing the appropriate jumper on the module. When enabled, a 600 Hz oscillator causes an internal interrupt. This interrupt initiates the execution of a microprogrammed routine which refreshes all dynamic memory devices used in the system. The refresh routine performs 64 BSYNC/BDIN cycles with BREF L asserted, occupying the system bus for about 130 microseconds. When processor-controlled refresh is employed, all dynamic memory mocules should have their Reply During Refresh options disabled, except for the memory module located farthest from the processor. This assures that the longest bus delays will be compensated for during execution of the refresh microprogram. Note that this form of refresh is not recommended for systems utilizing lengthy microprograms.

Direct Memory Access Refresh is performed by a DMA device and relieves the processor of refresh responsibility. Examples of DMA refresh are the REV11-A and REV11-C modules. The DMA refresh technique differs from the processor-controlled method in that the memory modules are refreshed one row at a time instead of all rows at once. Therefore the DMA device controls the system bus for purposes of refresh for only 1.3 microseconds at a time, thus eliminating the 130 microsecond dead time inherent in the former method. Any user-designed DMA device may also perform refreshing as long as proper sequencing and timing assure that each row of memory is refreshed at least once each 2 milliseconds or less.

Distributed Refresh is the technique used by the MSV11-CD memory module. This module is equipped with timing and sequencing circuitry which performs refreshing for its own memory devices. It refreshes one row at a time every 25 microseconds and isolates itself from the system during each row refresh. If the LSI-11 processor, or DMA device, requires a memory access when the module is refreshing a row, the memory module merely delays its response by returning the BRPLY L signal after the row refresh is completed. The relatively short, occasional delays that occur with this technique are compatible with the essentially asynchronous system bus data transactions.

The memory component of a specific LSI-11 machine may be composed of more than one memory type to satisfy user requirements.

2.2.2.3 Magnetic Memory - Magnetic Memory is non-volatile, so it does not require refreshing. Therefore, memory contents stored in magnetic core are not lost during a power failure. Magnetic memory can maintain the information of a partially-executed program or routine when power is removed so that the routine may continue to completion when power is restored. A Power-Fail routine (which is initiated by the processor Power-Fail trap) must save all volatile machine states (e.g., the General Purpose Registers) in the magnetic portion of memory before power goes down (BDCOK H goes passive).

2.2.3 Input/Output Devices

The Input/Output devices connected to the LSI-11 system bus provide a means for interfacing control and data information between the LSI-11 machine and the outside world. An example of an I/O device which bi-directionally passes both data and control is the DLV11 serial line unit which connects to the console terminal.

2.2.3.1 Device Address Format - All Input/Output device locations on the LSI-11 system bus are accessed in the same manner as memory. Normally, each I/O device has four sequentially-numbered word locations associated with it. These four locations provide for both control and data transfer between the processor and I/O device according to the following convention:

XXXXX0	Receive Control Status Register	(RCSR)
XXXXX2	Receive Buffer	(RBUF)
XXXXX4	Transmit Control Status Register	(XCSR)
XXXXX6	Transmit Buffer	(XBUF)

The Receive Buffer (RBUF) holds data which has been received from the I/O device and which can be transferred to the processor or to memory. The Receive Control Status Register (RCSR) contains control flags related to the device receive function. The Transmit Buffer (XBUF) holds data which has been transferred to the I/O device for presentation to the outside world. The Transmit Control Status Register (XCSR) contains control flags related to the device transmit function.

2.2.3.2 Enabling Device Interrupts - Input/Output devices which support interrupt-driven I/O transactions are equipped with an interrupt-enabling mechanism which must be explicitly set by the processor before the device can initiate an interrupt. The interrupt-enabling mechanism is reset or disabled at Power-Up time by the system initialization signal BINIT L.

2.2.3.3 DMA Transfer Restrictions - A Direct Memory Access (DMA) data transfer is accomplished by the DMA device becoming master of the system bus (which suspends LSI-11 processor I/O operations). If the processor is responsible for dynamic refresh, only single byte or single word transfers are allowed to give the processor opportunity to execute a memory refresh routine. A long burst of DMA transfer could cause the refresh period to delay beyond the 2 millisecond maximum allowable time. If Processor-Controlled Refresh is not utilized, restrictions on DMA transfer are eliminated.

2.2.4 The LSI-11 Processor

The LSI-11 Processor module which appears in Figure 2.1 is presented with greater internal detail in Figure 2.3. The three functional areas are:

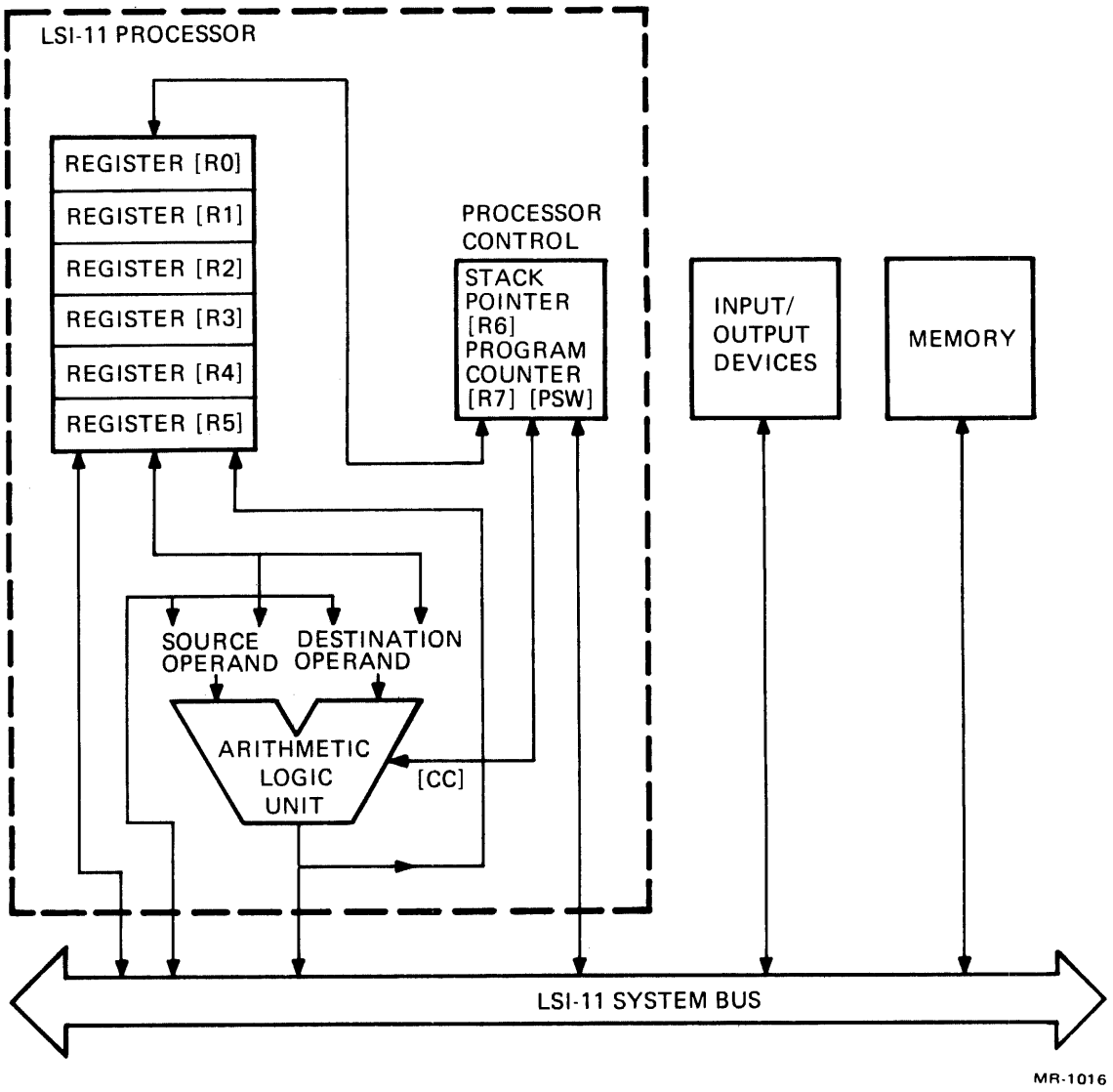
1. The Arithmetic Logic Unit
2. The general Purpose Registers
3. The Processor Control

2.2.4.1 Arithmetic Logic Unit - The Arithmetic Logic Unit (ALU) performs the operations required for the machine instruction set. Arithmeic operations employ two's complement number representation in fixed point format. With the addition of the KEV11 EIS/FIS MICROM (Extended/Floating Instruction Set), floating point arithmetic operations are also performed. Arithmetic and logical operations are executed on both byte and word data. The results of ALU operations are monitored by four Condition Code Flags (N,Z,V,C) which are part of the Processor Status Word (PSW) register. The source and destination operands which constitute the inputs to the ALU may be located in General Purpose Registers, Memory, or Input/Output device registers.

2.2.4.2 General Purpose Registers - The General Purpose Registers are located within the Processor and thus their contents are accessed without the use of a system bus operation. The registers may contain data or address information. Registers R6 and R7 are dedicated to Stack Pointer (SP) and Program Counter (PC) use, respectively, and are therefore associated with Processor Control. Both byte and word addressing is supported for registers R0 through R5. Because of their dedicated application, registers R6 and R7 allow word addressing only.

LSI-11 Processor Detail

Figure 2-3



2.2.4.3 Processor Control - There are four main areas into which all functions peformed by the Processor Control may be divided. These four areas are:

1. Processor Control (Overall Control)
2. Machine Instruction Execution
3. Address Generation
4. System Bus Control

These four areas and their associated processor resources are illustrated in Figure 2-4. Overall Processor Control is determined by (1) the status of the system power supply and (2) the operator.

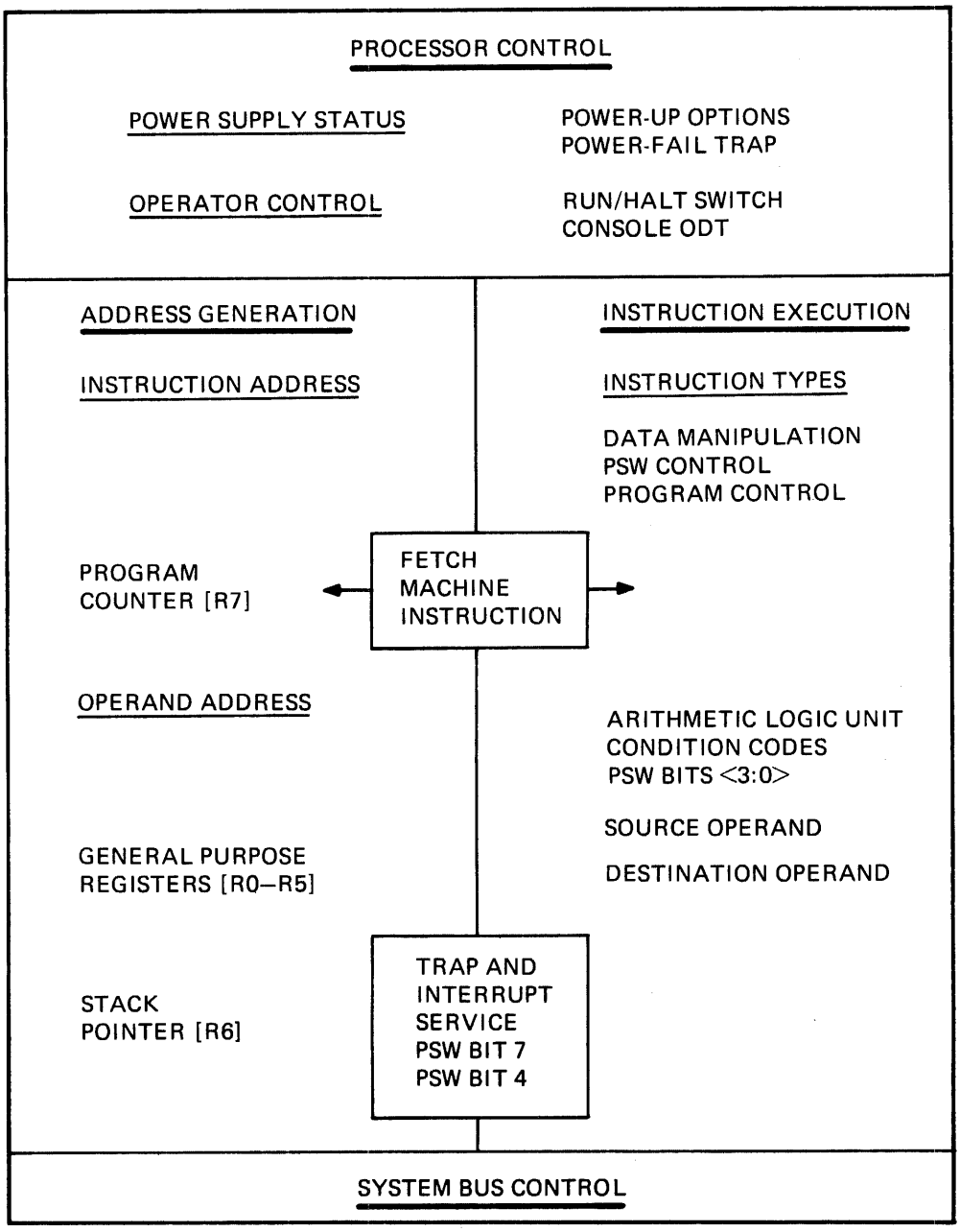
In the PDP-11/03, the processor is informed of power supply status by means of the BDCOK H and BPOK H signals (see Section 2.2.1.3). The processor will cause either a Power-Up or a Power-Fail operation to be performed in response to the status of these signals. BDCOK H and BPOK H originate in the H780 power supply and their exact sequence and timing details are contained in The Microcomputer Handbook. In general, BPOK H is the last signal to be asserted in a Power-Up sequence and the first signal to go passive in a Power-Fail sequence. When BDCOK H is passive, indicating the lowest state of the system power supply, it asserts BINIT L and forces the microprocessor Control chip to the Reset state. When the processor is in the Run state, the change of BPOK H from active to passive will cause a Power-Fail trap to be performed. The Power-Up mode is determined via jumper configuration on the processor module. The means by which the Processor interprets the jumper configuration and the power supply status signals are further detailed in the control flow diagram of Figure 2.16-2 in the Machine Operation section (Section 2.3).

Operator Control over the LSI-11 processor is achieved through two means: (1) the state of the front panel Run/Halt switch and (2) Console ODT. Placing the Run/Halt switch in the Halt position causes a Halt interrupt which passes control to the microprogrammed ODT routine. Once the processor has entered the Console ODT/Halt state, the Run state may be reentered by operator execution of the "P" or "G" commands. When the Run/Halt switch is in the Halt position and the "P" command is repeatedly issued, single-step program execution is achieved. A complete description of Console ODT is in The Microcomputer Handbook.

As shown in the Figure 2.4, the contents of the LSI-11 Processor Status Word (PSW) register have been divided up and allotted to two areas. The four ALU condition code flags appear in the Machine Instruction execute sub-area and the Trace Trap and External Interrupt Enable flags appear in the Trap & Interrupt Service sub-area. The complete PSW, which the operator may access by either the "RS" ("$S") Console ODT command or under program control via the MFPS or MTPS machine instruction, is illustrated in Figure 2.5. The four lower flags are conditionally set as a result of any processor operation which manipulates data in the ALU or moves data within the LSI-11 machine.

Processor Control Functions

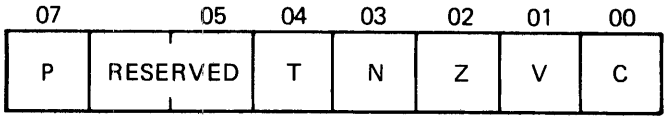
Figure 2-4



MR-1017

Processor Status Word (PSW)

Figure 2-5



P: EXTERNAL INTERRUPT ENABLE

T: TRACE TRAP ENABLE

N: NEGATIVE CONDITION CODE

Z: ZERO CONDITION CODE

V: OVERFLOW CONDITION CODE

C: CARRY CONDITION CODE

MR-1018

Data moved between a memory location and a device register will affect the condition codes as will the execution of an arithmetic or logical operation. The specific condition code functions for each machine instruction are found in The Microcomputer Handbook. The Machine Instruction Execute area performs the operations dictated by the fetched machine instruction. All members of the LSI-11 Machine Instruction Set may be classified in the three following groups:

1. Data Manipulation Instructions
2. Program Control Instructions
3. Processor Status Word Control Instructions

Data Manipulation instructions include all single and double operand instructions with the exception of the PSW operators MFPS and MTPS. All instructions in this group set or reset the ALU condition codes as a result of the operation performed. None of the instructions in this group can change the processor priority, PSW BIT 7, or the trace trap enable, PSW BIT 4.

These instructions are listed below.

Single Operand

General: CLR(B), COM(B), INC(B), DEC(B), NEG(B), TST(B)
Shift & Rotate: ASR(B), ASL(B), ROR(B), ROL(B), SWAB
Multiple Precision: ADC(B), SBC(B), SXT

Double Operand

General: MOV(B), CMP(B), ADD, SUB
Logical: BIT(B), BIC(B), BIS(B), XOR

The KEV11 EIS/FIS (Extended/Floating Instruction Set) adds four fixed point and four floating point instructions to the group.

Extended Fixed Point: MUL, DIV, ASH, ASCH

Floating Point: FADD, FSUB, FMUL, FDIV

The Program Control Instructions are divided into two sub-groups, depending on whether the PSW contents are affected. The execution by the processor of any instruction in the first sub-group has no effect on the PSW contents. This sub-group includes all Branch, Jump & Subroutine, and Miscellaneous instructions.

Branch: BR, BNE, BEQ, BPL, BMI, BVC, BVS, BCC, BCS

Signed Conditional Branch: BGE, BLT, BGT, BLE

Unsigned Conditional Branch: BHI, BLOS, BHIS, BLO

Jump & Subroutine: JMP, JSR, RTS

Miscellaneous: HALT, WAIT, RESET, SOB

The second sub-group of Program Control instructions is executed in the Trap & Interrupt Service area shown in Figure 2.4. These instructions can control every working bit in the PSW by moving a byte to the PSW register from a vector location or from the stack.

Trap & Interrupt: EMT, TRAP, BPT, IOT, RTI, RTT

Processor Status Word Control Instructions exert direct control over the PSW register contents. The condition code operators may be used to set or clear any combination of the condition code flags. The NOP instruction is also included here.

Condition Code Operators

Clear:	CLC, CLV, CLZ, CLN, CCC
Set:	SET, SEV, SEZ, SEN, SCC
	NOP

Two single operand instructions belong to this group because their execution affects the PSW register contents.

Processor Status Word Operators

MFPS
MTPS

The MFPS (Move byte From Processor Status word) instruction transfers the PSW register contents to the destination contained in the instruction. If the destination is mode 0, PSW BIT 7 is sign extended through the upper byte of the register. However, the movement of the PSW contents through the processor to the destination will modify the information in the PSW register according to the following rules.

N = Set if PSW Bit <7> = 1; cleared otherwise
Z = Set if PSW<7:0> = 0; cleared otherwise
V = cleared
C = not affected

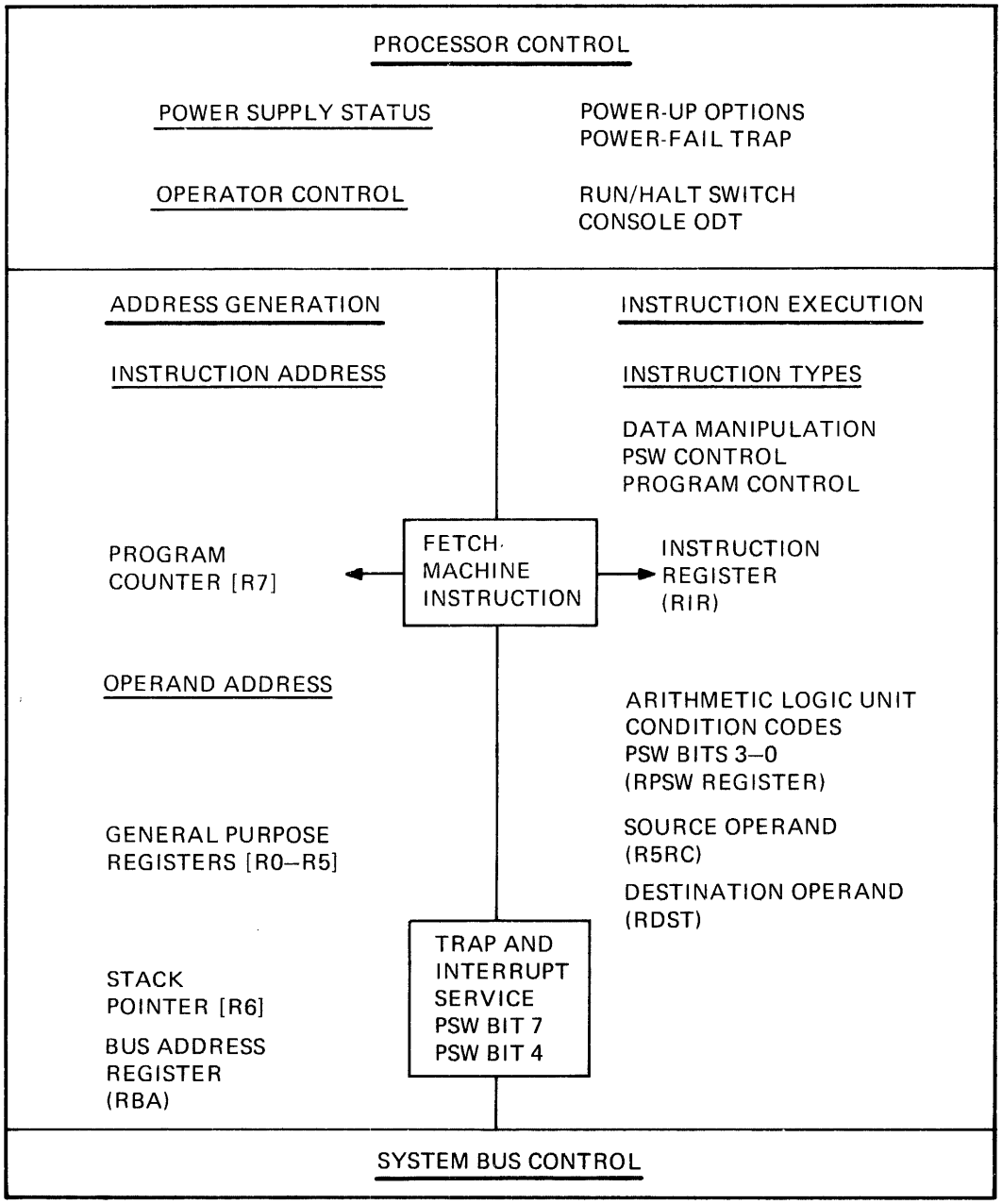
The MTPS (Move byte To Processor Status word) instruction transfers the 8 bits of the source operand to the PSW register. All working bits may be set or cleared, except the Trace Trap Enable (PSW Bit <4>) which may only be cleared.

The LSI-11 machine instruction set contains four additional reserved instruction groups which have no assigned mnemonic: 21R, 220-227, 075040-075777, 076000-076777.

The 21R opcode (where R is a 0 - 7) causes the contents of the internal temporary registers to be transferred to consecutive locations pointed to by the contents of register R. Register R is not restored at the end of execution. The internal registers accessed by this instruction are illustrated in Figure 2.6, which also shows their relationship to the Processor Control Functions. This instruction is used for diagnostic purposes only and belongs to the Data Manipluation group listed above.

Processor Control Functions (With Internal Registers)

Figure 2-6



MR-1019

Machine-level execution of instructions in the range of 220-227 causes control to be transferred to the microinstruction located at microaddress 3000 (in user control store). If control store is not present (or if it is disabled) a reserved instruction trap through memory location 10 will occur. (See the Microcomputer Handbook for a description of illegal instruction traps).

Instructions in the range 075040 through 075777 cause control to be transferred to the microinstruction located at microaddress 3003 (in user control store). If control store is not present (or if it is disabled), a reserved instruction trap through memory location 10 will occur.

The availability of Writable Control Store enables the user to design unique Machine Instructions. These instructions are based on the 0767XX operation code assignment, as shown in Figure 2.7. The execution of this type of instruction causes control to be transferred to microaddress 3001 where the microprocessor begins execution of the user-microprogrammed routine. Note that all instructions of the 076XXX format transfer control to the same microlocation (3001), but only opcodes in the range 076700 to 076777 may be utilized for user instructions. The lower six bit positions may then be employed by the user to differentiate between user instructions or to carry data to the microprocessor.

The Address Generation area serves both the Instruction Address and Operand Address generation functions. Instruction addressing employs dedicated register R7 as the Program Counter (PC) and increments the counter by the number of word addresses required by the machine instruction currently under execution. Instruction addressing may also be modified by Program Control instructions, Trap & Interrupt Service, Power-Up routines, or by operator intervention through Console ODT.

Operand Address generation supports the eight General Purpose Register addressing modes and the four Program Counter addressing modes for determing the source and destination operands. Register R6, dedicated to Stack Pointer use, is employed by the Operand Address generation function, Trap & Interrupt Service operations, and by the Jump & Subroutine machine instructions in the Program Control group.

2.2.5 The LSI-11 Writable Control Store

The User-Microprogrammed machine instruction format (illustrated in Figure 2.7) transfers control to the user control store area (Writable Control Store). The user control store area is composed of random access Read/Write semiconductor memory which contains the user-programmed microinstructions to be accessed by the LSI-11 microprocessor. The interconnection between the LSI-11 processor module and the Writable Control Store (WCS) module is shown in Figure 2.8.

User-Microprogrammed Machine Instruction Format

Figure 2-7

USER
(USER-MICROPROGRAMMED MACHINE INSTRUCTION)

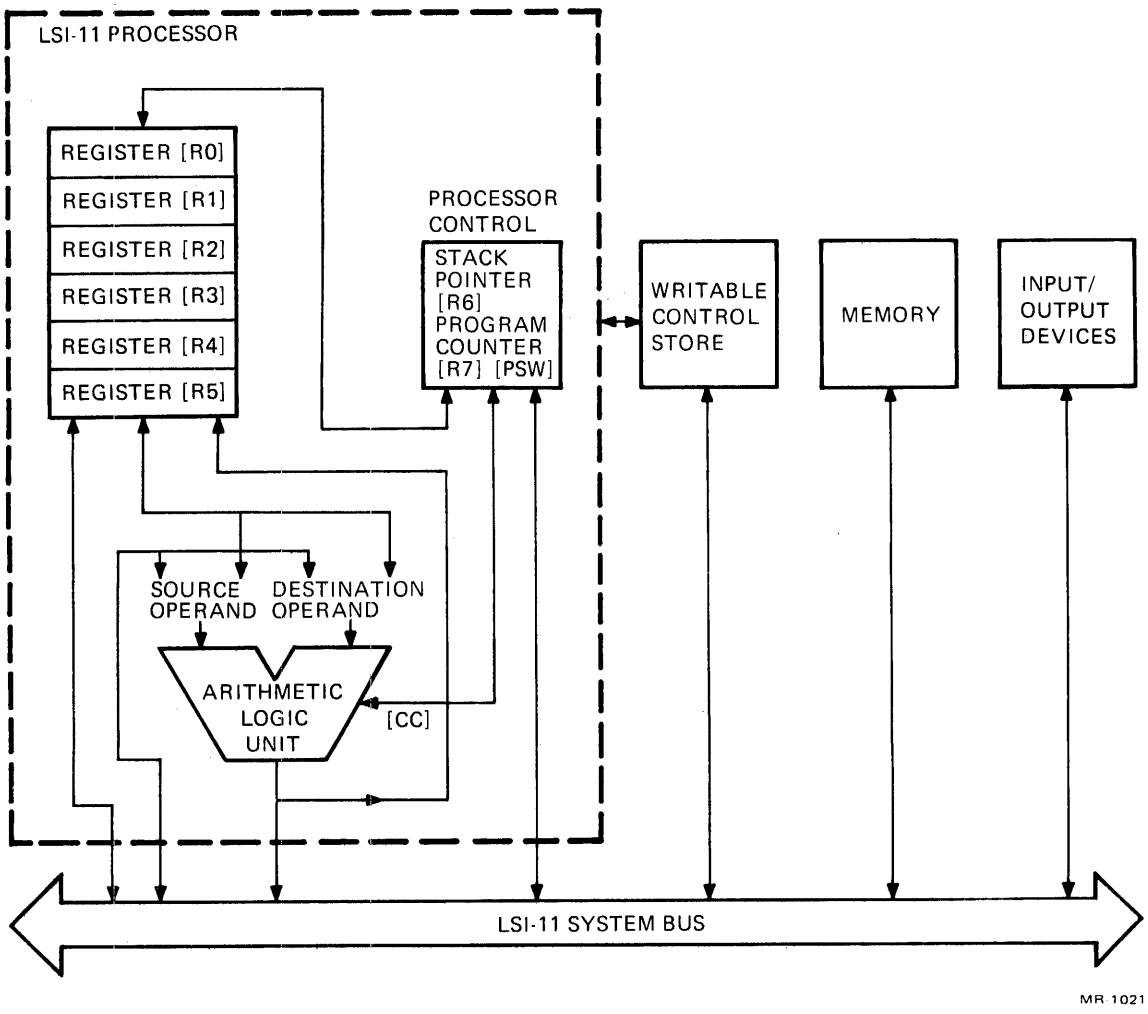
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	1	1	1	1	0	1	1	1	0/1	0/1	0/1	0/1	0/1	0/1

OP CODE OCTAL : 076700 THROUGH 076777
OPERATION : DESIGNED BY THE USER

MR-1020

LSI-11 Processor-Writable Control Store Interconnection

Figure 2-8



The WCS module is connected to the LSI-11 machine in two ways:

1. LSI-11 System Bus
2. Micromachine Microinstruction Bus

2.2.5.1 LSI-11 System Bus Connection - This connection is established by the printed circuit contact fingers which insert into the system backplane. The WCS module contains 1024 24-bit microlocations which may be read and written via Programmed I/O operations. The LSI-11 system bus interconnection and WCS module control and data registers are detailed in Chapter 6.

2.2.5.2 Microinstrucion Bus Connection - The connection between the LSI-11 processor control area and the WCS module is established by a special MicroInstruction Bus (MIB) cable. The third MICROM socket on the LSI-11 module (E32 on the M7264), which is usually occupied by the EIS/FIS option, provides access to the MIB. The processor control area sends microlocation address information to the WCS module and the contents of the selected location are returned for use by the microprocessor as a microinstruction. The processor/WCS port is Read Only; WCS memory contents can be read but not altered by the LSI-11 processor via this port. The WCS memory performs the same function with respect to the micromachine as do the MICROMs located on the LSI-11 module. The details of microinstruction access are contained in Chapter 3.

2.3 MACHINE OPERATION

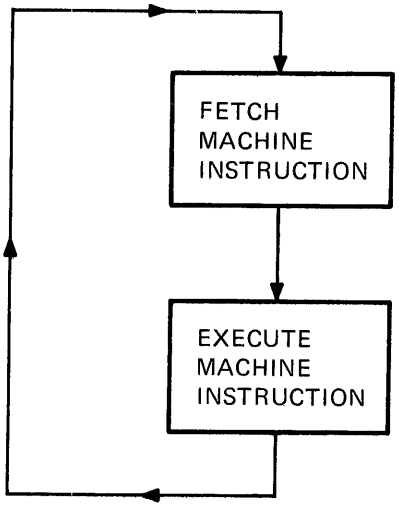
2.3.1 Basic Machine Cycle

The basic machine cycle in its simplest form is a repeating sequence of Fetch Machine Instruction - Execute Machine Instruction operations as illustrated in Figure 2.9. The Fetch operation requires one DATI (Data-In) bus cycle and the Execute operation may require one or more bus cycles as determined by the instruction being executed.

2.3.1.1 Bus Error Trap - Implemented with the basic machine cycle is the system bus error trap mechanism, which aids the processor in recovering from a system bus error. A bus error occurs whenever the processor addresses a memory (or device) location which does not exist on the system bus or which does not respond due to a malfunction. The bus error trap is initiated by a timeout sequence which is implemented in the processor bus control circuitry. The bus error condition occurs when no memory or device response is received within 10 microseconds after initiating the bus cycle.

Basic LSI-11 Machine Cycle

Figure 2-9



A bus error can occur for the Fetch DATI bus cycle as well as for any bus cycles performed during the Execute operation. The trap which responds to the bus error causes the processor to push the Program Counter (PC) counter and Processor Status Word (PSW) values onto the stack and to load new values from the trap vector locations (10 and 12). The vector addresses contain the new PC and the new PSW. When these operations have been completed, the processor will fetch its next instruction from the location pointed to by the new PC. However, it is also possible that a fetch from the memory location pointed to by the stack pointer will also cause a bus error, resulting in a double bus error condition. The processor response to this condition is to enter the Halt state. The single and double bus error trap operations are illustrated in Figure 2.10.

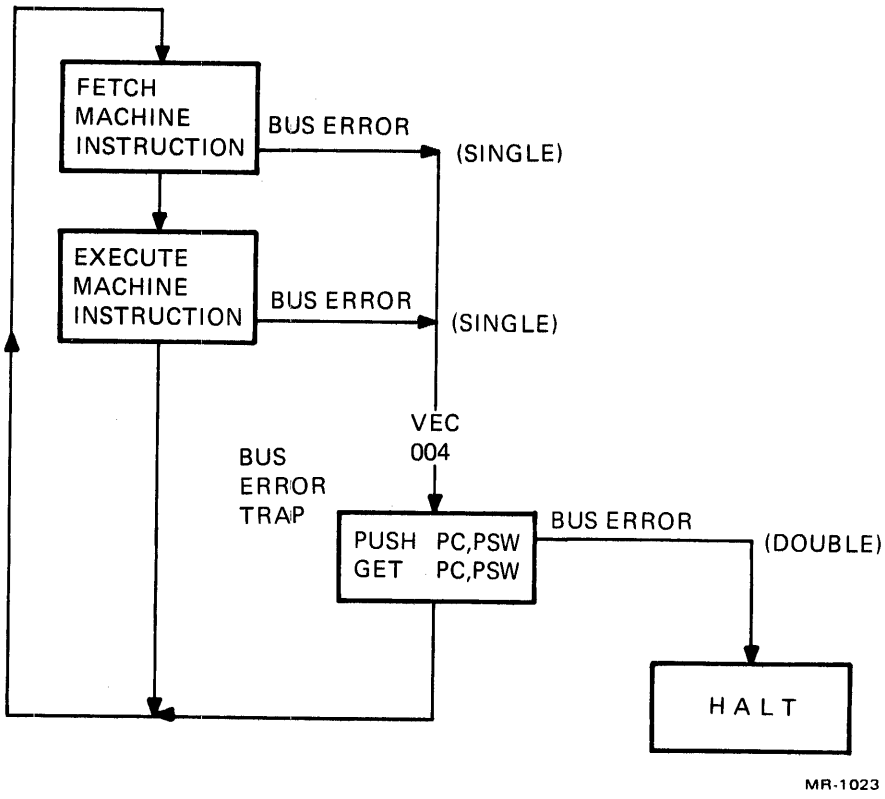
2.3.1.2 External Interrupt - The basic Fetch-Execute machine cycle can be modified to allow an external event to gain control of the processor, as illustrated in Figure 2.11. Once the execution of a fetched machine instruction is completed, control passes to a decision path which interrogates the machine interrupt status. If an interrupt is pending, the processor action is nearly identical to that caused by the trap (the current PC and PSW values will be pushed on the stack and new values loaded from the interrupt vector). The interrupt vector may be automatically known to the processor (as in the Event interrupt case) or the processor can obtain the vector from the interrupting external device (see Section 2.2.1.2).

When the PC and PSW contents are replaced, control is returned to the interrupt decision. This control flow enables the creation of an interrupt (and trap) priority structure that determines which one of a number of simultaneously active interrupts is to receive service. Note that the granting of service to external device interrupts is still dependent upon electrical bus position relative to the processor. Since control is always returned to the top of the decision path, all interrupts are assured of service, in order of decreasing priority, before normal program execution resumes.

2.3.1.3 Combined Trap and Interrupt Cycle - Because of the similarity between the interrupt and trap operations, a single microprogrammed routine implements the required functions. The vector information is stored in an internal register before the routine is entered. An additional flag internal to the processor indicates the double bus error condition (see Section 8.9). The combined interrupt and trap control flow diagram is illustrated in Figure 2.12.

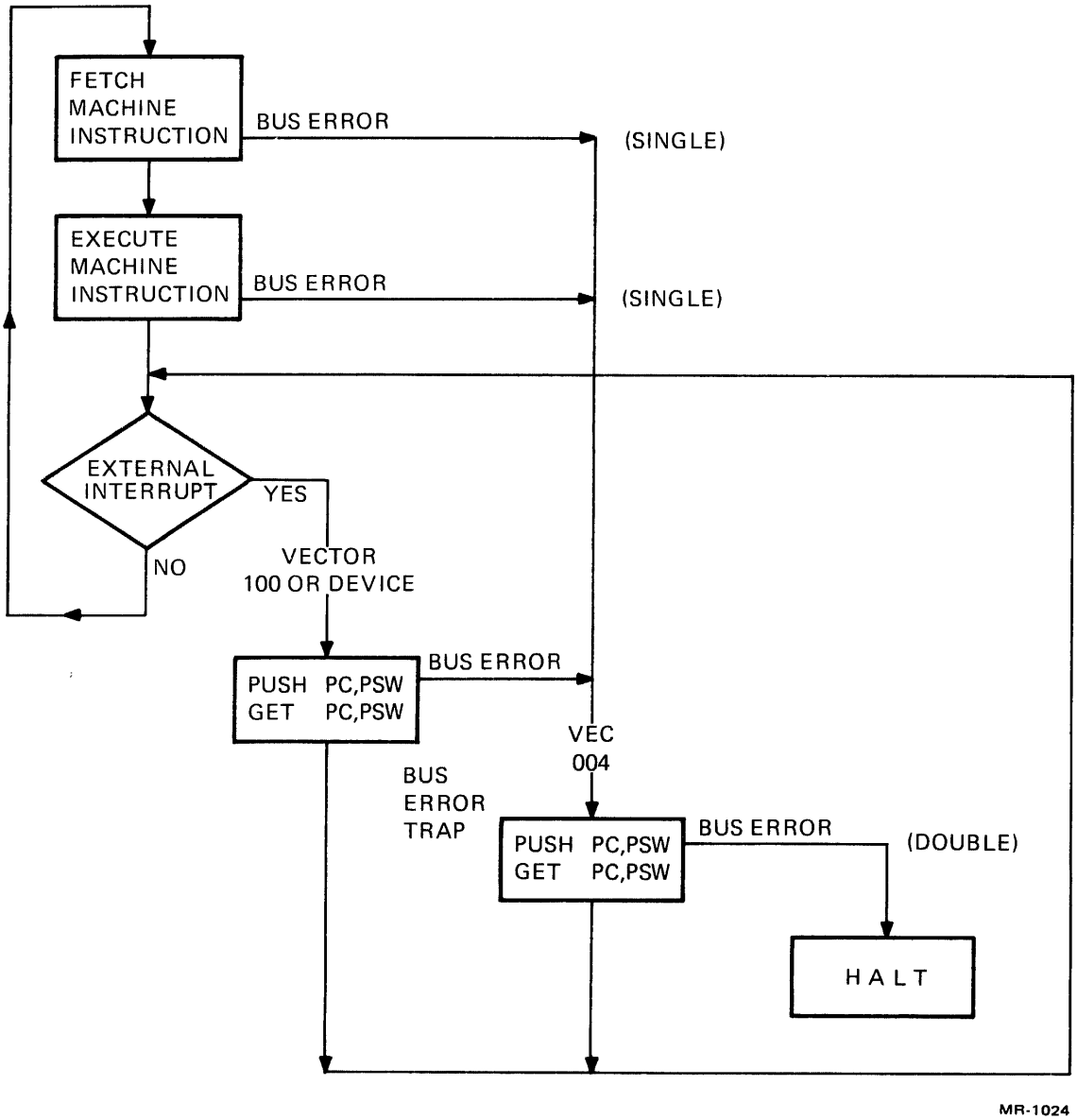
Single and Double Bus Errors

Figure 2-10



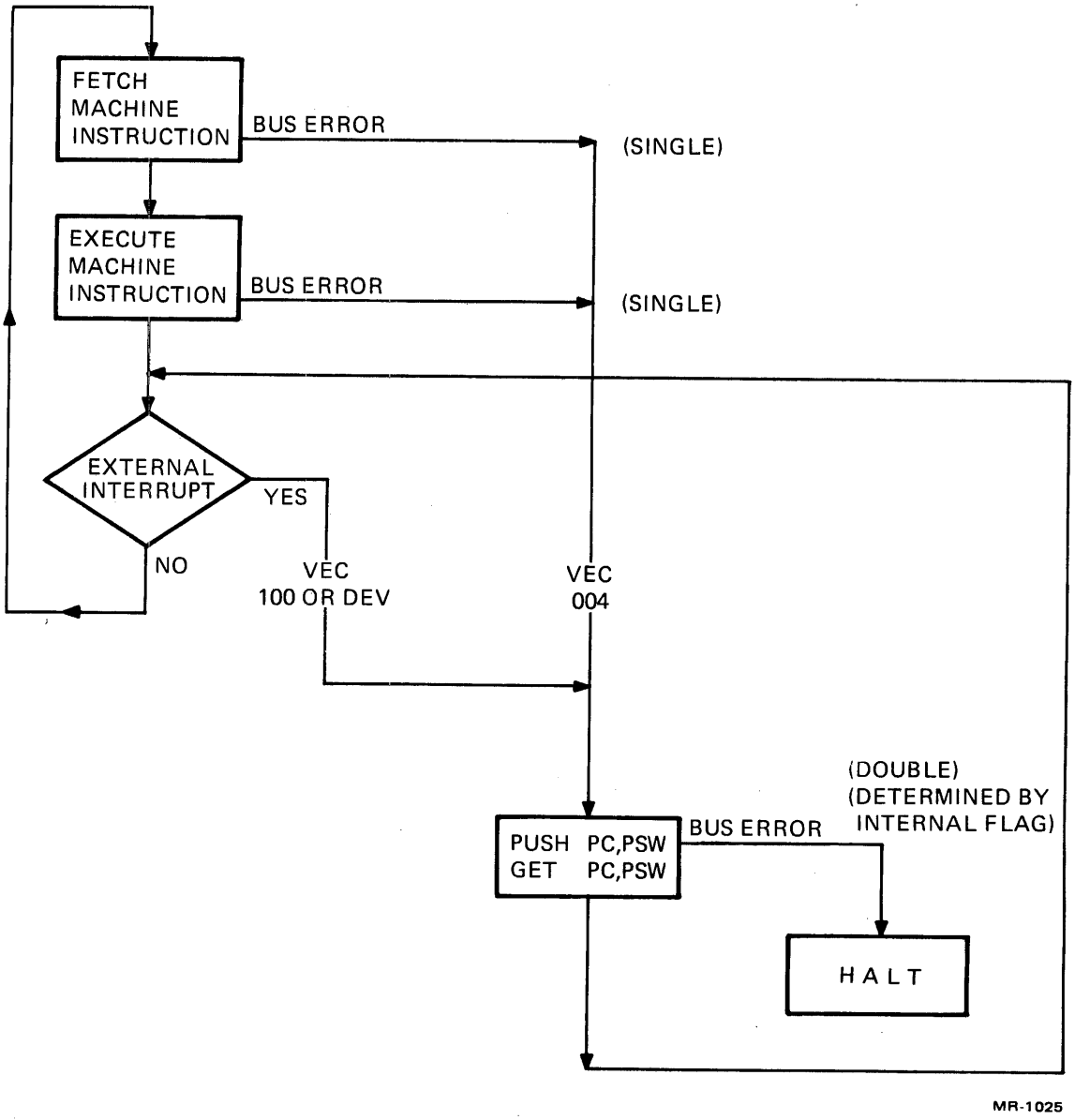
External Interrupt and Bus Error Trap

Figure 2-11



Combined Interrupt and Trap Operations

Figure 2-12



2.3.2 Complete Machine-Level Operating Cycle

The essential principles of the control flow configuration illustrated in Figure 2.12 may be summarized as follows:

1. The untrapped, uninterrupted machine cycle is a repeating sequnce of Fetch Machine Instruction - Execute Machine Instruction operations.

2. The trap facility allows the processor to recover from a (single) bus error condition.

3. The control flow configuration of the interrupt and trap operations, in conjunction with the hardware stack, implements the interrupt priority hierarchy.

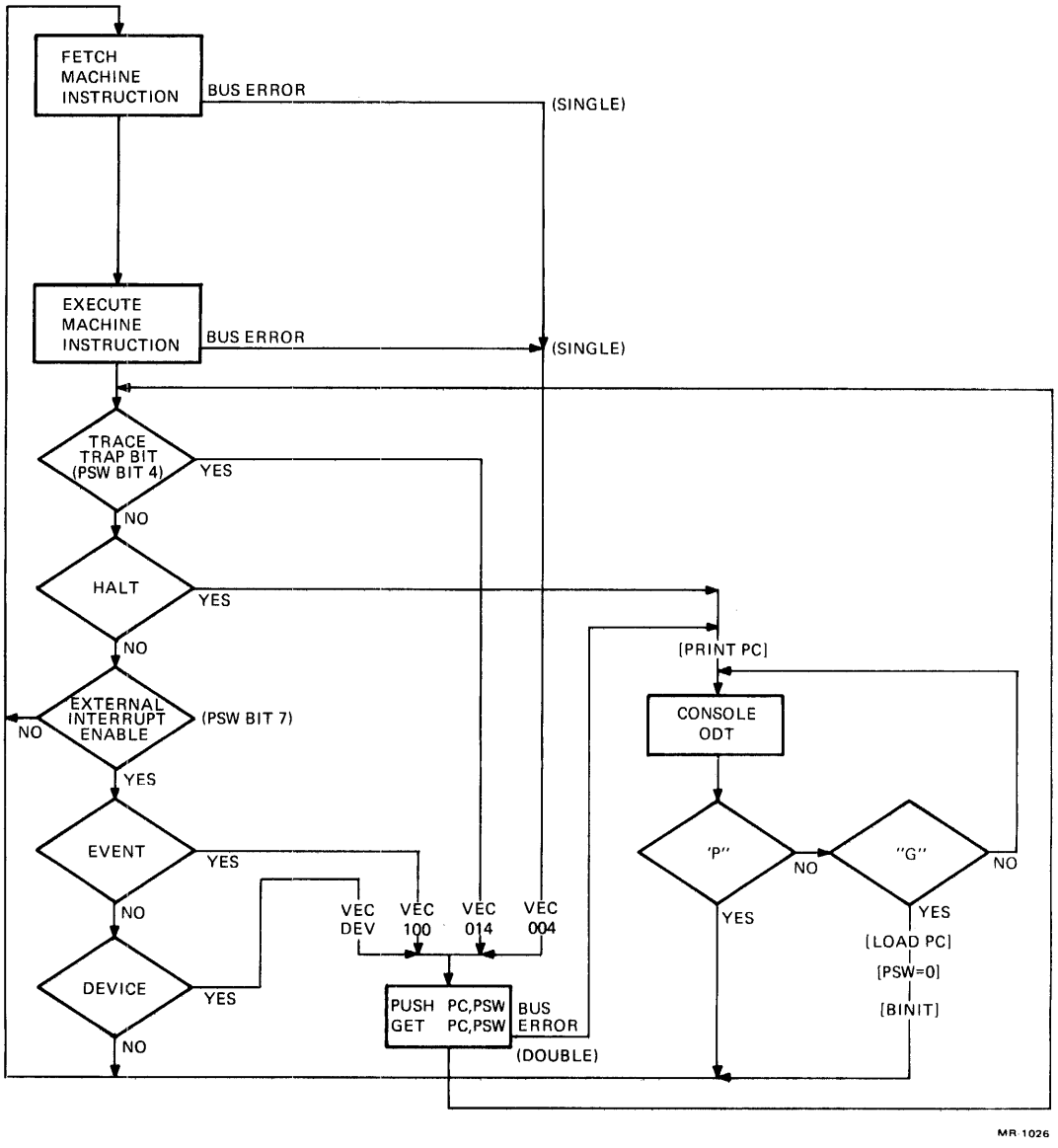
These principles are also apparent in Figure 2.13, which illustrates a more exact machine-level control flow diagram. The complete diagram must detail the transfer of control between the machine and micromachine levels and is presented in a later section. Figure 2.13 illustrates the Fetch-Execute-Trap/Interrupt machine cycle which is sufficient for conventional machine level programming. It shows the location in the control flow of the Trace Trap Bit (PSW BIT <4>), and the External Interrupt Enable Bit (PSW Bit <7>). Also shown are the two paths to the Console ODT/Halt state as well as the two paths which leave ODT and re-enter the Fetch-Execute cycle of the Run state.

2.3.2.1 Run/Halt Portion - The Run/Halt portion of Figure 2.13 is extracted and illustrated in Figure 2.14. The two means of entering the Halt state are: (1) the execution of the HALT machine instruction and (2) the assertion of the Halt interrupt. The latter is asserted via the bus control signal BHALT L by setting the front panel Run/Halt switch to Halt or by pressing the console terminal BREAK key. The PC contents are printed on the terminal immediately upon entering the Halt state. This gives the location of the next instruction to be executed. Either the "P" or "G" commands may be entered by the operator at the console terminal. Entering the "P" (PROCEED) command passes control directly to the Machine Instruction Fetch operation. The "G" (Go) command loads a new PC value (nnnnnnG) and zeroes the PSW (which enables external interrupts) before passing control to the Machine Instruction Fetch operation.

2.3.2.2 Trap/Interrupt Portion - The Trap/Interrupt portion of Figure 2.13 is extracted and illustrated in Figure 2.15. The Trace Trap has the highest machine-level priority and affects control flow before any external event. The Trace Trap uses the same vector value, (014), as the Breakpoint Trap (BPT) instruction. The hardware Trace Trap, implemented via PSW BIT 4, and the software Trace Trap, implemented via the execution of the BPT instruction, are used to support program debugging.

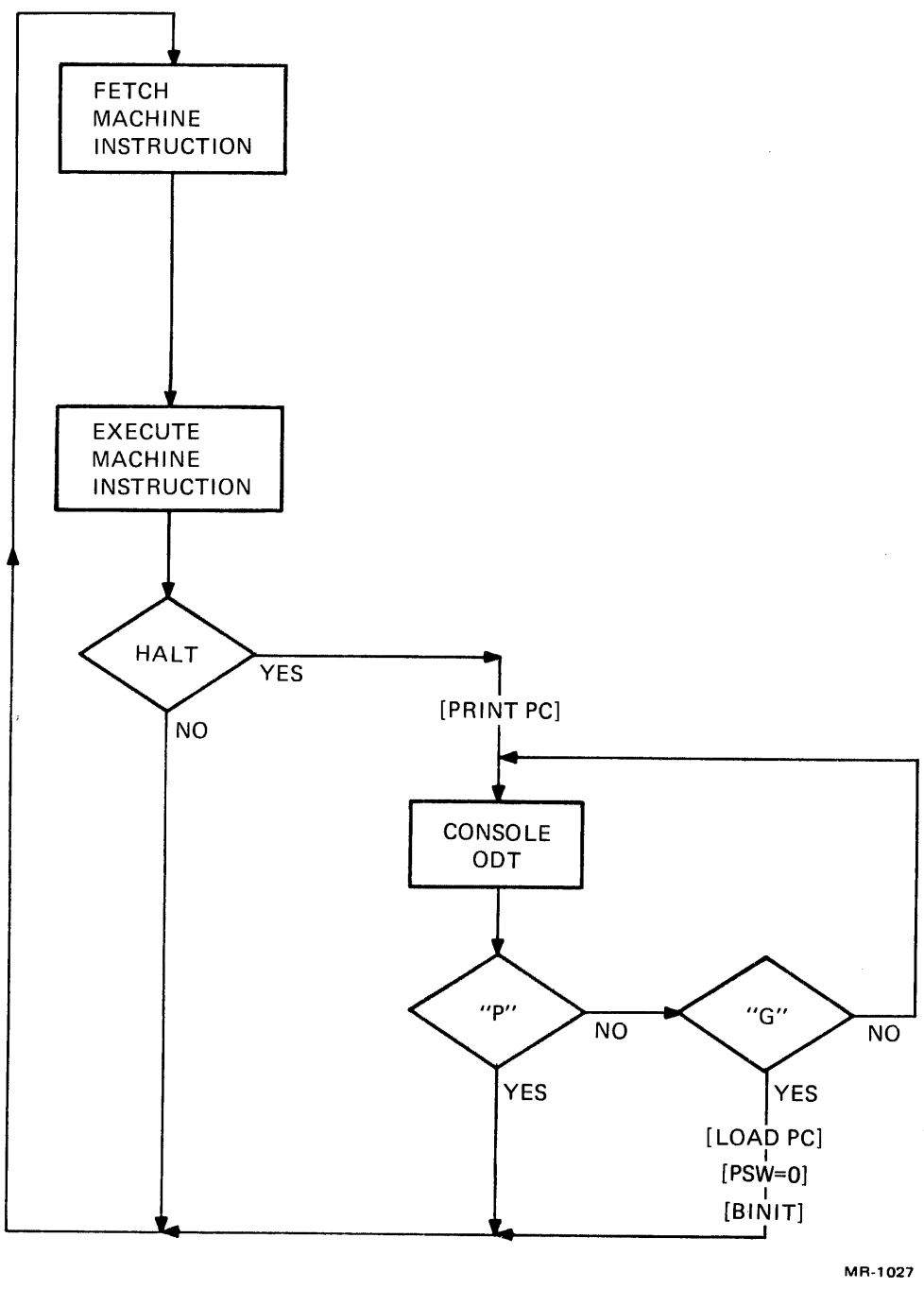
Complete Fetch-Execute-Interrupt Cycle

Figure 2-13



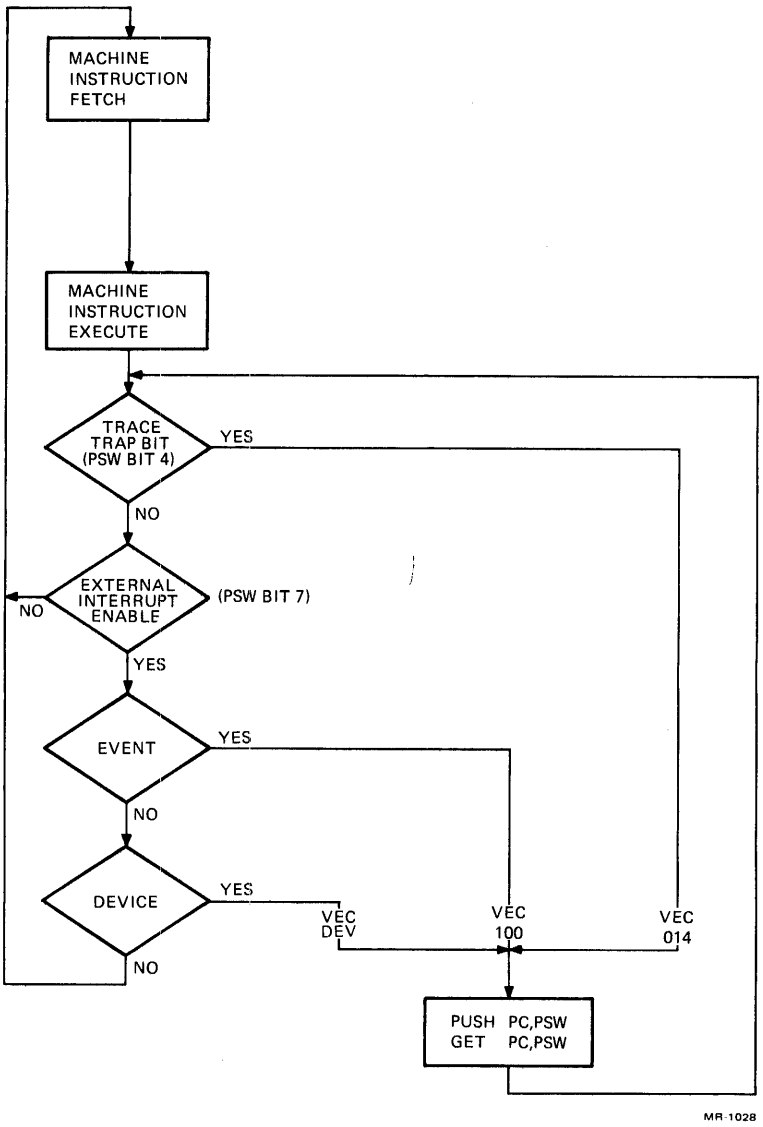
Run Halt Portion of Machine Cycle

Figure 2-14



Interrupt and Trap Portion of Machine Cycle

Figure 2-15



The External Interrupt Disable (PSW Bit <7>=1) can divert control flow around Event and device interrupts. When enabled, the Event interrupt, asserted via the bus signal BEVNT L, will receive service before any device interrupts. All external devices having interrupt capability assert the same interrupt request line, BIRQ L. Interrupt priority external to the processor is determined by the position of the module in the LSI-11 backplane.

2.3.3 Complete Machine-Micromachine Operating Cycle

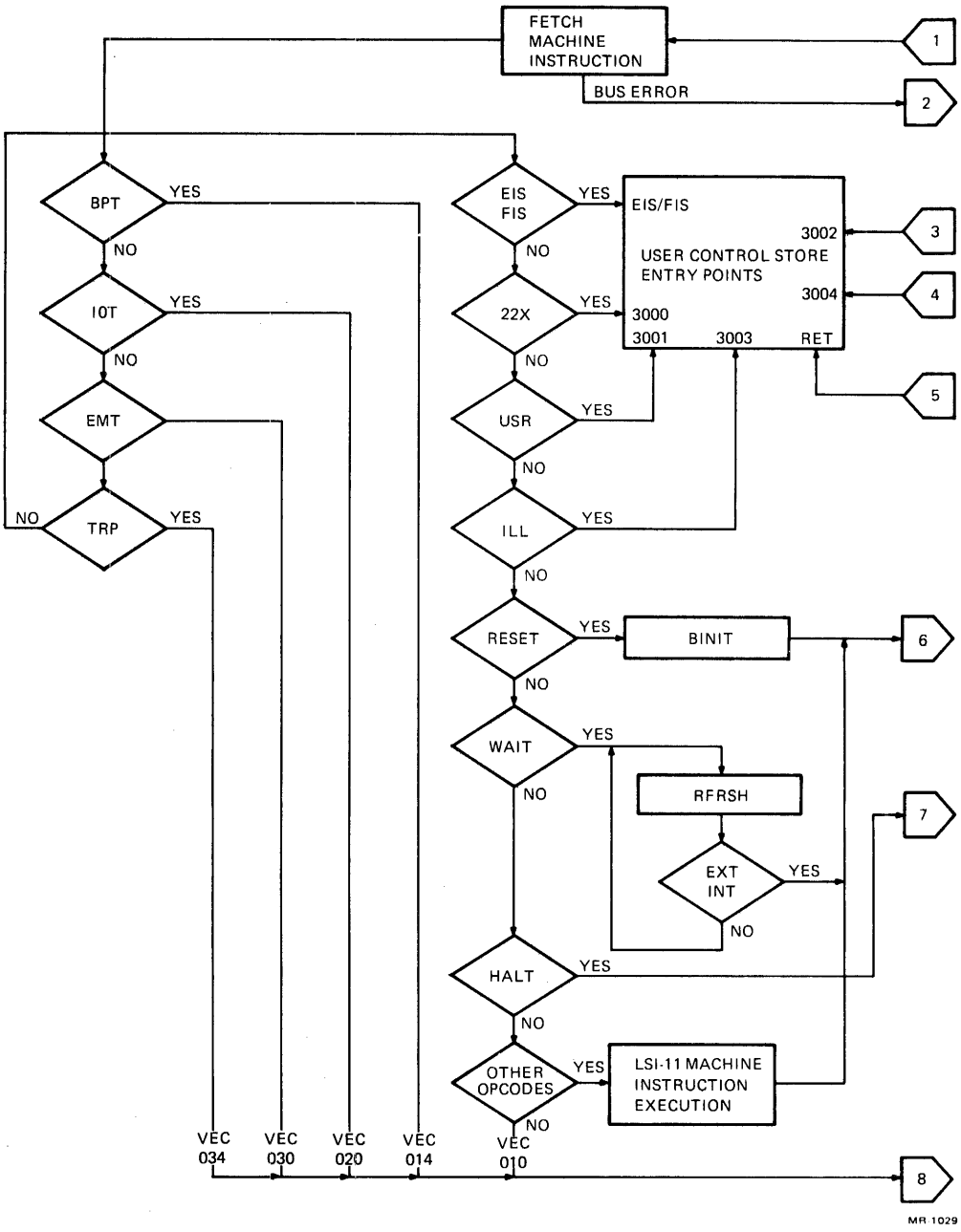
The complete control flow diagram which makes apparant the transfer of control between the machine and micromachine levels is illustrated in Figures 2.16-1 and 2.16-2. A greater number of machine instruction examples are used to represent the decisions made during the Execute Machine Instruction operation of the basic machine cycle. Many of the examples used demonstrate the internal sharing of the microprogrammed Trap/Interrupt service routine.

2.3.3.1 Bus Error Processing - The entry point at the top of the Power-Up decision flow in Figure 2.16-2 is the result of a hardware reset in the case of a bus error. A wait state occurs due to an unresponding bus device, but the wait is terminated by a 10 microsecond timer on the LSI-11 CPU module that resets the microprocessor. When reset, the microprocessor begins executing microinstructions at microlocation 0001. The FDIN (Fast Data-In) operation is used to to determine how control flow arrived at that entry point, either by bus error or by Power-Up. If a bus error was the cause, only one of the 6 possible bus error types will result in a trap through vector location 004. The first possible bus error is used by a microprogrammed ODT routine to determine memory size (Boot Self-Size). The second bus error type occurs when the operator attempts to examine (using Console ODT) a memory or device register which does not respond. In this case, control is returned to a point within the ODT microcode and a "?" is printed on the console terminal. The next three bus error types are regarded as fatal and result in a processor Halt. These errors occur (1) when an interrupting device does not return a vector, (2) when a microprogrammed refresh does not receive a reply, or (3) when a double bus error occurs.

2.3.3.2 Trap/Interrupt Processing - The Trap/Interrupt decision flow indicates the priority with which the micromachine interrupt register is interrogated. This internal micromachine interrupt register is illustrated in Figure 2.17. Of the seven interrupts, four are external and three are internal. An internal interrupt in this context is one which can be set or reset only under microprogram control.

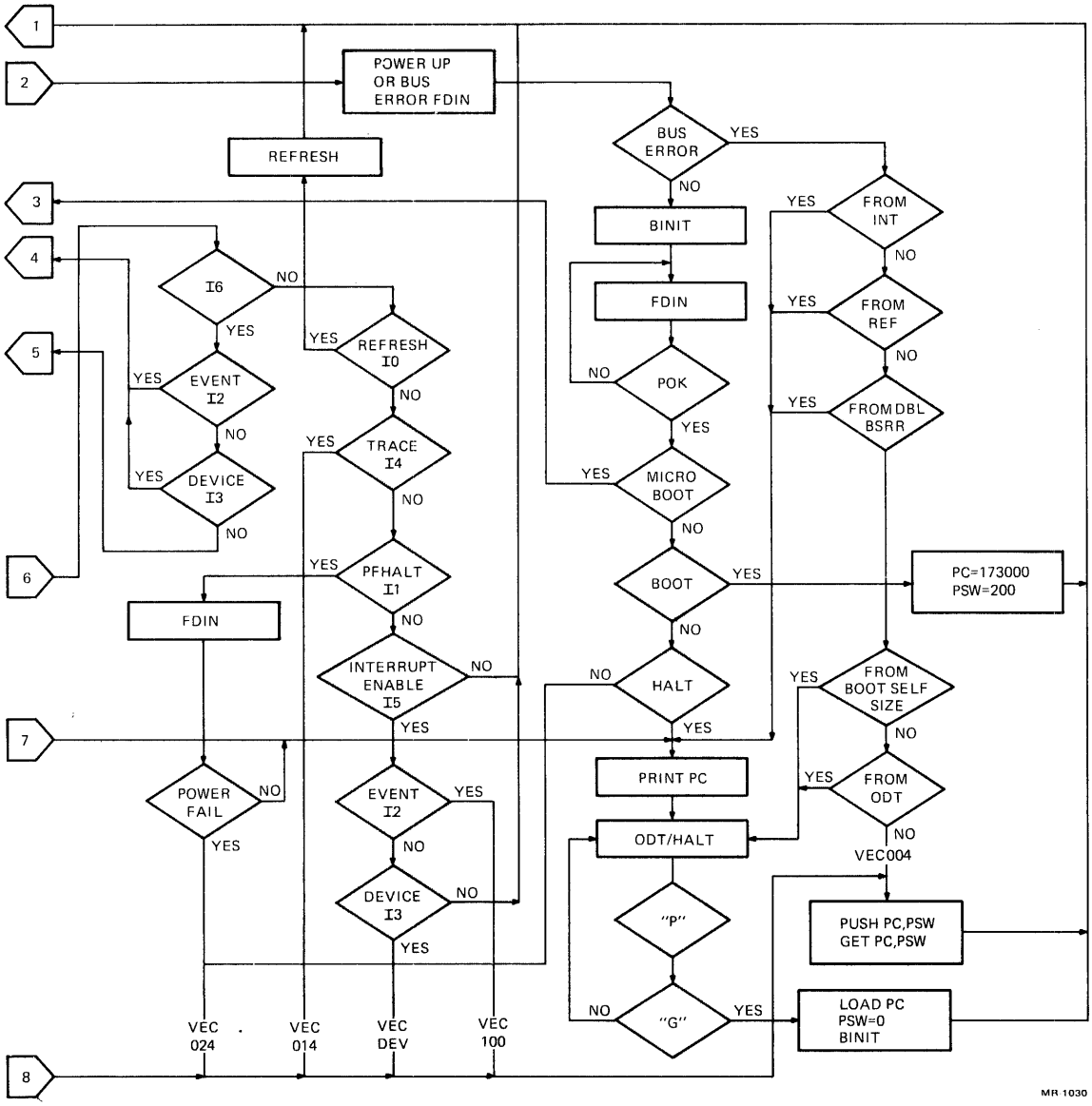
Complete Machine-Micromachine Control Flow Diagram

Figure 2-16-1



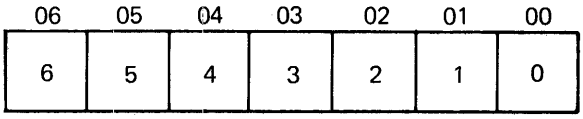
Complete Machine-Micromachine Control Flow Diagram

Figure 2-16-2



Micromachine Interrupt Register

Figure 2-17



LISTED IN ORDER OF DECREASING PRIORITY

I6: INTERNAL TEST FOR EXTERNAL INTERRUPTS I 2 OR I 3
I0: EXTERNAL DYNAMIC MEMORY REFRESH
I4: INTERNAL TRACE TRAP BIT (PSW BIT-4)
I5: EXTERNAL POWER-FAIL/HALT INTERRUPT
I5: INTERNAL ENABLE EXTERNAL INTERRUPTS I 2 AND I 3 (COMPLEMENT OF PSW BIT-7)
I2: EXTERNAL EVENT INTERRUPT
I3: EXTERNAL DEVICE INTERRUPT

MR-1032

The highest priority interrupt is I6 which is used only at the micromachine level to determine whether an external interrupt (I2: Event, I3: Device) is pending. This facility enables a lengthy microroutine to abort execution and grant interrupt service to the external Event or Device interrupts only. If I2 or I3 is asserted, while I6 is enabled, control is transferred to microlocation 3004 from any microinstruction which has the RSVC bit (bit <17>) set to a "1" after the next subsequent microinstruction is executed (only if neither one of the microinstructions is a Jump or Return from Subroutine microinstruction). The Refresh interrupt (I0) has the highest priority of all interrupts external to the micromachine. When enabled via a jumper on the LSI-11 processor, I0 is asserted every 1.6 milliseconds. This interrupt is usually transparant to the machine-level operation of the processor.

The refresh (RFRSH) operation which follows I0 has control transfer links to the WAIT (WAT) machine instruction, to the FDIN-POK sequence and to the ODT/Halt routine. These links exist by means of special translations which are implemented in the microprocessor Control chip.

The Trace Trap Bit (I4), the External Interrupt Enable Bit (I5) and the two external interrupts, Event (I2) and Device (I3), are unchanged from their representation in Figure 2.15. However, the external Halt interrupt shown in the earlier figure is shared with the Power-Fail function (PFHALT). The micromachine employs a Fast Data-In operation to determine which event has occured.

The Trap/Interrupt priority structure is the composite result of the internal micromachine interrupt register priorities, the microprogrammed Power-Up and bus error routines, and the over-all control flow configuration. The combined priorities may be ordered as follows:

1. Bus Error Trap
2. External Interrupt Test (I6)
3. Memory Refresh
4. Machine Instruction Traps
5. Hardware Trace Trap
6. Halt Line
7. Power-Fail Trap
8. Event Line Interrupt
9. Device (Bus) Interrupt Request

2.3.3.3 Power-Up Processing - If the top decision in the Power-Up flow determines that no bus error occured, a Power-Up routine begins. The first event is to issue an initialization signal on BINIT L and then wait for bus power to come up (BPOK H active). If enabled, dynamic memory refresh will also occur in this sequence. When bus power arrives, the module jumpered Power-Up option is accessed by the microprocessor through the FDIN operation and performed. The four possible Power Up modes are listed below:

MODE 0 - The PC and the PSW are loaded from locations 24 and 26, respectively, and machine execution begins. If BHALT L is asserted, control will be returned to Console ODT/Halt.

MODE 1 - Control passes immediately to Console ODT/Halt.

MODE 2 - The PC is loaded with 173000, the PSW with 200 (External Interrupts Disabled), and machine execution begins. As in MODE 0, the processor will halt before the first instruction is executed if BHALT L is asserted.

MODE 3 - Control immediately goes to microlocation 3002, the control store entry point for a user-microprogrammed bootstrap routine. The status of BHALT L has no effect on this control transfer. If control store does not exist at that microaddress, a trap to vector location 10 occurs.

CHAPTER 3

THE LSI-11 MICROMACHINE STRUCTURE

3.1 GENERAL

The LSI-11 processor illustrated in Figure 2.3 is implemented with a Large-Scale-Integration (LSI) microprocessor which is microprogrammed to emulate the PDP-11 architecture. Emulation is the technique whereby the general resources of the microprocessor are made to serve as the specific architectual components (GP registers, 16-bit ALU, etc.) of the LSI-11. This chapter contains a detailed description of the microprocessor which is made up of the Control chip, the Data chip, and one or more 40-pin MICrocode Read Only Memory (MICROM) chips.

For the purposes of microprogramming, it is useful to view the Control and Data chips as a single microprocessor. The major interconnection path between the two chips is the MicroInstruction Bus (MIB). In addition to providing a common bus for the micromachine instructions, the MIB is also time-multiplexed to provide auxilliary control paths between the Data, Control, and MICROM chips.

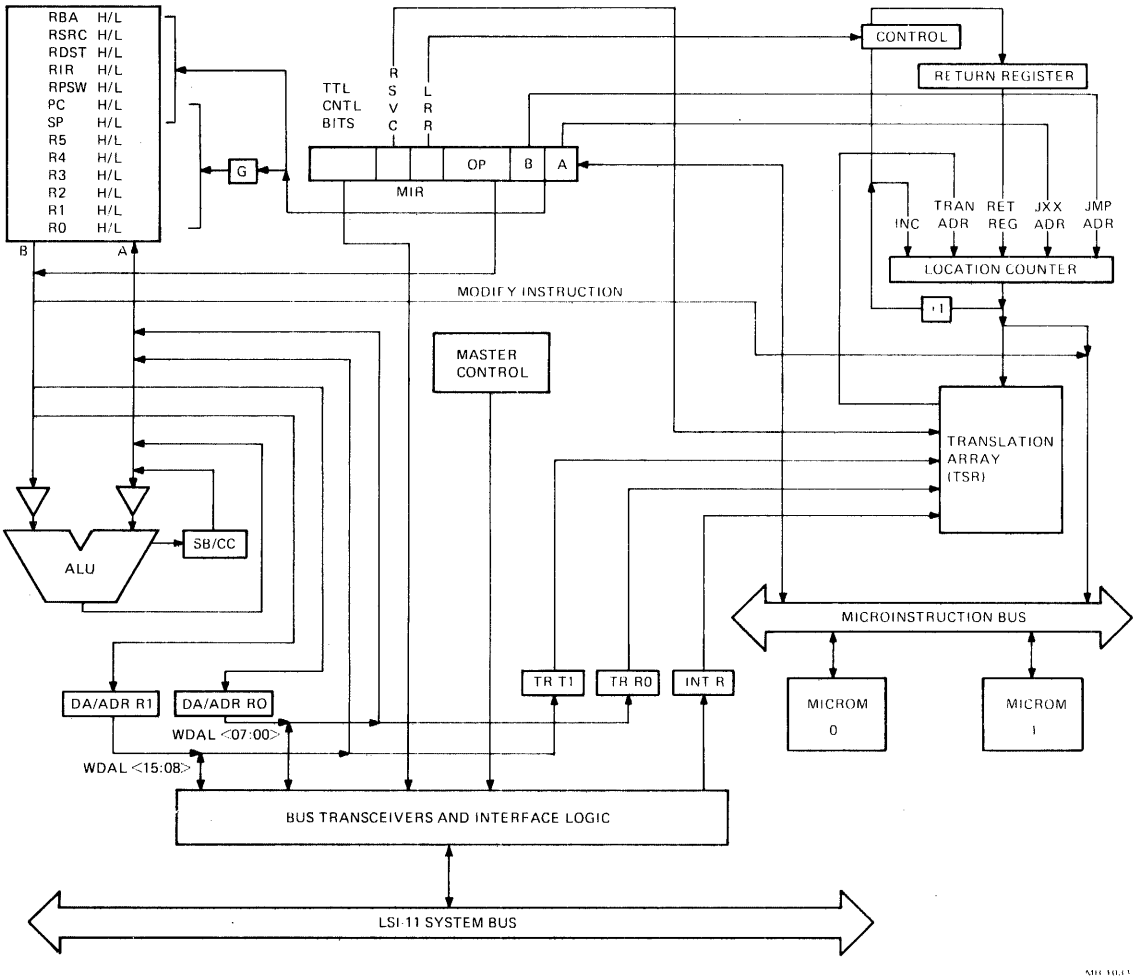
3.2 THE MICROPROCESSOR

The complete microprocessor is illustrated in Figure 3.1. There are several similarities which may be drawn between this illustration and Figure 2.3 (LSI-11 Processor). Both figures contain a register file and an arithmetic logic unit; there is also a memory component evident in both figures. (Figure 2.3 contains the system memory while Figure 3.1 contains the MICROMs). However, the programs contained in the MICROMs (along with the translation array) efficiently organize the resources of the microprocessor to emulate the machine-level architecture shown in Figure 2.3. An example of the differences in the two architectures may be seen in the register files. All six of the general purpose registers in the LSI-11 processor are 16-bits wide and support both byte and word addressing. However, the register file in the microprocessor is composed of 26 8-bit bytes which support a combination of direct and indirect addressing techniques. The microprocessor register labels correspond to the six general purpose registers, the stack pointer, the PC, and the five internal registers indicated in Figure 2.6.

There is a master control section in Figure 3-1 which is roughly analogous to the processor control section in Figure 2.3. All micromachine instructions, whether fetched from MICROM or from the WCS

Microprocessor Control and Data Chip Detail (Including MICROM)

Figure 3-1



module, are loaded into the microinstruction register for execution at the micromachine level. The two Data/Address (DA/ADR) registers, the two Translation (TR) registers and the Interrupt (INT) register provide register interface and buffer functons between the micromachine and the world of the LSI-11 system bus. To complete the analogy with Figure 2.3, the LSI-11 system bus is to the microprocessor as the Input/output devices are to the LSI-11 processor.

Figure 3.2 provides an overview of microprocessor operations. The figure is in the same format as Figure 2.4, (Processor Control Functions). The Compute and Reset controls and the Microlocation Address Generation functions are discussed in Section 3.3.2. The Microinstruction Execution functions are discussed in Section 3.3.1.

3.3 MICROPROCESSOR PARTITIONING

Microprocessor partitioning is illustrated in Figure 3.3. All chips within the micromachine (Control, Data, MICROMs) receive a four-phase micromachine clock from the LSI-11 circuitry. The 22-bit microinstruction bus provides a communication path between all micromachine elements. Sixteen lines of the MIB are common to all three chip types, while MIB<16> and MIB<17> are connected only between the MICROM chips and the Control chip. The last four lines, (MIB<21:18>) lead directly from the MICROM chips to the TTL control logic on the LSI-11 module. The TTL control logic is composed of the Bus Transceivers and Interface Logic shown in Figure 3.1. The operation of this logic is detailed in The Microcomputer Handbook. The signal paths and control functions on the LSI-11 system bus side of this logic were discussed in Chapter 2. The connections on the micromachine side are listed in Figure 3.4.

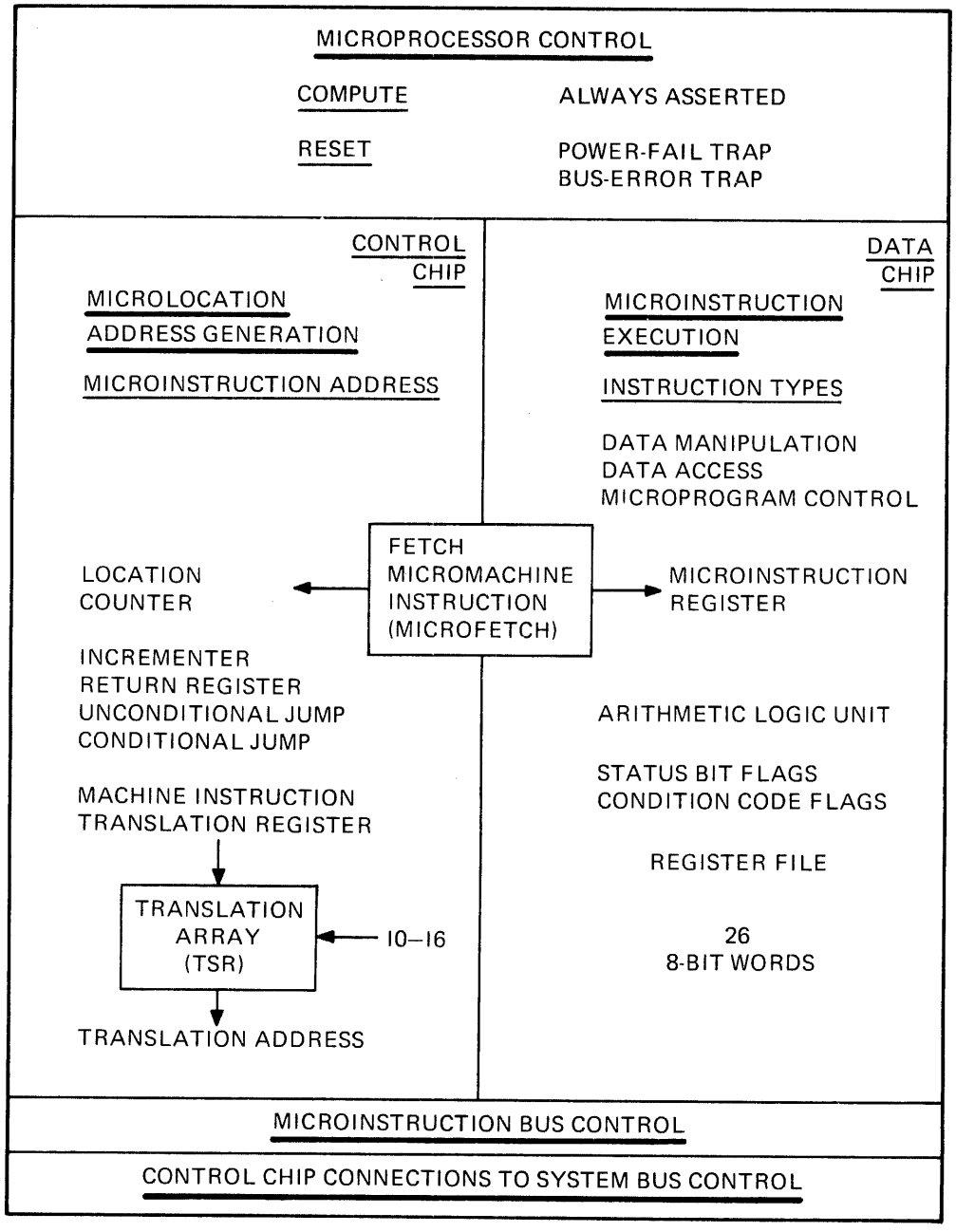
The Special Control Functions which are generated by the 4 highest MICROM bits (MIB<21:18>) are distributed to the two major logic areas as listed in Figure 3.5.

3.3.1 Microprocessor Data Chip

The components of the microprocessor which are implemented in the Data chip have been extracted from Figure 3.1 and are illustrated in Figure 3.6. The Data chip is connected to the sixteen lowest lines of the Microinstruction Bus (MIB<15:00>) and to the WAIT signal. The Data chip access to the LSI-11 System Bus is provided by WDAL<07:00> and WDAL <15:08>. Microinstructions fetched from MICROM are loaded into the MicroInstruction Register (MIR) for execution by the Data chip. The MIB also provides a signal path back to the Control chip for conditional jump instruction results. The 16 WDAL lines provide bidirectional access to the LSI-11 system bus lines, BDAL L<15:00>. The output path of the WDAL lines is buffered by the two Data/Address Registers, DA/ADR R0 and DA/ADR R1, which hold the output data or address information for the LSI-11 bus drivers.

Microprocessor Control Functions

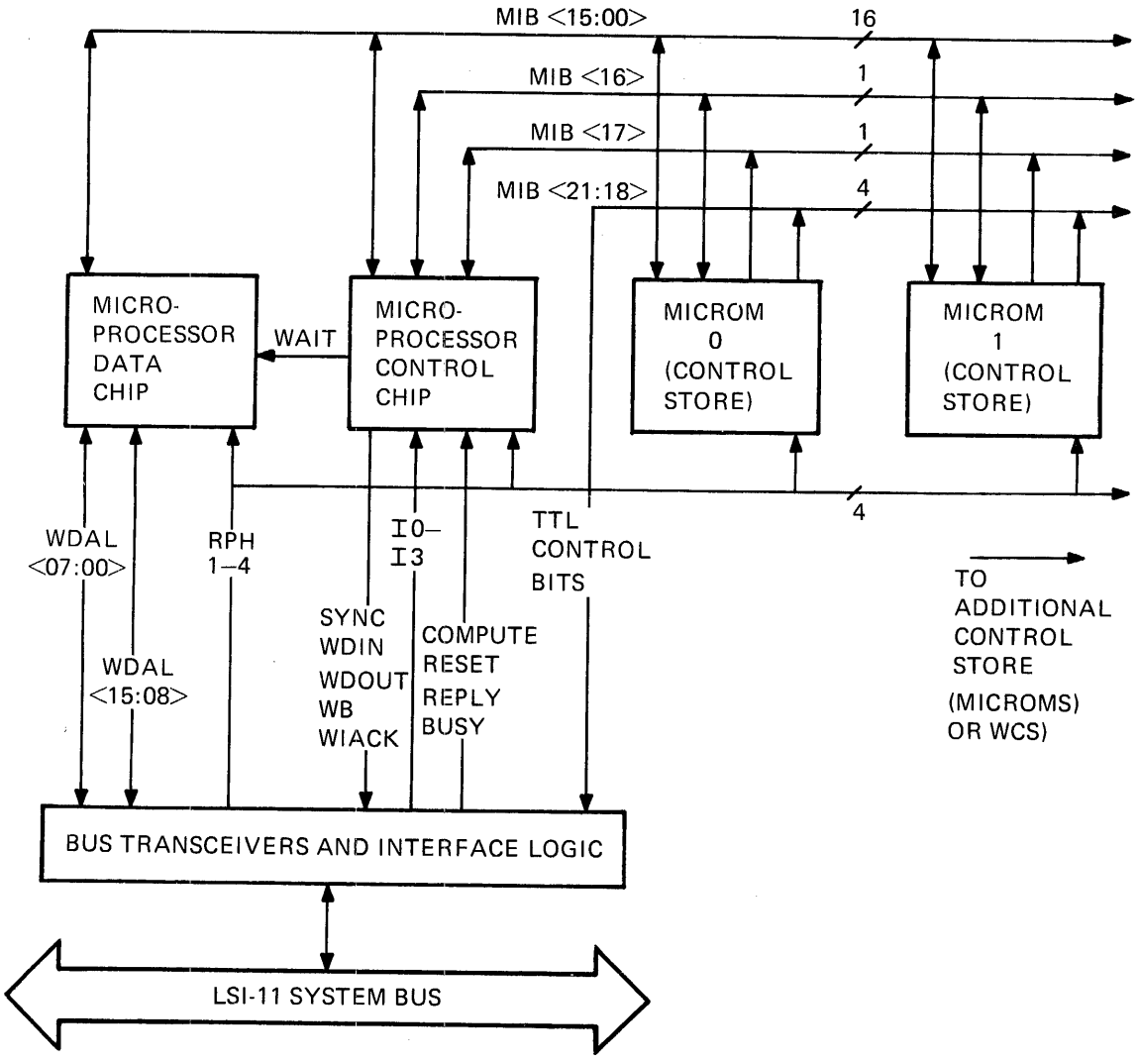
Figure 3-2



MR-1034

Microprocessor Inter-Chip Wiring Detail

Figure 3-3



Interface Logic Reference Chart

Figure 3-4

LSI-11 BUS DRIVER/ RECEIVER INTERFACE	BUS I/O CONTROL SIGNAL LOGIC		INTERRUPT CONTROL AND RESET LOGIC		SPECIAL CONTROL FUNCTIONS
DATA CHIP	CONTROL CHIP		CONTROL CHIP		MICROM
WDAL <07:00> (INPUT/OUTPUT)	WSYNC	(OUTPUT)	RESET	(INPUT)	MIB <21:18> (OUTPUT)
WDAL <15:08> (INPUT/OUTPUT)	WDIN	(OUTPUT)			
	WDOUT	(OUTPUT)	WIACK	(OUTPUT)	
	WWB	(OUTPUT)			
	REPLY	(INPUT)	10 RFIRQ	(INPUT)	
	BUSY	(INPUT)	11 IPIRQ	(INPUT)	
			12 EVIRQ	(INPUT)	
			13 IOIRQ	(INPUT)	

MR 1036

Special Control Function Distribution Chart

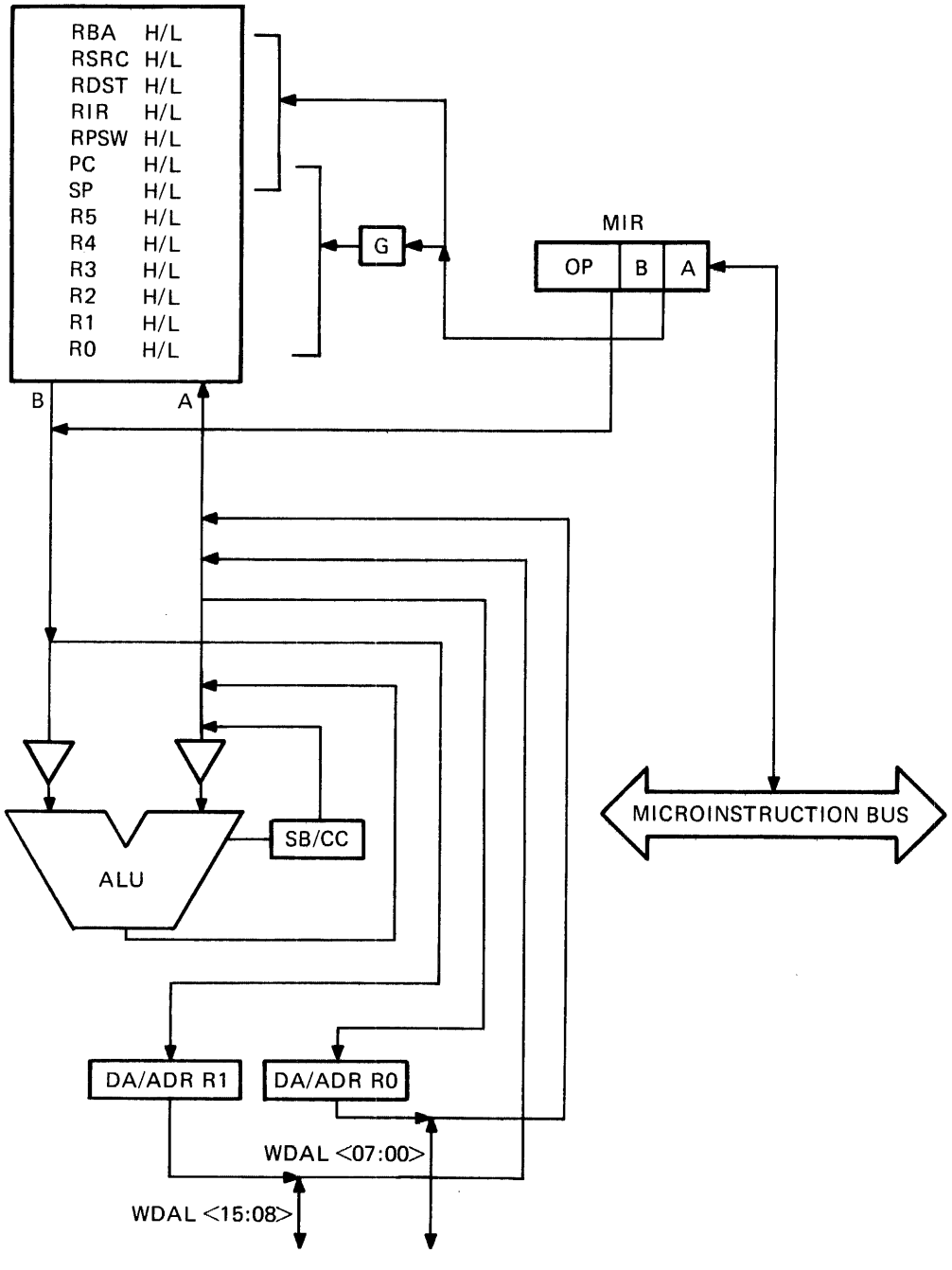
Figure 3-5

BUS I/O CONTROL SIGNAL LOGIC	INTERRUPT CONTROL AND RESET LOGIC
INIT (1) L	IFCLR AND SRUN L
INIT (1) H	TFCLR L
	RFSET L
	INITIALIZE SET
	FAST DIN
	PFCLR L
	EFCLR L

MR-1037

Microprocessor Data Chip Detail

Figure 3-6



The major elements of the Data chip are:

1. Microinstruction Register
2. Register File and Indirect Addressing Register
3. Arithmetic Logic Unit
4. Status Bit/Condition Code Flag Register

3.3.1.1 Microinstruction Register - Micromachine instructions delivered to the Data chip are loaded into the MIR for execution. The portion of the MIR contained in the Data chip is only 16 bits wide, reflecting the fact that only MicroInstruction Bus lines MIB<15:00> are connected to the Data chip. There are four types of microinstruction opcode formats, as listed below:

1. Jump Format
2. Conditional Jump Format
3. Literal Format
4. Register Format

The Jump Format is illustrated in Figure 3.7. The opcode (0) occupies bits <15:12> and the jump address is contained in bits <10:00>. A JMP Microinstruction can transfer micromachine control to any of the 2048 microlocations addressable within the control store.

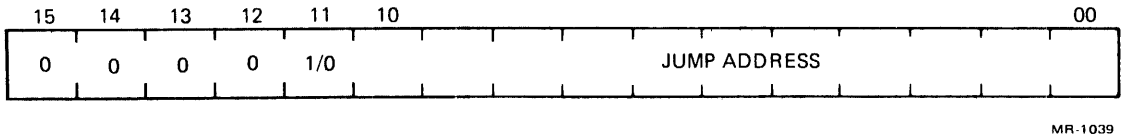
The second of the two possible Jump format instructions is determined by bit <11> of the microinstruction. When bit <11> is a 1, the microinstruction executes a Return From Subroutine (RFS) operation. The LSI-11 microprocessor supports only one level of subroutine at the microprogramming level.

The Conditional Jump Format is shown in Figure 3.8. A conditional jump microinstruction is indicated when the opcode field (bits <15:12>) is a 1 (0001). The condition field, bits <11:8>, indicate which condition must be satisfied for the jump to take place. The condition tests the ALU status bit and condition code flags as well as an Indirect Condition Status bit. If jump conditions are satisfied, micromachine control is transferred to the microlocation contained in the 8-bit address field. Since only 8 bits are available for address information, control may be transferred within the current 256 location page only. The remaining 3 address bits are equal to the corresponding 3 bits of the (updated) value of the current location counter.

The Literal Format microinstructions, illustrated in Figure 3.9, provide a way to generate constants in a microprogram. The contents of the 8-bit literal field MI<11:4> are determined at the time of microprogram assembly and may not be changed by the execution of any microinstruction. The remaining field, bits <3:0>, is called the "A" field and determines which microprocessor register (accessed through the "A" register port) will be the other 8-bit argument for the microinstruction.

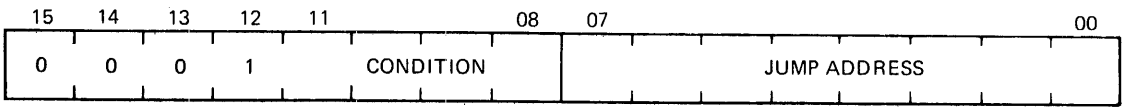
Unconditional Jump Format

Figure 3-7



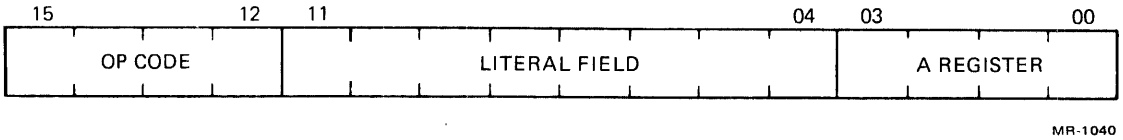
Conditional Jump Format

Figure 3-8



Literal Format

Figure 3-9



The Register Format microinstructions, illustrated in Figure 3.10, contain a second register designator field (MI<7:4>) called the "B" field. Note that the B port on the microprocessor register file is a Read Only port, whereas the A port is a Read/Write port. Register format microinstructions are either byte or word instructions. In the case of a byte operation, the microinstruction executes in one micromachine cycle and the operands are determined by the contents of the A and B register fields. A word operation requires two micromachine cycles to execute. The second pair of bytes of a word operand is determined by complementing the lowest bit of the register field during the second micromachine cycle. Thus the low bytes of a word instruction are supplied by the contents of Ra (or Rb) and the high bytes are supplied by the contents of Ra + or - 1 (or Rb + or - 1). The ALU status bit and condition code flags reflect the result of a word or byte operation. Direct and indirect register addressing is supported by both A and B register fields (see Section 3.3.1.2).

3.3.1.2 Register File and Indirect Addressing Register - The microprocessor register file is composed of 26 8-bit registers which provide temporary storage of data and address information for machine or micromachine programs. These registers supply operands to the ALU without system bus cycles and are, therefore, accessed at high speed.

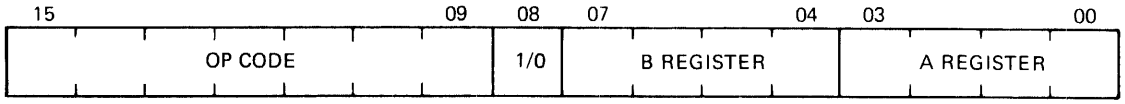
The contents of the register file may be accessed either directly or indirectly. One group of registers may be accessed only directly, a second group only indirectly, and a third group may be accessed in either way, as illustrated in Figure 3.11. This illustration uses the convention that directly addressed registers are preceded by "R" and indirectly addressed registers are preceded by "G". Figure 3.11 also indicates the machine- or micromachine-level use of each register.

The three highest bits of either the A and B register fields determine whether direct or indirect addressing is in effect for that field. When the three highest bits are all zero, indirect mode is indicated and the 3-bit G register contributes its three bits to the final, indirect register address. Since the lowest bit in each field may be either a 1 or a 0, indirect word addressing is achieved by complementing the lowest bit during the second cycle of a two-cycle word microinstruction. Figure 3.12 illustrates indirect addressing as used with a Literal microinstruction. It should be noted that all Literal format microinstructions are single-cycle and that since the register field is in the A position, register access is obtained through the Read/Write register file port.

Indirect addressing with a Register Format microinstruction is illustrated in Figure 3.13. The G register contents are the same for both the A and B field indirect addresses. The register addressing modes used in the Register Format may be any combination of direct and indirect modes. Additional specific details of Register Format microinstructions are supplied in Chapter 4.

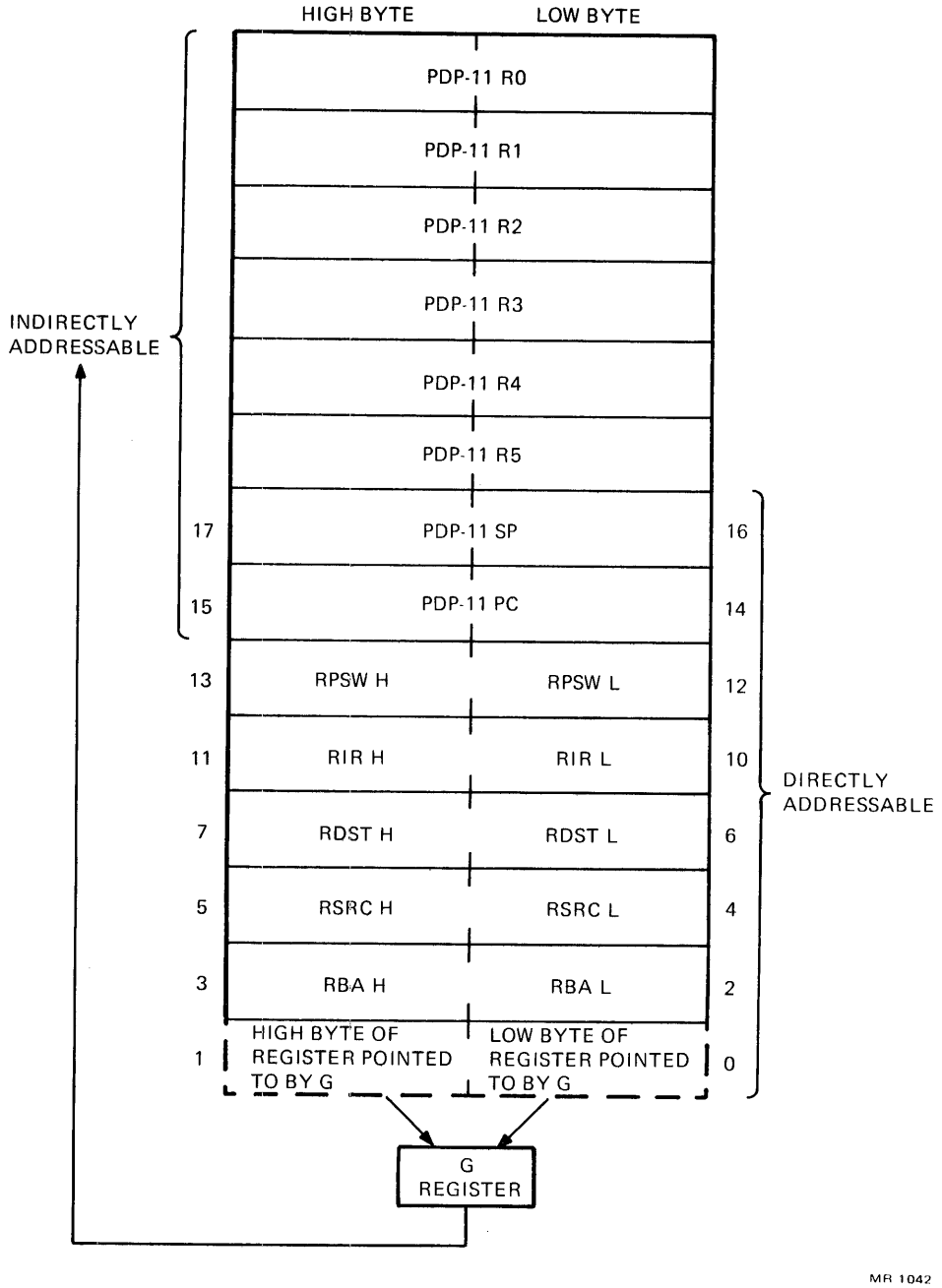
Register Format

Figure 3-10



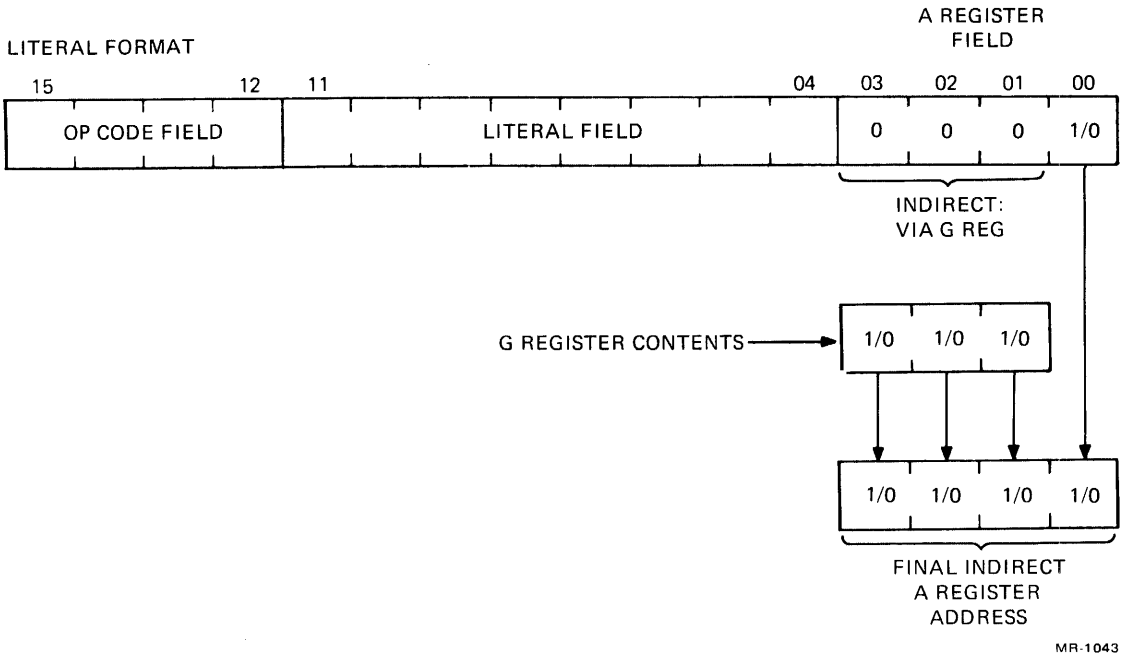
Direct and Indirect Register Addressing

Figure 3-11



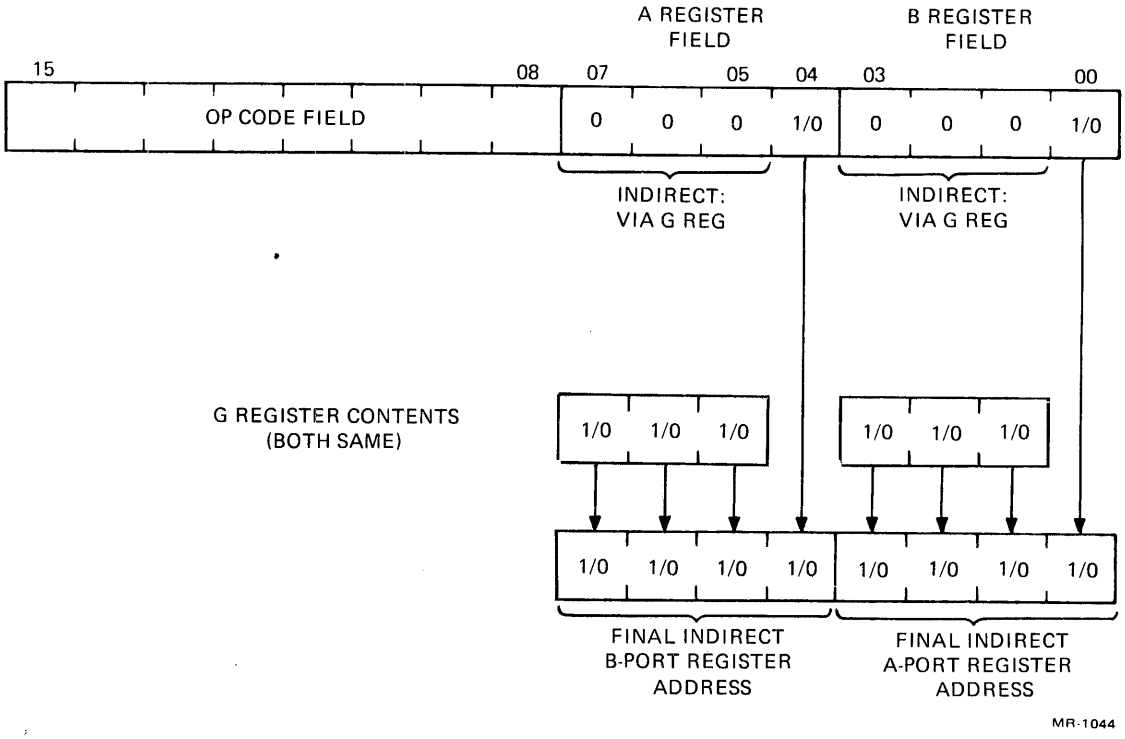
Indirect Register Addressing-Literal Instruction Format

Figure 3-12



Indirect Register Addressing-Register Instruction Format

Figure 3-13



Data may be written into specified register locations only through the A port. There are four possible data sources which lead to the A port, listed as follows:

1) ALU Output
   The output of the ALU is returned to the register file.

2) ALU Status Bit and Condition Code Flags
   The result of ALU operations as monitored by the status bit and condition code flags may be stored in a specified register.

3) WDAL <07:00>

4) WDAL <15:08>
   Since the Write access to the register file is only 8 bits wide, an operation which loads a complete word (low and high byte) from the LSI-11 system bus must require at least two micromachine cycles.

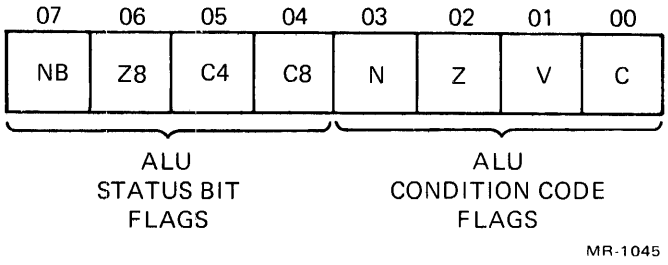
The Indirect or G register supplies the 3 highest bits of the final 4-bit indirect register address. The G register is loaded only via the Load G Low (LGL) or Input Word (IW) microinstructions which are described in the next chapter. Note that there is no mechanism which increments, decrements, or clears the G register contents.

3.3.1.3 Arithmetic Logic Unit - The microprocessor Data chip ALU accepts an 8-bit operand from both the A and B ports of the register file. Both register ports operate in Read mode to supply the operands and the ALU result can be stored in a register through the A port (in the Write mode). The ALU performs extensive logical and arithmetic operations. Arithmetic operations executed at the micromachine level may use two's complement or Binary Coded Decimal (BCD) number representation. Byte operations are executed in a single micromachine cycle while word operations require two cycles.

3.3.1.4 Status Bits and Condition Code Flags - The results of ALU operations are monitored by the status bits and condition code flags. These flags are organized in a register format and are updated after each ALU operation. Figure 3.14 illustrates the flag register organization. The condition code flags, which are also the LSI-11 PSW Flags, are updated selectively. A Register Format microinstruction will leave the condition code flags unmodified if the opcode is an even number, whereas an odd opcode microinstruction will update the flags. This feature is useful in performing intermediate data manipulations which do not affect the PDP-11 condition code flag results. The ALU status bit flags are updated after each ALU operation. Appendix A lists which Condition Code Flags are affected for each microinstruction.

Arithmetic Logic Unit Status Bit and Condition Code Flag Register

Figure 3-14



The operating rules for the condition code flags are listed below:
CONDITION CODE FLAGS

Z — This flag is set if the result of a byte or a word operation is a zero. This flag is cleared otherwise.

N — This flag is set if the most significant bit of the result of a byte or a word operation is a "1". This flag is cleared otherwise.

C — This flag monitors bits which are carried, borrowed, or shifted as follows:

Add and Increment - This flag is set if there is a carry from the most significant bit of a byte or a word operation. This flag is cleared otherwise.

Subtract and Decrement - This flag is set if there is a borrow (complement of carry) from the most significant bit of a byte or a word operation. This flag is cleared otherwise.

Shift - This flag is set if the result of a right or left shift operation causes a "1" to shift off the end of the byte or word. This flag is cleared otherwise.

Note that the C flag is affected only for operations in the areas listed above.

V — This flag is set if an arithmetic operation results in an overflow. This flag is cleared if no overflow occurs or if the operation performed is not an arithmetic operation.

The ALU status bit flags are updated after each 8-bit ALU operation and are used by the microprocessor to perform carries and borrows (for example, during a word operation). They are updated only for arithmetic, shift, test and compare microinstructions. Appendix A lists which status bit flags are affected for each microinstruction. The operating rules for the status bit flags are listed below:

ZB — This flag is set if the result of a byte or a word operation is "0". This flag is cleared otherwise.

NB — This flag is set if the most significant bit of the result of a byte or a word operation is a 1. This flag is cleared otherwise (except for SRW and SRWC).

C4 — This flag is used only for operations involving decimal arithmetic. It is set if a carry from the third bit to the fourth bit is a "1". This flag is cleared otherwise.

C8 — This flag is set if the carry from the most signficant bit is a "1" or if the result of a shift operation is to shift off a "1". This flag is cleared otherwise. Note that this status bit is not set to borrow for subtract as in the case with the C Condition Code Flag.

Each of the status bit and condition code flags shown in Figure 3.14 may be tested by a conditional jump instruction, except for the C4 status bit. In place of this bit, there is a test of the Indirect Condition Status (ICS) bit which is used to determine specific microprogram-level jump conditions. The jump on ICS microinstructions are used to directly implement the LSI-11 machine-level Branch instructions. The 16 Conditional Jump microinstructions and the Indirect Conditon Status rules are given in the next chapter.

The Status Bit/Condition Code flag register contents may be written into a register using the Copy Condition Flags (CCF) microinstruction. Conversely, the 8-bit contents of a specified register may be loaded into the flag register by the Load Condition Flags (LCF) microinstruction. In the latter case, the flags which are loaded are individually controllable.

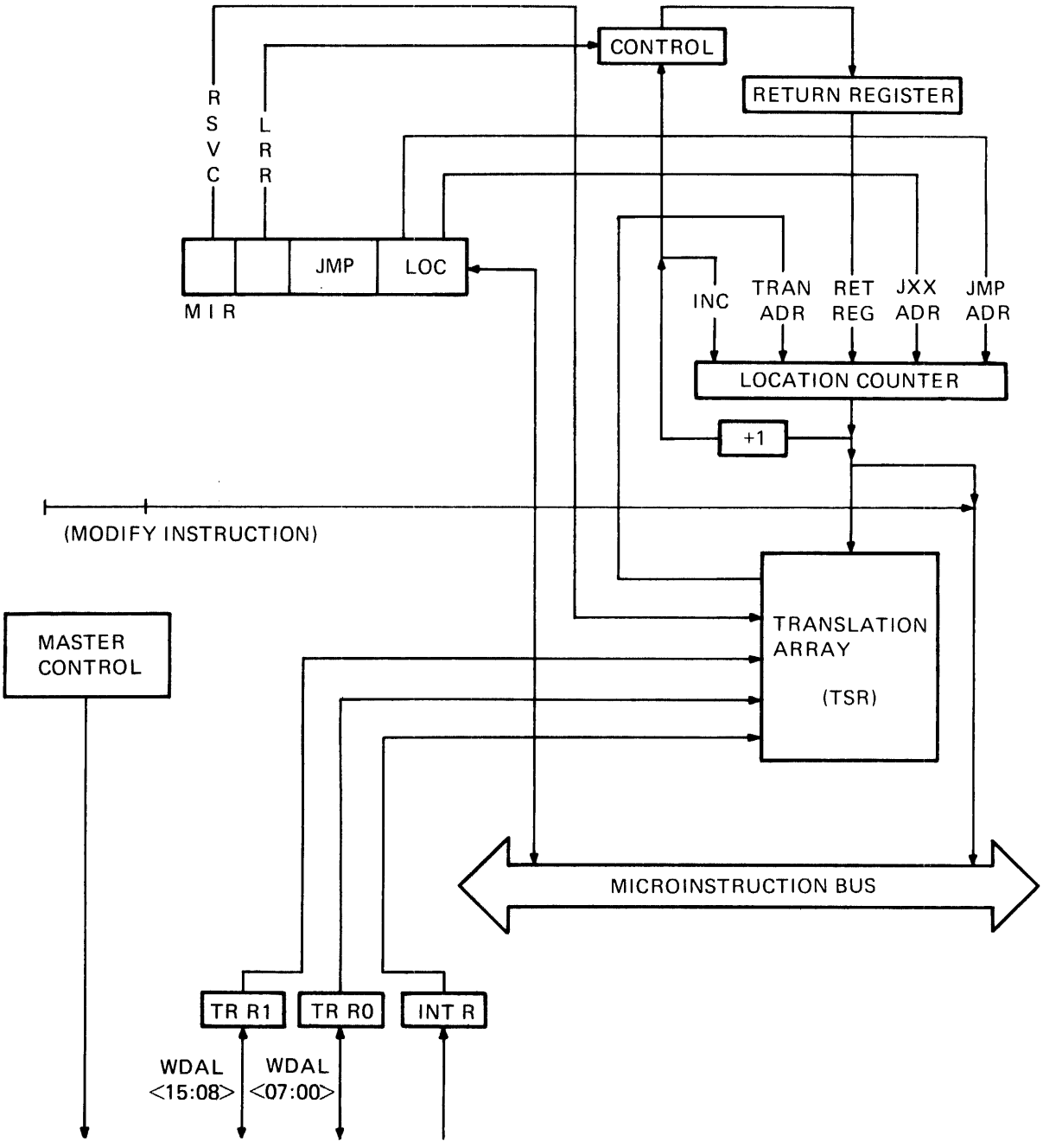
3.3.2 Microprocessor Control Chip

The microprocessor Control chip functions primarily in two areas: (1) it controls the flow of micromachine instructions which are fetched from control store (MICROM) and delivered to the Data chip for execution and (2) it administrates all LSI-11 system bus transactions by means of its connection to the Bus Transceivers and Interface Logic shown in Figure 3.15. It should be emphasized that Figure 3.15 is a functionally accurate Control chip representation, but that the actual Control chip circuitry is somewhat different. The difference arises because the data/address line inputs (WDAL <07:00> and WDAL <15:08>) which lead to the translation registers (TR R0 and TR R1) do not appear on the Control chip.

The 16-line path between the WDAL lines on the Data chip and the two translation registers on the Control chip is established by time-multiplexing the MicroInstruction Bus. The primary function of the MIB is to deliver control store addresses to the MICROMS (or to the WCS module) and to retrieve the selected microinstruction for execution. When the MIB is not occupied with its primary function, it provides the 16-line path to the translation registers. This path is established during the the execution of the Input Word microinstruction which is part of the Fetch Machine Instruction operation discussed in the previous chapter. The fetched machine instruction is loaded into the translation register(s) for examination by the translation array.

Microprocessor Control Chip Detail

Figure 3-15



3.3.2.1 Microinstruction Register - The microinstruction register shown in the Control chip detail of Figure 3.15 contains 18 bits. The additional 2 bits (not found in the Data chip microinstruction register) correspond to the LRR (Load Return Register) and RSVC (Read Next Instruction - Interrupts & Trap Service) function.

As shown in Figure 3.15, the path leading into the Return Register originates at the Location Counter incrementer. When the LRR bit (bit<16>) is set it causes a value of LC + 1 (the updated LC) to be loaded into the Return Register. This bit is part of the microinstruction at microprogram assembly time. The return register is loaded in preparation for the execution of a microprogram subroutine. Subroutine execution is terminated by the Return From Subroutine (RFS) microinstruction.

Bit <17> of the microinstruction register is a direct input to the translation array. The RSVC bit is assembled into the microinstruction sequence one location before the translation is to be invoked. This translation causes machine-micromachine control to enter the top of the interrupt/trap decision chain shown in Figure 2.16-2, which will cause all traps and interrupts to be serviced according to their priorities. If the control flow does not encounter any traps or interrupts, control proceeds to the Fetch Machine Instruction event, which causes the next machine instruction to be read into the translation registers for examination by the translation array.

3.3.2.2 Microinstruction Address Generation - Each valid microinstruction address produced by the Control chip is stored in the Location Counter. The Location Counter contents are gated onto the Micro-Instruction Bus at the appropriate phase of the micromachine cycle as the first part of the microinstruction fetch (microfetch). The Location Counter may be loaded from any one of five possible sources, as described below:

Incrementer	A value of 1 is added to the Location Counter output and written into the Location Counter. The next microinstruction is, therefore, fetched from the next sequential control store location.
Translation Array	The Translation Array, in response to its inputs, determines the location in control store from which the next microinstruction is to be fetched. The address of this location (shown as TRA ADR) is loaded into the Location Counter.
Return Register	Upon execution of the Return From Subroutine microinstruction, the contents of the Return Register are loaded into the Location Counter.
Conditional Jump Field	When the conditional jump criteria is met, the 8-bit contents of the conditional jump field are loaded into the lowest 8 bits of the Location Counter. The three highest bits are unchanged from the updated values.

Unconditional Jump Field — Upon execution of the JMP microinstruction the 11-bit contents of the unconditional jump field are loaded into the Location Counter.

A microlocation address may be determined in one additional way, by the execution of the Modify Instruction (MI) microinstruction. The Modify Instruction path which is labeled in Figure 3.15 enables the contents of a specified register pair to be ORed with the Unconditional Jump microinstruction (or any other microinstruction) on the Micro-Instruction Bus. This microinstruction is discussed in the next chapter.

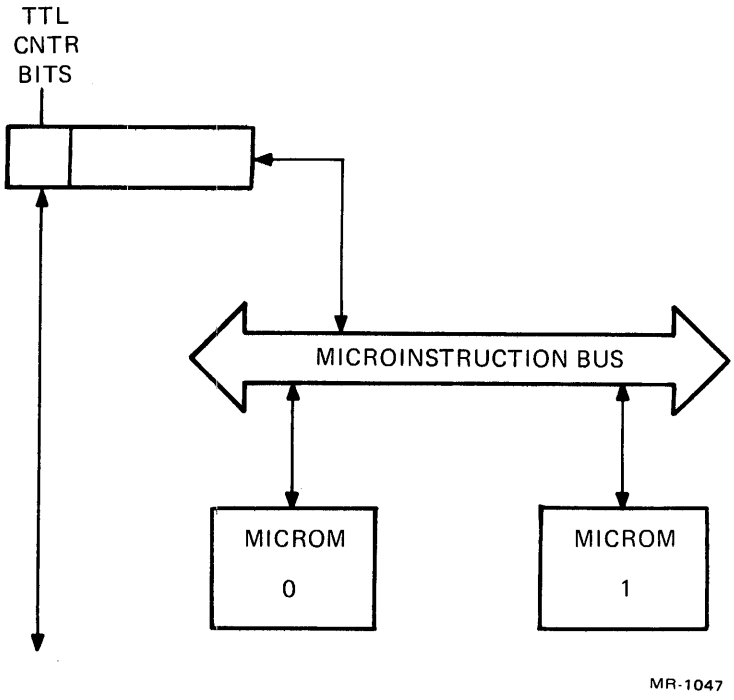
The translation array is a large combinatinal logic network which performs rapid examination of the fetched machine instruction in the Translation Register to determine where microprogram execution is to begin. The examination is completed in one machine cycle and the result is a translation address which is loaded into the Location Counter. The translation array can examine only 8 bits of a 16-bit machine instruction at one time. The translation state register, which is shown as (TSR) in Figure 3.15, determines which of the two translation registers (TR R0 or TR R1) is input to the translation array. When a new machine instruction is fetched, the TSR is reset and causes the upper byte of the instruction, which is contained in TR R1 to be examined first. This allows the LSI-11 machine instruction type to be determined before the the address modes are examined. The TSR contains 3 bits, one of which controls the translation register input, while the other two are used for purposes internal to the translation array.

The interrupt register, INT R, is composed of 3 internal interrupts and 4 external interrupts which were illustrated earlier in Figure 2.17. The external interrupts are connected to the LSI-11 system bus interface logic. The 3 internal interrupts may be set or cleared by the Set Interrupt (SI) or the Reset Interrupt (RI) microinstruction, respectively. When the translation array initiates service for a pending external interrupt, an early step in the service microprogram is to reset the interface logic flip-flop which had stored that external interrupt request. The flip-flop reset pulse is produced by the TTL Control Bits as decoded by the Special Function Logic (Figure 3.5). Figure 3.16 illustrates the Microprocessor TTL Control Bit Path as extracted from Figure 3.1. Figure 3.2 reveals that the TTL Control Bits do not actually appear in the MicroInstruction Register of either the Data chip or the Control chip. However, Figure 3.15 emphasizes the association of the TTL Control bits with the microinstructions. These bits must be assembled at the correct positions in the microinstruction flow to assure proper microprogram execution. The exact details of timing and sequencing are discussed in the following sections.

3.3.2.3 Control Signals - In addition to the 4 external interrupts, eight other signals pass between the Control chip and the LSI-11 processor interface logic. The operation of these signals are explained below and additional specific details are available in the references cited in Figure 3.4.

Microprocessor TTL Control Bit Path

Figure 3-16



The RESET line is an input to the Control chip and when active, causes microprocessor operation to be suspended. When the RESET input subsequently goes passive, the first microinstruction to be executed will be fetched from control store location 0001. The RESET line is asserted whenever the power supply signal BDCOK H goes passive or when a bus timeout error occurs. The microprogram executed after the RESET line goes passive is flow-charted in Figure 2.16-2.

The COMPUTE signal is sampled during PH 3 of the micromachine cycle. This line is maintained in an active high state, and is used during manufacturing for test purposes.

The remaining signals are concerned with the control chip's administration of the three types of LSI-11 system bus I/O transfers. Programmed I/O, Interrupt-Driven I/O, and DMA I/O. These signals are interfaced to the LSI-11 bus system by the Bus I/O Control Signal Logic described in The Microcomputer Handbook.

The transition of the SYNC signal to its active state indicates that the address data on WDAL <15:00> is valid. The sync signal remains asserted until the I/O transfer is completed. Additional interface circuitry assures that the corresponding system bus signal BSYNC L remains active until after the bus slave device terminates its reply signal.

The REPLY signal originates in the addressed memory or I/O device and informs the Control chip that the I/O operation should be continued. Specifically, this signal must be asserted during PH 3 of the micromachine cycle while an Input or an Output microinstruction is being executed. The state of the reply signal is also interrogated before a Read or Write operation is initiated. This is to assure that a previously-addressed device has completely released the system bus.

The DIN (Data-In) signal is asserted by the Control chip after the address information is removed from WDAL <15:00>, or one micromachine cycle after the SYNC signal is asserted. The DIN signal returns to the passive state at the completion of the Input Byte (IB) or Input Word (IW) microinstruction, or when SYNC is made passive. This signal causes the addressed device to place its data on BDAL<15:00> for input to the processor.

The DOUT (Data-Out) signal is asserted by the Control chip when output data is placed on WDAL <15:00> by the Data chip and remains asserted until the Output microinstruction is completed.

The WB (Write/Byte) signal is asserted when the address information is placed on WDAL <15:00> to indicate that a Write operation follows. If it remains active during the Data-Out portion, a Byte operation (DATOB) is signified.

The BUSY signal is sampled during PH 3 of the micromachine cycle during the first cycle of a Read or Write microinstruction. If the BUSY signal is asserted, the microprocessor enters a wait state and suspends operation until BUSY goes passive. This mechanism is used by the direct memory access interface logic, so an I/O device on the LSI-11 bus can gain control of the system bus.

The Interrupt Acknowledge (WIAK) signal is asserted by the Control chip along with the SYNC line when a Read Acknowledge or Write Acknowledge microinstruction is executed. The WIAK signal indicates that the microprocessor is responding to an interrupt. The LSI-11 interface logic delays the appearance of WIAK as BIACK H until BDIN L is asserted. This is to allow the BDIN L signal to stabilize priorities between the two possible requests in the interrupting device (see The Microcomputer Handbook).

3.4 MICROMACHINE OPERATION

All micromachine operations are controlled by the four-phases of the micromachine clock. Each phase has a nominal period of 95 nanoseconds giving a cycle period of 380 nanoseconds. The clock phases are distributed to the Control, Data, and MICROM chips of the microprocessor as MOS-level non-overlapping pulses (RPH <4:1>). True and complement TTL clock phase pulses are distributed to the Bus Transcievers and Interface Logic circuitry for synchronization with the micromachine.

Micromachine operating speed is maximized by the use of pipelining techniques to determine microlocation addresses and to access microinstructions for execution. The pipeline techniques are implemented in the context of a time-multiplexed MicroInstruction Bus, which reduces inter-chip wiring.

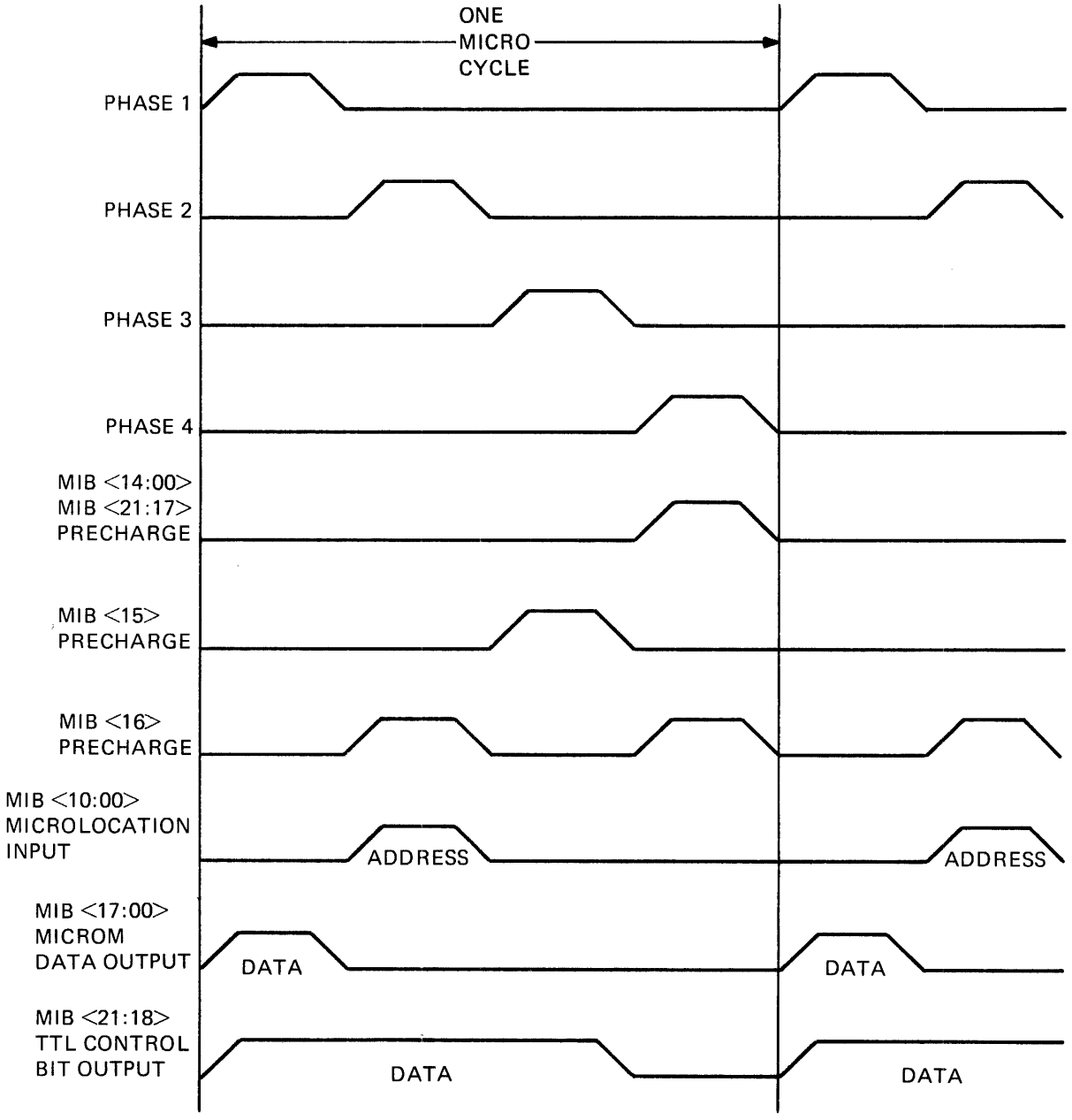
3.4.1 Microinstruction Bus Data Transfer

The data transfer method employed on the microinstruction bus is a precharge-conditional discharge technique which is compatible with LSI MOS memory and microprocessor devices. Data is transmitted on the microinstruction bus in logical complement form (the lower voltage represents a Logical 1). Each microinstruction bus line is unconditionally precharged high at one phase and selectively discharged at a later phase. The receiving device senses the discharged state during the appropriate phase and interprets the lower voltage as a logical 1. All precharging is performed by the MICROMs. The receiving device may be the Control chip, the Data chip, or a MICROM. In the case of a microinstruction fetch, the Control and Data chips receive the microinstruction simultaneously, each chip storing the needed portions in its microinstruction register.

3.4.1.1 Control Store (MICROM) Microinstruction Bus Cycle - The MICROM operations as a function of micromachine clock phases are illustrated in Figure 3.17. The Location Counter contents are gated onto the microinstruction bus by the Control Chip during PH 2 and remain valid during PH 3. All microinstruction bus lines are then unconditionally precharged at PH 4 (by the MICROMs) with the exception of MIB <15>, which is precharged at PH 3. mIB <16> was also precharged at PH 2 in addition to being precharged at PH 4.

MICROM Access Microinstruction Bus Cycle

Figure 3-17



During the execution of a microinstruction which requires only one micromachine cycle, the contents of the microlocation addressed during PH 2 and PH 3 are returned to the Data and Control chips during PH 1 of the following micromachine cycle. However, if the control chip discharges MIB 16 at PH 3, the selected MICROM will not conditionally discharge MIB <15:00> and MIB <21:18> during the following PH 1. This circumstance occurs during the second cycle of a two-cycle microinstruction, or in response to a WAIT state. Figure 3.18 illustrates the disabling of the MICROM during the first cycle, resulting in a delay of one micromachine cycle.

As shown in the Figures, the TTL Control Bits are valid beginning with PH 1 throughout PH 3. The LSI-11 external circuitry latches the TTL Control Bits during PH 3.

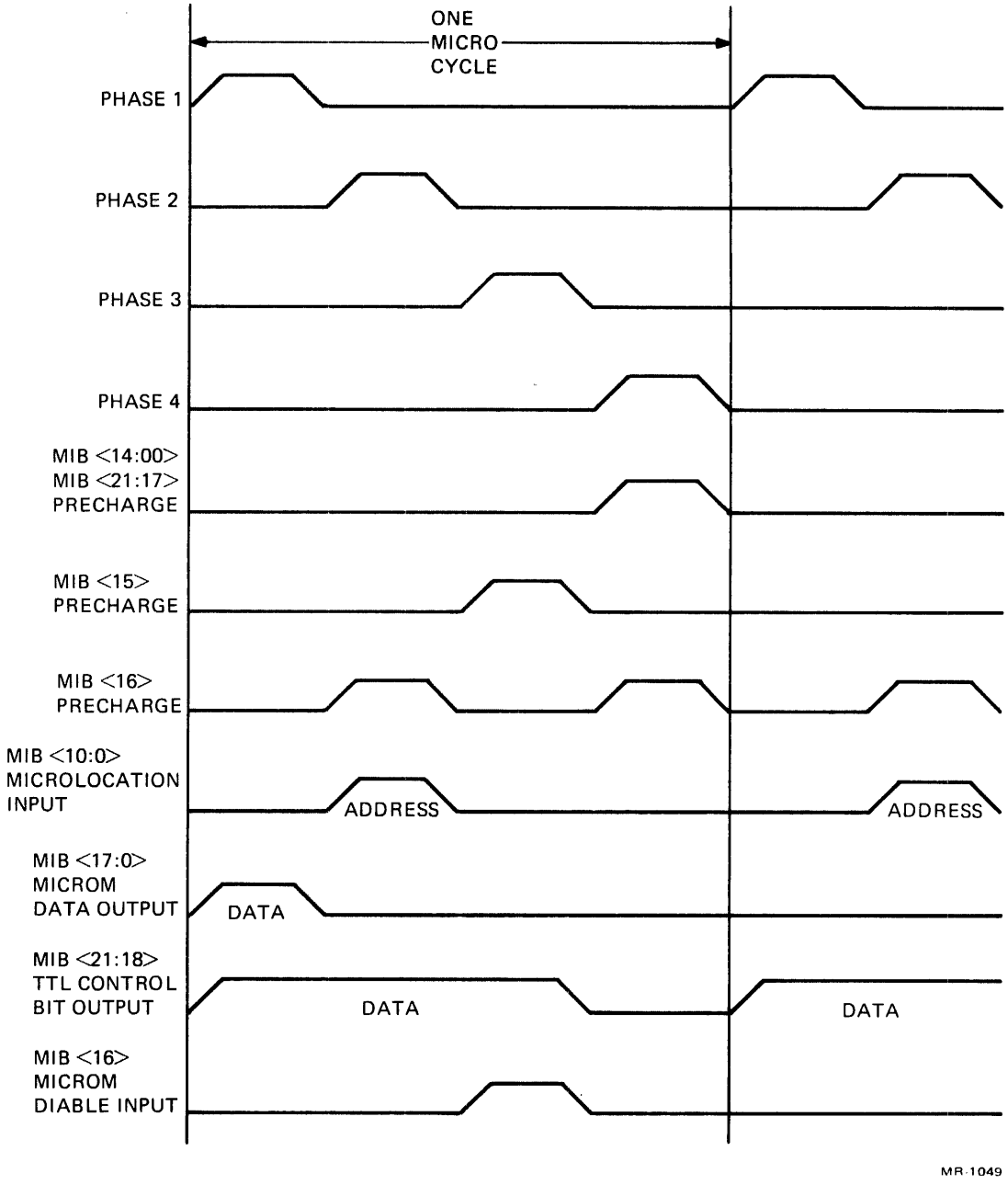
3.4.1.2 Control Chip Microinstruction Bus Cycle - The Control chip determines the micromachine instruction flow by selectively loading the Location Counter. The basic micromachine cycle of Control chip operations is shown in Figure 3.19. The Location Counter is loaded with a new value during PH 1. This value can originate from 1 of 5 sources as discussed in Section 3.3.2.2. The contents of the Location Counter are output onto the microinstructin bus at the beginning of PH 2 and remain valid through PH 3. Also during these two phases, the translation array is processing its inputs to see if a translation is required. During PH 4 the translation address becomes ready to load into the Location Counter and the translation state register may be loaded.

An example of Control chip operation effected by the WAIT state is shown in Figure 3.20. Events proceed normally until PH 3 when the Control chip determines a WAIT state. This determination is the result of sampling the BUSY and REPLY lines during PH 3. The Control chip response to the WAIT state is to discharge MIB<16> during PH 3 to disable the MICROMs response and to assert the WAIT line during PH 4, which prevents the Data chip from loading a new microinstruction. Because of the WAIT state, the Location Counter value remains unchanged during the second micromachine cycle and the translation array processing produces the same output which may be a conditional loading of the translation state register or a translation address. Since the WAIT state was not re-established during the second micromachine cycle, a new Location Counter value will be loaded during the following PH 1 and a new microinstruction will be fetched from the selected MICROM.

The execution of a two-cycle microinstruction produces a Control chip sequence similar to the above. However, since both the Control chip and the Data chip independently recognize two-cycle machine instructions, the WAIT signal is unneccessary and therefore unasserted at PH 4.

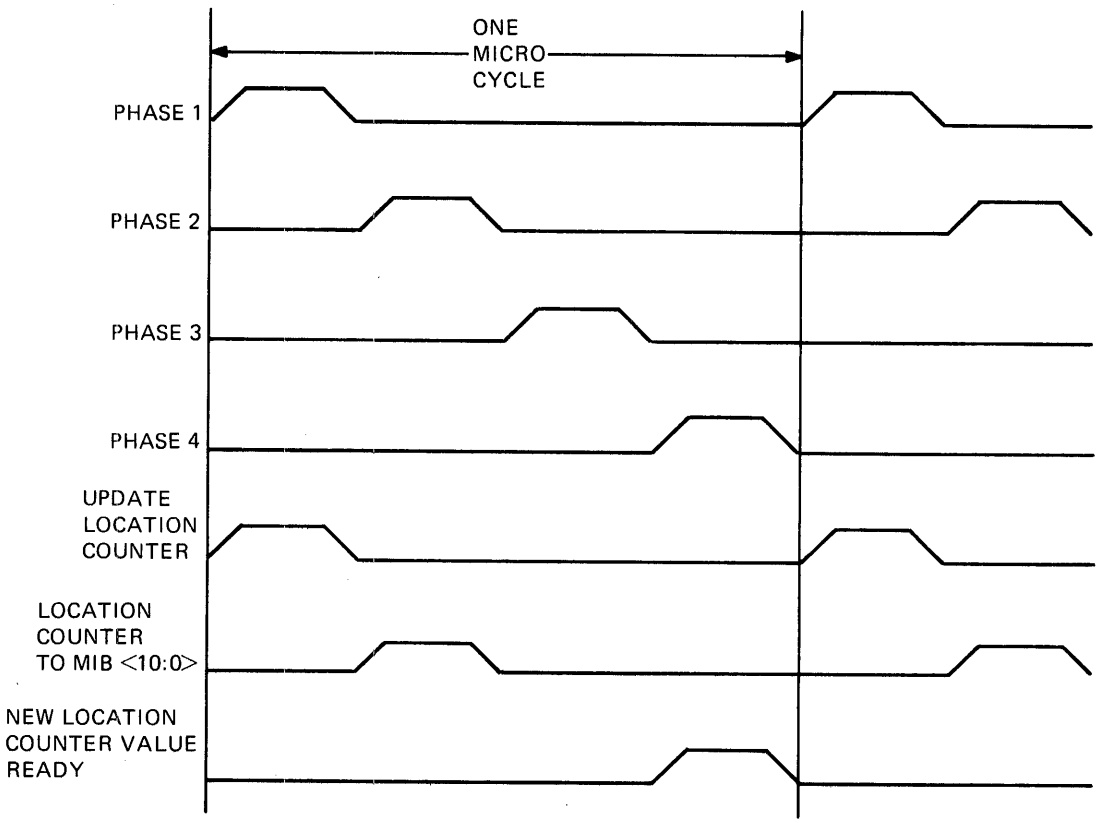
MICROM Access Microinstruction Bus Cycle (Disabled)

Figure 3-18



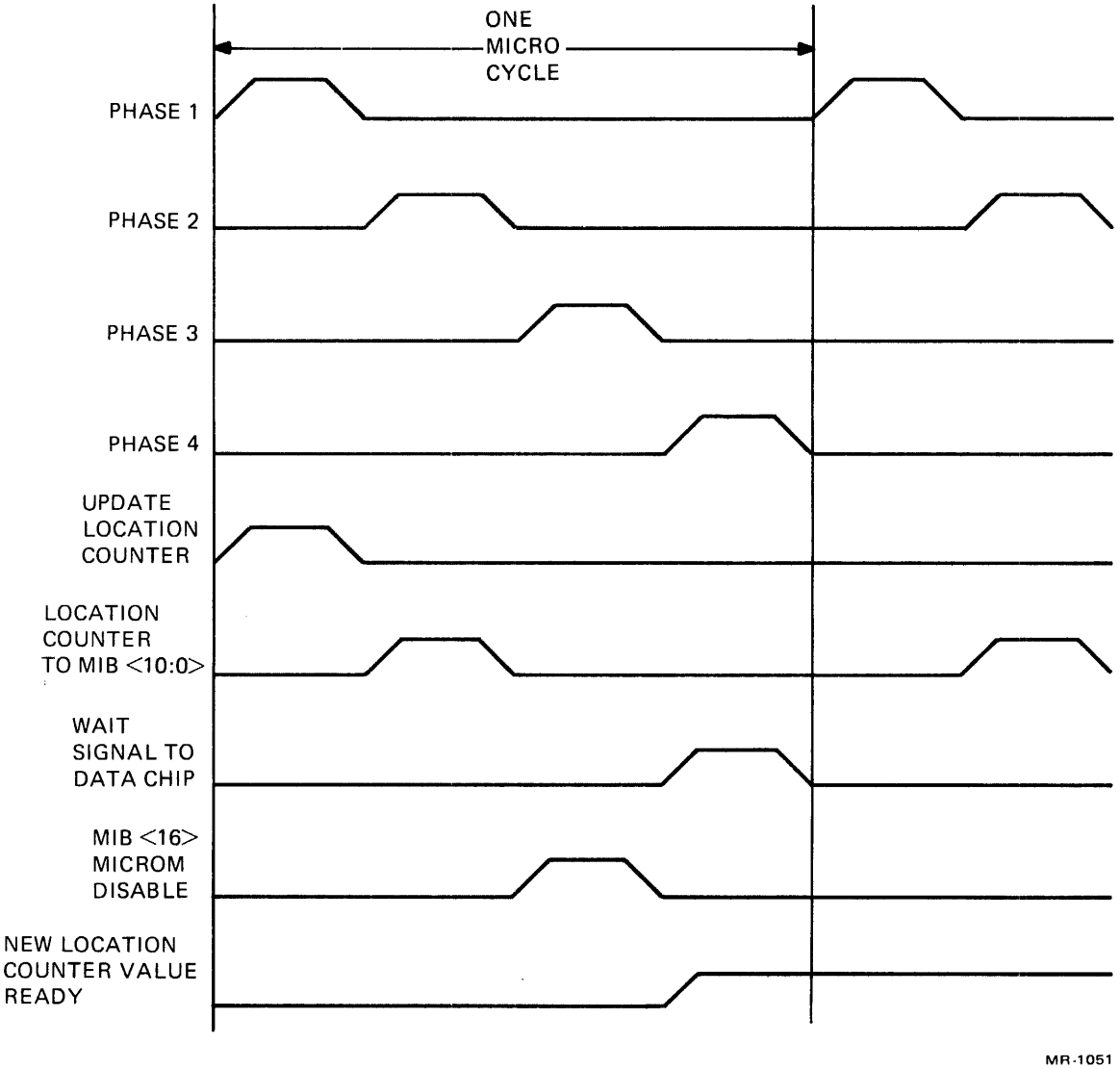
Control Chip Single-Cycle Microinstruction Bus Cycle

Figure 3-19



Control Chip Wait Response

Figure 3-20



3.4.1.3 Data Chip Microinstruction Bus Cycle - The microprocessor Data chip is completely controlled by the microinstruction in its microinstruction register and by the WAIT line which originates in the Control chip. The basic micromachine single-cycle sequence is shown in Figure 3.21. During PH 1 the microinstruction contents placed on MIB<15:00> by the selected MICROM are loaded into the Data chip microinstruction register. The register fields are decoded during PH 2, the selected registers are accessed during PH 3 and the ALU processes the operands input to it during PH 4. The ALU result is written into the register file via the A port on the next PH 1. The status bits and condition code flags which monitor the ALU result are updated during PH 2.

When a two-cycle Data Manipulation microinstruction is executed, the Data chip retains its microinstruction register contents during the second cycle as shown in Figure 3.22. The low bit of each register field is complemented during PH 1 of the second cycle which allows the second cycle of a word microinstruction to be processed in the ALU.

All Jump microinstructions require two micromachine cycles for execution because the Location Counter contents cannot be loaded with a new value until PH 1 of the second cycle.

A Conditional Jump microinstruction examines the flags which have settled during PH 2 and sends the result to the Control chip via MIB<15> during PH 4. If the result is true, the 8-bit address will be loaded into Location Counter bits <7:0> during the following PH 1.

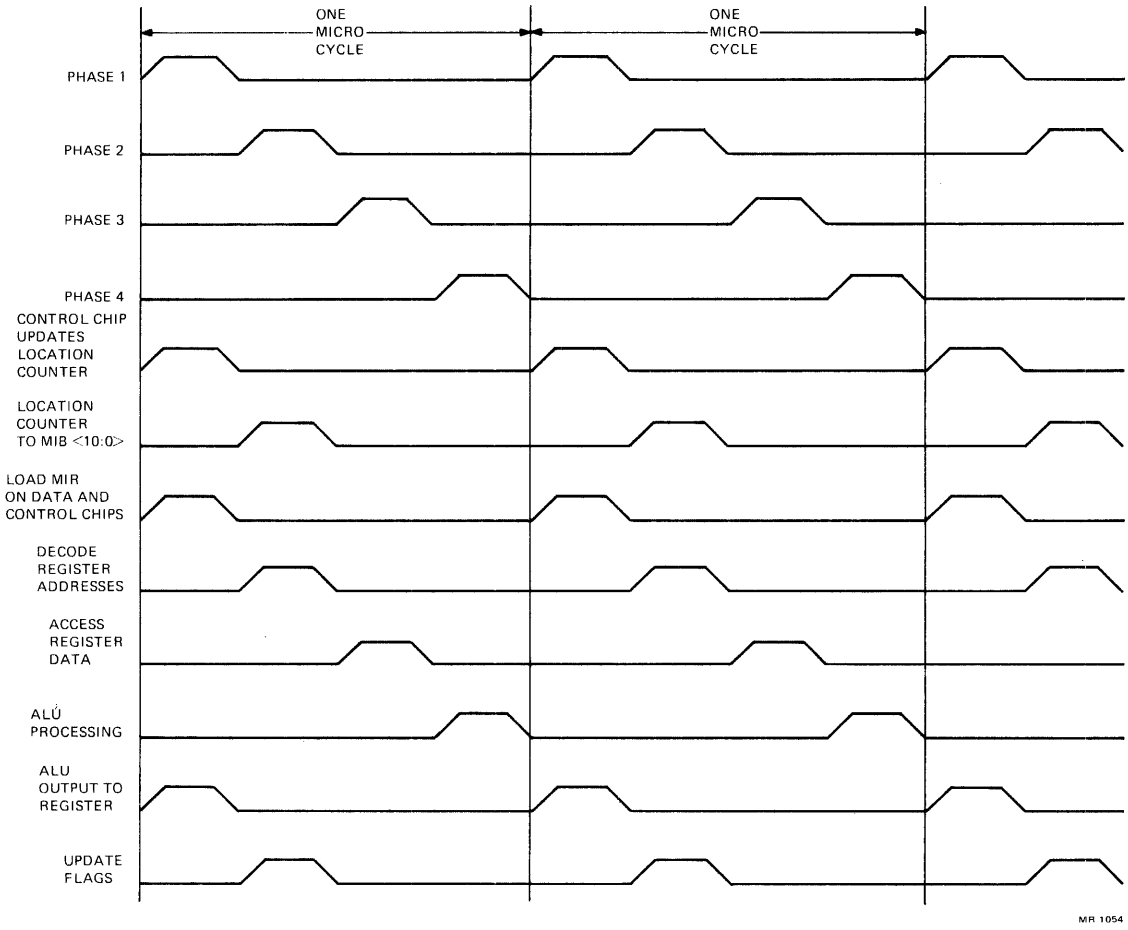
Normal operation of the Data chip is suspended if the Control chip asserts the WAIT signal during PH 4 of any micromachine cycle.

3.4.1.4 Complete Micromachine Cycle - The inter-relationship between the cycles of the Control, Data and MICROM chips is illustrated in Figure 3.23 for the case of a single-cycle microinstruction. The operation sequences listed for each chip are repetitive, but only the events related to the execution of a single microinstruction are shown in the figure. The complete sequence begins with a new value being loaded into the Location Counter during PH 1 of the first micromachine cycle and ends when the status bit and condition code flags are updated during PH 2 of the third micromachine cycle. An accurate representation of micromachine operations can be constructed by combining the individual chip operation sequences discussed previously. The sequences chosen depend upon the microinstruction flow being represented.

The Execution of the Input Word microinstruction changes the function of the microinstruction bus during its second cycle. Since PH 1 of the second cycle is not used to update the microinstrucion registers with new contents, the Data chip uses MIB<15:00> to send the fetched LSI-11 machine instruction to the translation register (for subsequent examination by the transalaion array). This is necessary because the Control chip has no direct connection to the WDAL lines.

Complete Micromachine Cycle

Figure 3-23



No special register addressing techniques are needed in such cases and the microinstruction can complete execution in a minimum of one microcycle.

4.2.4.2 Word Operand Formation - The microinstruction register fields, MI<7:4> and MI<3:0>, are interpreted as shown below to form 16 bit operands:

All word operations other than right shifts:

A Register Field - (A) Even	Normal Use
A Register Field - (A) Odd	Bytes Reversed
B Register Field - (B) Even	Normal Use
B Register Field - (B) Odd	Sign Extension

Right shift word operations:

A Register Field - (A) Odd	Normal Use
A Register Field - (A) Even	Not Recommended
B Register Field - (B) Odd	Normal Use
B Register Field - (B) Even	Not Recommended

Normal Use: A and B Register Field argument is even.

Word operand processing will normally be performed when both the A and B field register arguments are even. During the first microcycle of a word microinstruction, the 2 register fields designate the 8-bit operands. During the second microcycle the low-order bit of each of the register fields is logically complemented, thus pointing to the next register locations. For example, if A and B are 00 and 10 (octal) during the first microcycle, the effective arguments become 01 and 11 during the second microcycle. This case is illustrated in Figure 4-6.

The register field assembly arguments reflect this convention. The bus address register symbol, RBA is equivalent to 12. To perform an operation on the lower byte of this register, the symbol RBAL would be used, which also assembles as 12. RBAH, the corresponding high byte assembles as 13.

If the B field argument is assembled with the low-order bit equal to 1, the B field contents point to the first byte to be processed. During the second cycle, however, the high byte is formed by extending the high-order bit of the first or low byte through 8 bit positions. This case is illustrated in Figure 4-7.

The sign extension procedure presented above does NOT apply in the case of an A register field odd argument. Instead, the low-order bit of the A field is complemented for the second cycle, and the byte register accessed is the Next Lower byte. Thus the word operand input via the A field is byte-swapped, as illustrated in Figure 4-8.

3.5 MICROPROGRAMMING THE BASIC LSI-11 MACHINE CYCLE

The essential characteristics of the microprograms which emulate the LSI-11 microcomputer architecture will be discussed for two simple cases. The first case is the basic two-event machine cycle and the second case is the basic machine cycle including external interrupts and bus error traps.

3.5.1 Fetch-Execute Machine Cycle Microprogramming

The basic LSI-11 machine cycle, repeated here as Figure 3.24, consists of a DATI operation followed by a microprogrammed routine which interprets and executes the fetched machine instruction. The DATI operation is itself microprogrammed as a Read/Input sequence which is discussed in Chapter 5. The specific Read microinstruction employed is Read And Increment Word By 2 (RIW2). This microinstruction places the contents of the register pair designated within it, PC H and PC L on BDAL L <15:00> and initiates a Data-In system bus cycle. It subsequently increments the word contained in PC H and PC L by 2, causing the Program Counter to point to the next address from which a machine instruction is to be fetched. The specific Input microinstruction used, which comprises the second half of the Read/Input sequence, is Input Word. This microinstruction places the fetched machine instruction into the instruction register RIR L and RIR H and also loads it into the Translation Registers TR R0 and TR R1. Note that the lower byte of the machine instruction is put in RIR H and the high byte is put in RIR L.

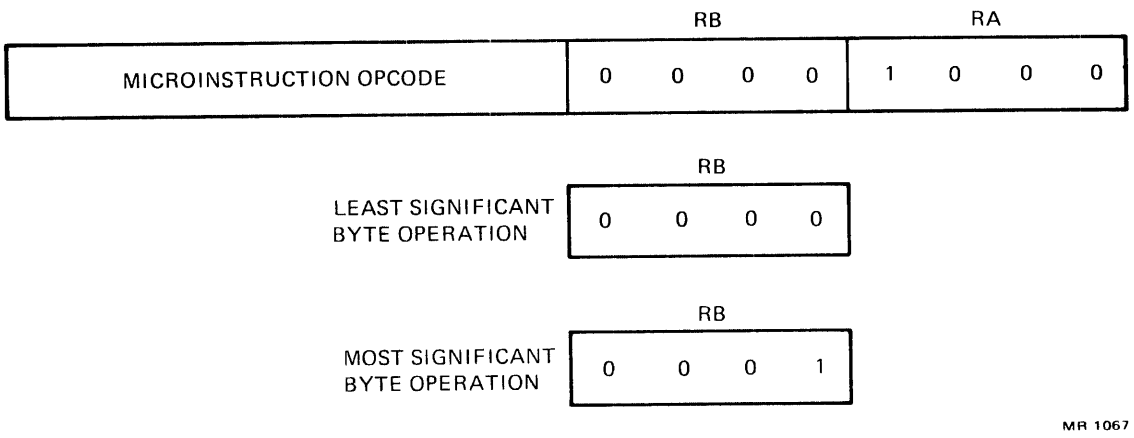
The Execute Machine Instruction portion of the basic machine cycle begins with the examination of the Translation Register contents by the translation array. The translation array output, which is available after one micromachine cycle, produces the address of the control store or MICROM location which contains the first instruction of the microprogrammed routine which will execute the machine instruction. The translation array may be used one, two or three times during the Execute Machine Instruction operation, depending upon the type of instruction being executed. In preparing the ALU operands, zero, one, or several DATI operations are performed, depending upon the addressing modes used in the machine instruction source and destination fields.

3.5.2 Fetch-Execute-Trap & Interrupt Machine Cycle Microprogramming

The basic machine cycle of Figure 3.24 is expanded in Figure 3.25 to include external interrupts and the single and double bus error traps. As explained in Chapter 2, a bus error occurs when an addressed device does not respond within 10 microseconds. This leaves the Read/Input sequence of the DATI operation uncompleted and the Bus Interface Logic asserts the Control chip reset line. As a result, the Location Counter is loaded with 0001 and a microprogrammed bus error recovery routine is executed. Two DATO cycles are performed to push the PC and PSW contents onto the stack, followed by two DATI cycles which load new values into the program counter and processor status word registers.

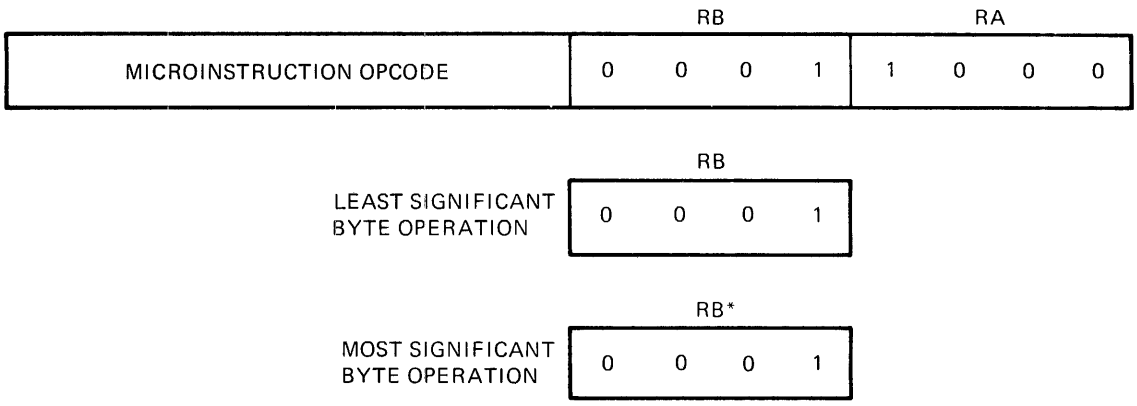
Microinstruction with Even RB

Figure 4-6



Microinstruction with Odd RB

Figure 4-7

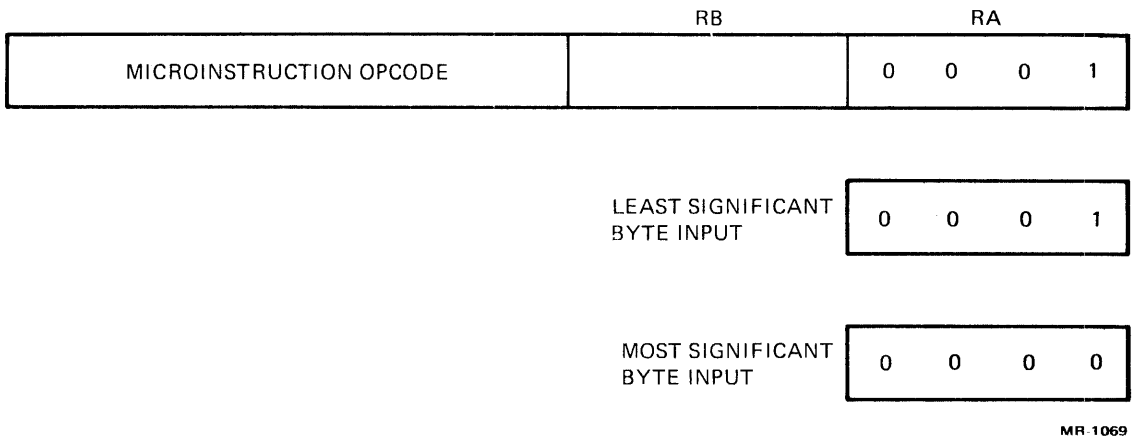


* THE MOST SIGNIFICANT BIT IS USED FOR ALL 8 BITS OF THE RB ARGUMENT.

MR 1068

Register Microinstruction with Odd RA Input

Figure 4-8



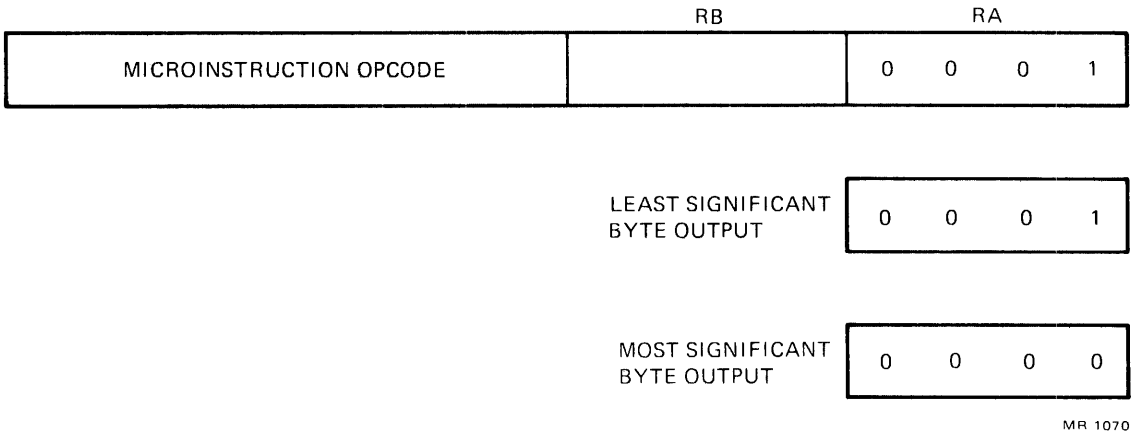
Since the A field also determines the destination of the ALU output, the abnormal reversed register coding sequence is apparent in the final register file contents. This case is illustrated in Figure 4-9, which shows that the low-order 8-bits of the word result are deposited in the high or odd byte and the high-order 8-bits are deposited in the low byte.

Right Shift Word Operations

Since all Right Shift Word operations begin with the left-most part of the word operand, both the A and B field argument must be odd. Left Shift Word operations follow the Normal Use conventions above.

Register Microinstruction with Odd RA Output

Figure 4-9



4.3 MICROINSTRUCTION SET FUNCTIONAL ORGANIZATION

Because of the large number of microinstructions in the microprocessor repetroire, it is useful to establish a system of organization along functional lines. The three major functional classifications are Data Manipulation, Data Access, and Microprogram Control. Subclassifications of microinstructions exhibit variations of a basic operation. For example, under the Data Manipulation classification there is the general subclassification "Move".

The final description of a microinstruction is determined by (1) whether it is a byte or word microinstruction, (2) whether the indicated operation is performed conditionally or unconditionally, and (3) whether it updates the ALU condition code flags (N, Z, V, C).

4.3.1 Data Manipulation Microinstructions

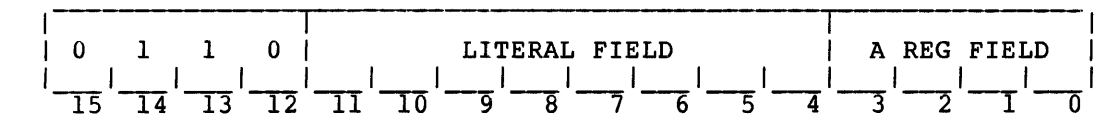
The microinstructions in this group provide the bulk of the micromachine data manipulation ability. The subclassifications here are: Move, Increment/Decrement, Logical, Shift, and Arithmetic. The operations performed by all microinstructions in this classification are completed within the micromachine and do not require I/O. These microinstructions affect microprogram control only in that they update the ALU status bit and condition code flags and that some are executed conditionally.

4.3.1.1 Move Microinstructions - The common feature of the Move Microinstructions is that they move data between locations within the micromachine. These locations are exclusively within the microprocessor Data and Control chips, with the exception of the Load Literal microinstruction which moves a constant contained in microinstruction to a designated register.

The Microinstruction descriptions are as follows:

LL

LOAD LITERAL



OPCODE: 06XXXX
MICROCYCLES: 1
OPERATION: (RA) <-- (LITERAL FIELD)
DESCRIPTION: The 8-bit contents of the literal field, MI<11:4>, are loaded into register RA. The NB and ZB status bit flags are updated. The contents of the literal field are not changed.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

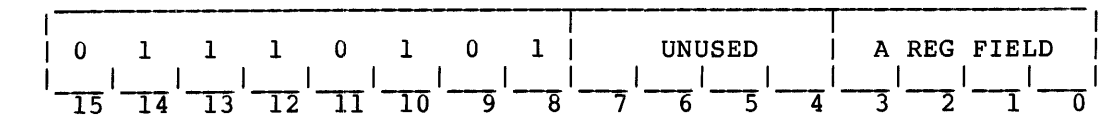
ASSEMBLY MNEMONIC: LL 200,RSRCH ;MOVE 200 TO RSRCH

ASSEMBLED OCTAL: 064005

BEFORE								AFTER							
(RSRCH) = 000								(RSRCH) = 200							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

LGL

LOAD G LOW



OPCODE: 072400-072777
MICROCYCLES: 1
OPERATION: (G REGISTER) <-- RA<2:0>
DESCRIPTION: The three lowest-order bits of RA are loaded into the G register. These 3 bits are subsequently used in indirect register addressing operations. With the exception of the INPUT WORD microinstruction, the LGL microinstruction provides the only means for controlling the G register contents. The contents of RA are unchanged and the flag register contents are not affected.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: LGL RIRL ;LOAD THE 6 REGISTER FROM RIRL
ASSEMBLED OCTAL: 072410

BEFORE

(RIRL) = 377
(G) = 0

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	1

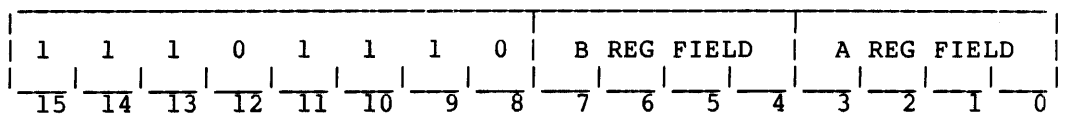
AFTER

(RIRL) = 377
(G) = 7

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	1

LTR

LOAD TRANSLATION REGISTER



OPCODE: 167000-167377
MICROCYCLES: 1
OPERATION: (TRANSLATION REGISTER) <-- (RB:RA)

DESCRIPTION: The 16-bit contents of RB:RA are loaded into the translation register. The contents of RB and RA are unchanged and the flag register contents are unaffected.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: LTR RIRH,RIRL ;LOAD TRANSLATION REGISTER

ASSEMBLED OCTAL: 071424

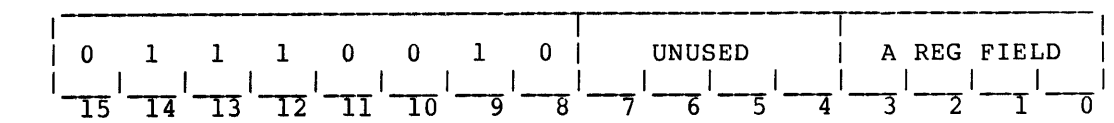
BEFORE	AFTER
(RIR) = 100200	(RIR) = 100200
(TR) = 000000	(RIR) = 100200

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0

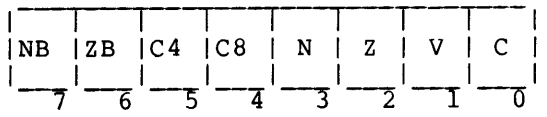
CCF

COPY CONDITION FLAGS



OPCODE: 071000-071377
MICROCYCLES: 1
OPERATION: (RA) <-- (FLAG REGISTER)
DESCRIPTION: The 8-bit contents of the ALU status bit and condition code flag register are placed in RA. The flag register contents are unaffected by this operation.

The ALU flags and condition codes are copied in the following format:



STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: CCF RDSTL ;COPY FLAGS TO RDSTL

ASSEMBLED OCTAL: 071006

BEFORE

(RDSTL) = 000

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	0	0	0	0

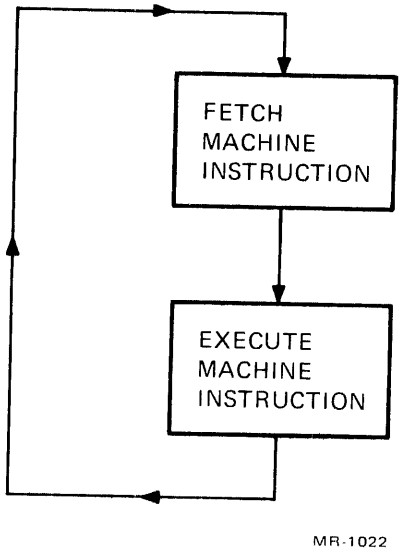
AFTER

(RDSTL) = 200

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	0	0	0	0

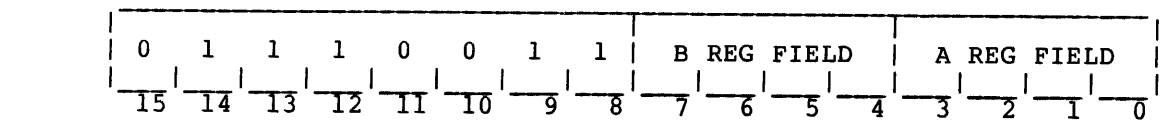
Basic LSI-11 Machine Cycle

Figure 3-24



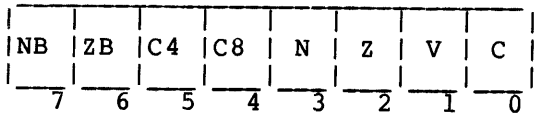
LCF

LOAD CONDITION FLAGS



OPCODE: 071400-071777
MICROCYCLES: 1
OPERATION: (FLAG REGISTER) <-- (RA)
DESCRIPTION: The 8-bit contents of RA are selectively loaded into the ALU flag register. Microinstruction bits 4, 5, 6, and 7 control the loading of the C, V, Z, and N condition code flags, respectively. The contents of RA are not affected.

The ALU flag register format is loaded in the following format:



STATUS BITS: NB: loaded from RA<7>, unconditionally
ZB: loaded from RA<6>, unconditionally
C4: loaded from RA<5>, unconditionally
C8: loaded from RA<4>, unconditionally

CONDITION CODES: N: loaded from RA<3>, when (MI<7>) = 1
Z: loaded from RA<2>, when (MI<6>) = 1
V: loaded from RA<1>, when (MI<5>) = 1
C: loaded from RA<0>, when (MI<4>) = 1

EXAMPLE:

ASSEMBLY MNEMONIC: LCF RDSTL ;LOAD FLAGS FROM RDSTL

ASSEMBLED OCTAL: 071424

BEFORE

(RDSTL) = 017

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0

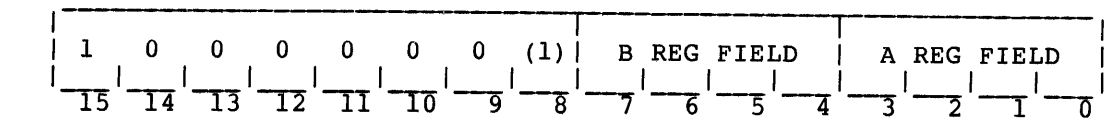
AFTER

(RDSTL) = 017

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	0	0	0	1

MB (MBF)

MOVE BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 100000-100377 (100400-100777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)
DESCRIPTION: The 8-bit contents of RB are placed in RA. The contents of RB are not affected. The NB and ZB status bit flags are updated. MBF causes the N and Z condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (MBF ONLY)
N: set if byte result < 0; cleared otherwise
Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: MB RDSTL,RSRCL ;MOVE BYTE FROM RDSTL TO RSRCL

ASSEMBLED OCTAL: 100344

BEFORE

(RDSTL) = 000
(RSRCL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

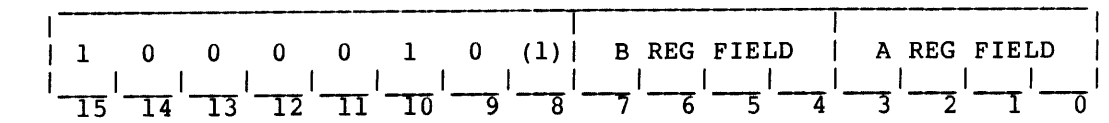
AFTER

(RDSTL) = 000
(RSRCL) = 000

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	0	0	0

CMB (CMBF)

CONDITIONALLY MOVE BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 102000-102377 (102400-102777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB), IF C FLAG = 1
DESCRIPTION: The 8-bit contents of RB are placed in RA, if the C flag was set = 1 by a previous operation. The contents of RB are not affected. The NB and ZB status bit flags are updated. CMBF causes the N and Z condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (CMBF ONLY)
N: set if byte result < 0; cleared otherwise
Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

NOTE: The status bit (and condition code) flags are updated only if the C flag was initially set = 1.

EXAMPLE:

ASSEMBLY MNEMONIC:
```
CMB   RDSTL,RSRCL   ;IF C = 1, MOVE BYTE
                    ;FROM RDSTL TO RSRCL
```

ASSEMBLED OCTAL: 102144

BEFORE

(RDSTL) = 000
(RSRCL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

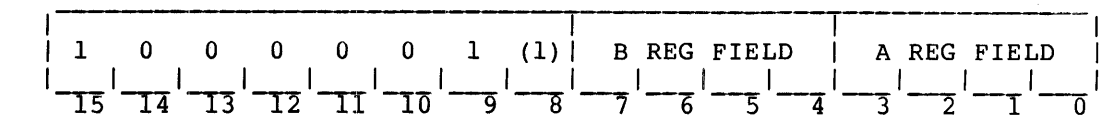
AFTER

(RDSTL) = 000
(RSRCL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

MW (MWF)

MOVE WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 101000-101377 (101400-101777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)
DESCRIPTION: The 16-bit contents of (RB+1:RB) are placed in (RA+1:RA) when B is even. When B is odd, the contents of RB are placed in RA and the high-order bit of RB replaces the 8 bits of (RA+1). The contents of (RB+1:RB) are not affected. The NB and ZB status bit flags are updated. MWF causes the N and Z condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if word result < 0; cleared otherwise
(MWF ONLY) Z: set if word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: MWF RDST,RSRC ;MOVE WORD FROM RDST TO RSRC

ASSEMBLED OCTAL: 101544

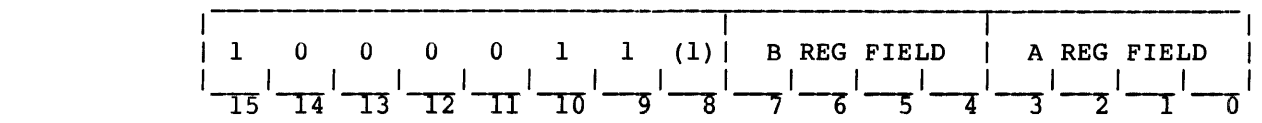
BEFORE	AFTER
(RDST) = 000000	(RDST) = 000000
(RSRC) = 177777	(RSRC) = 000000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	1	0	0

CMW (CMWF)

CONDITIONALLY MOVE WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 103000-103377 (103400-103777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB), IF C FLAG = 1
DESCRIPTION: The 8-bit contents of RB+1:RB are placed in RA+1:RA if the C flag was set = 1 by a previous operation when B is even. When B is odd, the contents of RB are placed in RA and the high order bit of RB replaces the 8 bits of RA+1 if the C flag was set=1 by a previous operation. The contents of RB+1:RB are not affected. The NB and ZB status bit flags are updated. CMWF causes the N and Z condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if word result < 0; cleared otherwise
(CMWF ONLY) Z: set if word result = 0; cleared otherwise
V: cleared
C: not affected

NOTE: the status bit (and condition code) flags are updated only if the C flag was initially set = 1.

EXAMPLE:

```
ASSEMBLY MNEMONIC:  CMWF  RDST,RSRC   ;IF C = 1, MOVE WORD
                                      ;FROM RDST TO RSRC

ASSEMBLED OCTAL:    103444
```

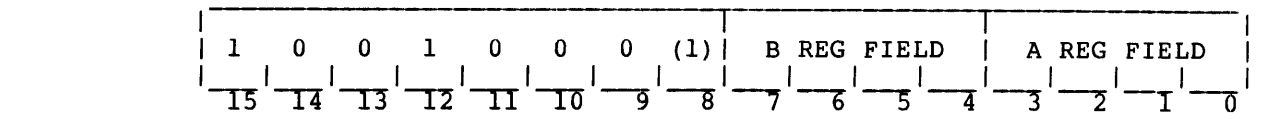
BEFORE		AFTER	
(RDST) = 177777		(RDST) = 177777	
(RSRC) = 000000		(RSRCL) = 177777	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		1 0 0 0 1 0 0 0	

4.3.1.2 Increment / Decrement Microinstructions - The Increment / Decrement Microinstructions operate only on register contents. Note that there is no means of incrementing or decrementing the G register contents.

The microinstruction descriptions are as follows:

ICB1 (ICB1F)

INCREMENT BYTE BY 1 (UPDATE CONDITION CODE FLAGS)



OPCODE: 110000-110377 (110400-110777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)+1
DESCRIPTION: The sum of the 8-bit contents of RB plus 1 is placed in RA. The contents of RB are unchanged. All status bit flags, except C4, are updated. ICB1F causes the N, Z, and C condition flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: ICB1F (ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

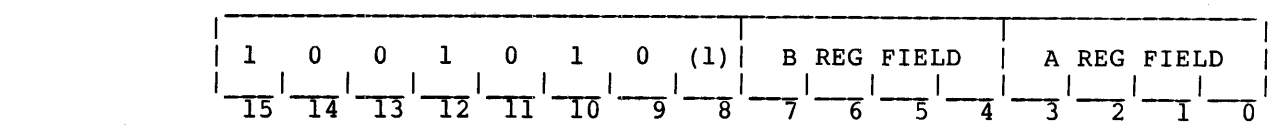
ASSEMBLY MNEMONIC: ICB1 RDSTL,RSRCL ;ADD 1 TO RDSTL, SUM IN RSRCL

ASSEMBLED OCTAL: 110144

BEFORE								AFTER							
(RDSTL) = 002								(RDSTL) = 002							
(RSRCL) = 376								(RSRCL) = 377							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

ICB2 (ICB2F)

INCREMENT BYTE BY 2 (UPDATE CONDITION CODE FLAGS)



OPCODE: 112000-112377 (112400-112777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)+2
DESCRIPTION: The sum of the 8-bit contents of RB plus 2 is placed in RA. the contents of RB are unchanged. all status bit flags, except C4, are updated. ICB2F causes the N, Z, and C condition code flags to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
(ICB2F ONLY) Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

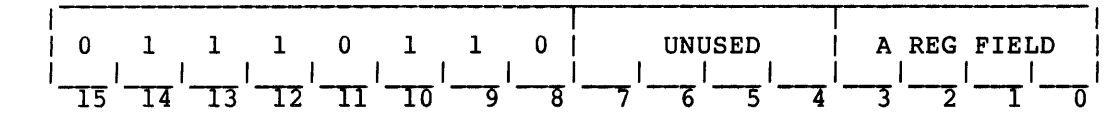
ASSEMBLY MNEMONIC: ICB2 RDSTL,RSRCL ;ADD 2 TO RDSTL, SUM IN RSRCL

ASSEMBLED OCTAL: 112544

BEFORE		AFTER	
(RDSTL) = 002		(RDSTL) = 002	
(RSRCL) = 376		(RSRCL) = 001	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0	

CIB

CONDITIONALLY INCREMENT BYTE



OPCODE: 073000-073377
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)+1, IF C8 = 1
DESCRIPTION: The 8-bit contents of RA are incremented by 1 if the C8 status bit flag was set = 1 by a previous operation. All status bit flags, except C4, are updated. The B register field is unused.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if the ZB flag was set prior to the operation and byte result was = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES:
N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

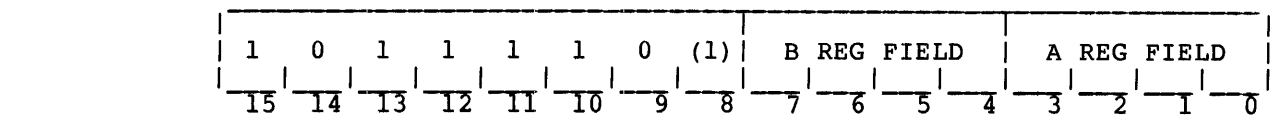
ASSEMBLY MNEMONIC: CIB RDSTL ;IF C8 = 1,INCR RDSTL BY 1

ASSEMBLED OCTAL: 073006

BEFORE								AFTER							
(RDSTL) = 002								(RDSTL) = 003							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

DB1 (DB1F)

DECREMENT BYTE BY 1 (UPDATE CONDITION CODE FLAGS)



OPCODE: 136000-136377 (136400-136777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)-1
DESCRIPTION: The 8-bit contents of RB are decremented by 1. The contents of RB are unchanged. All status bit flags, except C4, are updated. DB1F causes the condition code flags, except V, to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (DB1F ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if a borrow occurs; otherwise cleared

EXAMPLE:

ASSEMBLY MNEMONIC: DB1 RDSTL,RSRCL ;DECR RDSTL BY 1, PLACE IN RSRCL

ASSEMBLED OCTAL: 136044

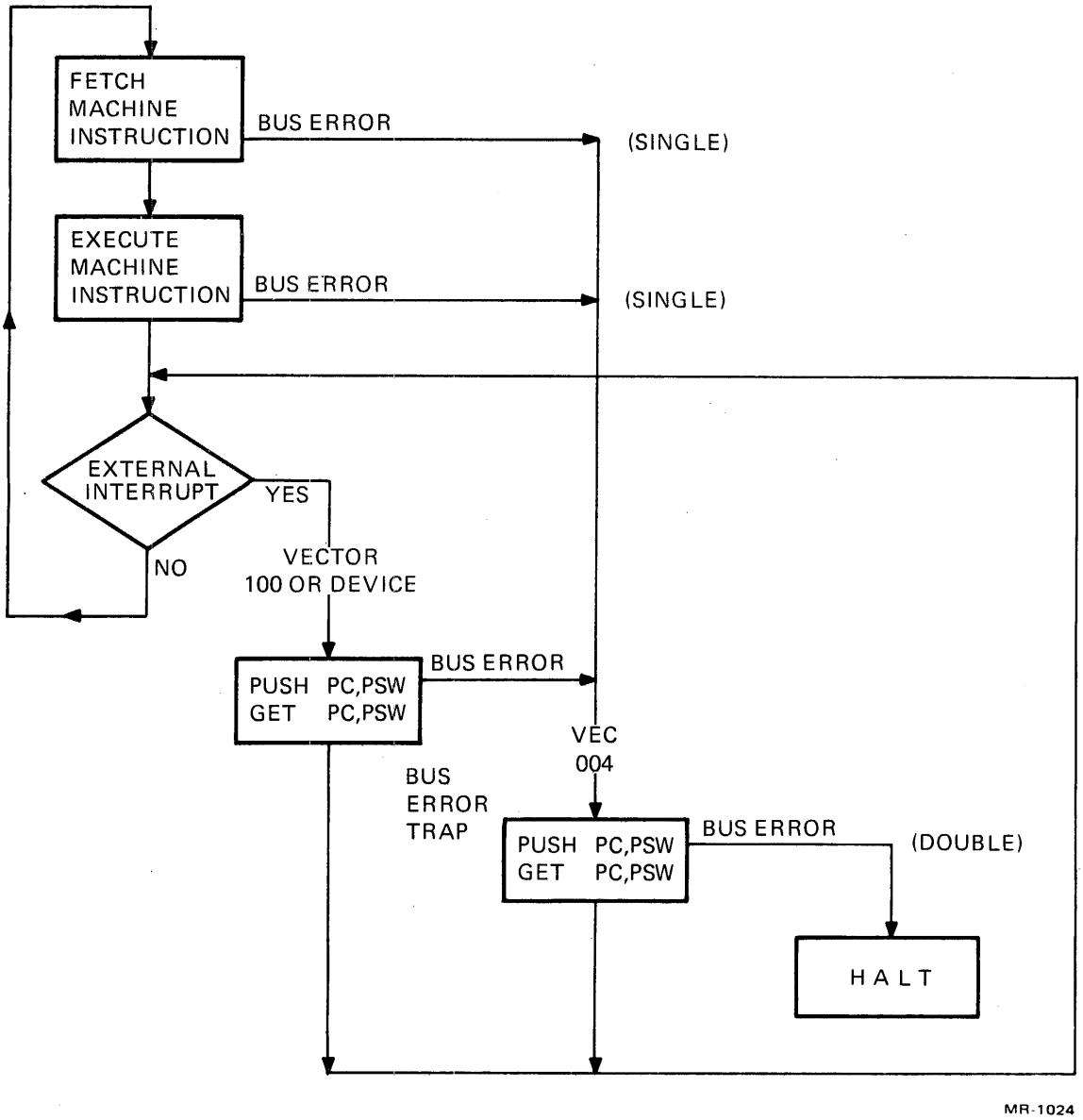
BEFORE	AFTER
(RDSTL) = 002	(RDSTL) = 002
(RSRCL) = 100	(RSRCL) = 001

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

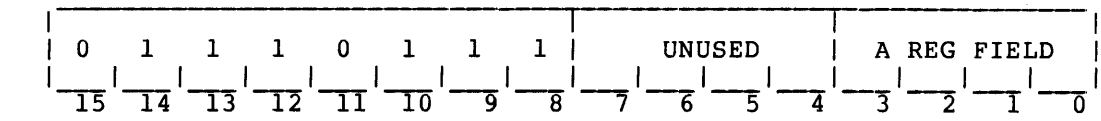
External Interrupt and Bus Error Trap

Figure 3-25



CDB

CONDITIONALLY DECREMENT BYTE BY 1



OPCODE: 073400-073777
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)-1, IF C8 = 0
DESCRIPTION: The 8-bit contents of RA are decremented by 1 if the C8 status bit flag was set = 0 by a previous operation. All status bit flags, except C4, are updated. The B register field is unused.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if the ZB flag was set prior to the operation and byte result was = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 0; cleared otherwise

CONDITION CODES:
N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: CDB RDSTL ;IF C8 = 1,DECR RDSTL BY 1

ASSEMBLED OCTAL: 073406

BEFORE

(RDSTL) = 002

NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0

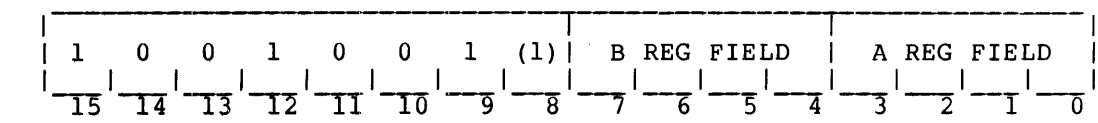
AFTER

(RDSTL) = 001

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

ICW1 (ICW1F)

INCREMENT WORD BY 1 (UPDATE CONDITION CODE FLAGS)



OPCODE: 111000-111377 (111400-111777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+1
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB plus 1 is placed in RA+1:RA when B is even. When B is odd, the sum of the contents of RB with its high-order bit extended through the 8 high-order bits and 1 is placed in RA+1:RA. The contents of RB+1:RB are unchanged. All status bit flags, except C4, are updated. ICW1F causes the condition code flags, except V, to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (ICB1F ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: ICW1F RDST,RSRC ;ADD 1 TO RDST, SUM IN RSRC

ASSEMBLED OCTAL: 111544

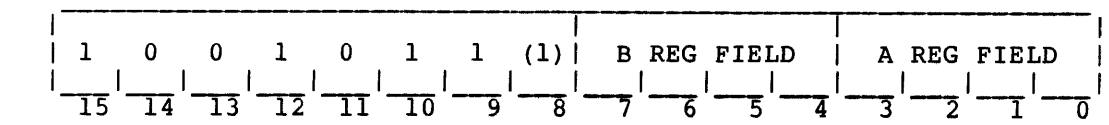
BEFORE	AFTER
(RDST) = 177776	(RDST) = 177776
(RSRC) = 000000	(RSRCL) = 177777

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	1	0	0	0

ICW2 (ICW2F)

INCREMENT WORD BY 2 (UPDATE CONDITION CODE FLAGS)



OPCODE: 113000-113377 (113400-113777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+2
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB plus 2 is placed in RA+1:RA when B is even. When B is odd, the sum of the contents of RB with its high-order bit extended through 8 high-order bits and 2 is placed in RA+1:RA. The contents of RB+1:RB are unchanged. All status bit flags, except C4, are updated. ICW2F causes the condition code flags, except V, to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (ICW2F ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: ICW2F RDST,RSRC ;ADD 2 TO RDST, SUM IN RSRC

ASSEMBLED OCTAL: 113544

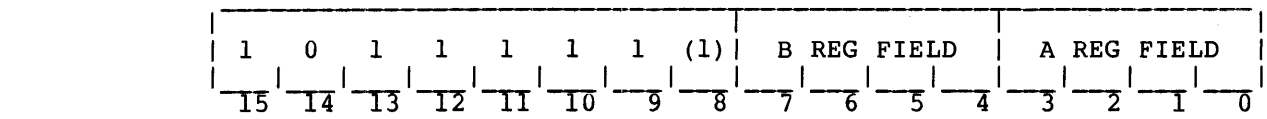
BEFORE	AFTER
(RDST) = 177776	(RDST) = 177776
(RSRC) = 000000	(RSRCL) = 000001

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

DWl (DWlF)

DECREMENT WORD BY 1 (UPDATE CONDITION CODE FLAGS)



OPCODE: 137000-137377 (137400-137777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)-1
DESCRIPTION: The 16-bit contents of RB+1:RB are decremented by 1 and placed in RA+1:RA. The contents of RB+1:RB are unchanged when B is even. When B is odd, the contents of RB with its high-order bit extended through 8 high-order bits minus 1 is placed in RA+1:RA. All status bit flags, except C4, are updated. DWlF causes the condition code flags, except V, to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (DWlF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if a borrow occurs; otherwise cleared

EXAMPLE:

ASSEMBLY MNEMONIC: DWlF RDST,RSRC ;DECR RDST BY 1, PLACE IN RSRC

ASSEMBLED OCTAL: 137544

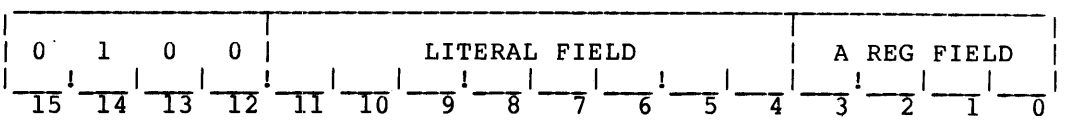
BEFORE		AFTER	
(RDST) = 000001		(RDST) = 000001	
(RSRC) = 000000		(RSRCL) = 000000	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 1 0 0 0 1 0 0	

4.3.1.3 Logical Microinstructions - The Logical Microinstructions perform variations of the basic unary and binary operations: And, And Complement, Or, Exclusive Or, Ones Complement, and Twos Complement.

The microinstruction descriptions are as follows:

NL

AND LITERAL



OPCODE: 04XXXX
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)"AND"(LITERAL FIELD)
DESCRIPTION: The 8-bit contents of the literal field, MI<11:4>, are ANDed with the 8-bit contents of RA and the result is loaded into RA. The content of the literal field remains unchanged. The NB and ZB status bit flags are updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

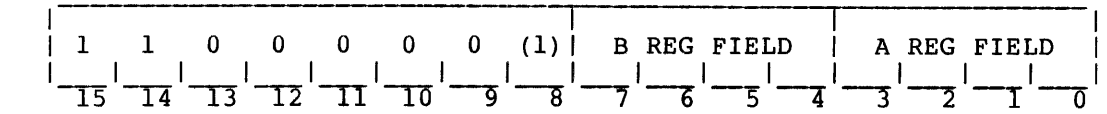
ASSEMBLY MNEMONIC: NL 200,RBAL ;200 "AND" RBAL, INTO RBAL

ASSEMBLED OCTAL: 044002

BEFORE								AFTER							
(RBAL) = 377								(RBAL) = 200							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

NB (NBF)

AND BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 140000-140377 (140400-140777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)"AND"(RA)
DESCRIPTION: The 8-bit contents of RB are ANDed with the 8-bit contents of RA and placed in RA. The content of RB is unchanged. The status bit flags are updated. NBF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (NBF ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

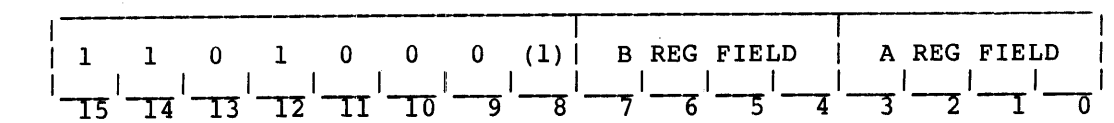
ASSEMBLY MNEMONIC: NBF RDSTL,RSRCL ;RDSTL"AND"RSRCL, RESULT IN RSRCL

ASSEMBLED OCTAL: 140544

BEFORE		AFTER	
(RDSTL) = 201		(RDSTL) = 201	
(RSRCL) = 176		(RSRCL) = 000	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 1 0 0 0 1 0 0	

NCB (NCBF)

AND COMPLEMENT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 150000-150377 (150400-150777)
MICROCYCLES: 1
OPERATION: (RA) <-- $\overline{(RB)}$"AND"(RA)
DESCRIPTION: The complement of the 8-bit contents of RB are ANDed with the 8-bit contents of RA and placed in RA. The content of RB is unchanged. The status bit flags are updated. NCBF causes the condition code flags to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (NCBF ONLY) N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

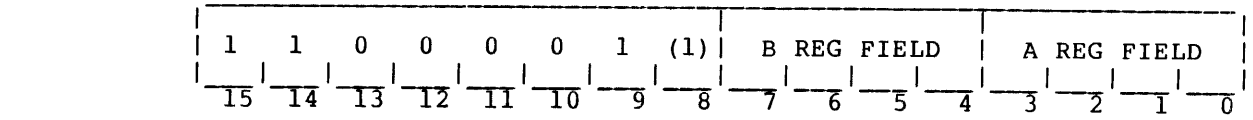
ASSEMBLY MNEMONIC: NCBF RDSTL,RSRCL ;$\overline{RDSTL}$"AND"RSRCL, RESULT IN RSRCL

ASSEMBLED OCTAL: 150544

BEFORE		AFTER	
(RDSTL) = 201		(RDSTL) = 201	
(RSRCL) = 176		(RSRCL) = 176	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0	

NW (NWF)

AND WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 141000-141377 (141400-141777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)"AND"(RA+1:RA)
DESCRIPTION: The 16-bit contents of RB+1:RB are ANDed with the 16-bit contents of RA+1:RA and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. NWF causes the condition code flags to be updated.

STATUS BITS:
NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (NWF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

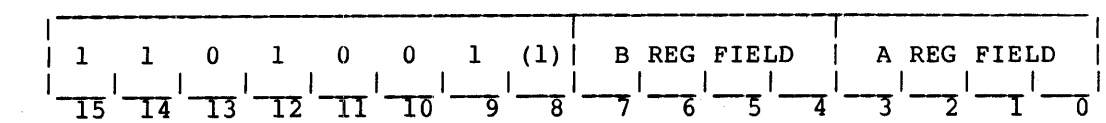
ASSEMBLY MNEMONIC: NW RDST,RSRC ;RDST"AND"RSRC, RESULT IN RSRC

ASSEMBLED OCTAL: 141144

BEFORE		AFTER	
(RDST) = 000000		(RDST) = 000000	
(RSRC) = 177777		(RSRC) = 000000	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 1 0 0 0 0 0 0	

NCW (NCWF)

AND COMPLEMENT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 151000-151377 (151400-151777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- $\overline{(RB+1:RB)}$"AND"(RA+1:RA)
DESCRIPTION: The 16-bit contents of RB+1:RB are complemented and ANDed with the 16-bit contents of RA+1:RA and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. NCWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(NCWF ONLY) Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

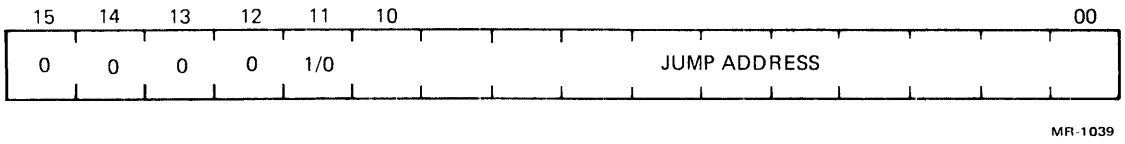
ASSEMBLY MNEMONIC: NCW RDST,RSRC ;$\overline{RDST}$"AND"RSRC, RESULT IN RSRC

ASSEMBLED OCTAL: 151144

BEFORE								AFTER							
(RDST) = 000000								(RDST) = 000000							
(RSRC) = 177777								(RSRC) = 177777							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0

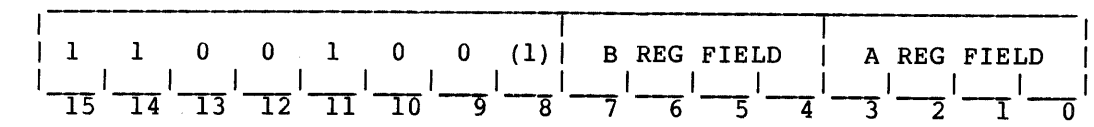
Unconditional Jump Microinstruction Format

Figure 4-2



ORB (ORBF)

OR BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 144000-144377 (144400-144777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)"OR"(RA)
DESCRIPTION: The 8-bit contents of RB are ORed with the 8-bit contents of RA and placed in RA. The content of RB is unchanged. The status bit flags are updated. ORBF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (ORBF ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

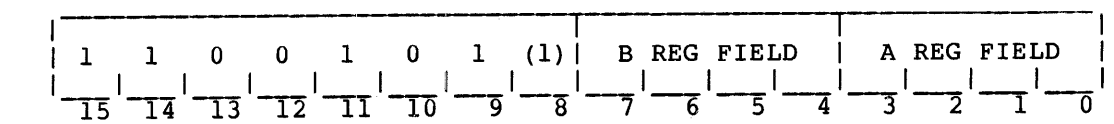
ASSEMBLY MNEMONIC: ORBF RDSTL,RSRCL ;RDSTL"OR"RSRCL, RESULT IN RSRCL

ASSEMBLED OCTAL: 144544

BEFORE		AFTER	
(RDSTL) = 201		(RDSTL) = 201	
(RSRCL) = 176		(RSRCL) = 377	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		1 0 0 0 1 0 0 0	

ORW (ORWF)

OR WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 145000-145377 (145400-145777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)"OR"(RA+1:RA)
DESCRIPTION: The 16-bit contents of RB+1:RB are ORed with the 16-bit contents of RA+1:RA and placed in RA+1:RA. The contents of RB+1:RB are unchanged. The status bit flags are updated. ORWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(ORWF ONLY) Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

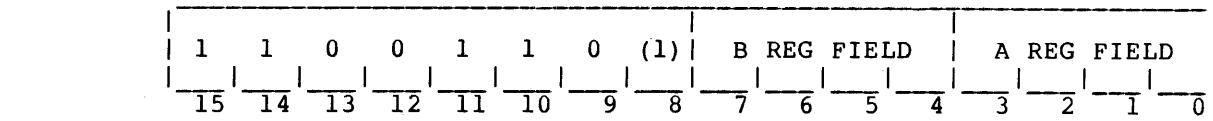
ASSEMBLY MNEMONIC: ORW RDST,RSRC ;RDST"OR"RSRC, RESULT IN RSRC

ASSEMBLED OCTAL: 145144

BEFORE		AFTER	
(RDST) = 125252		(RDST) = 125252	
(RSRC) = 052525		(RSRC) = 177777	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		1 0 0 0 0 0 0 0	

XB (XBF)

EXCLUSIVE-OR BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 146000-146377 (146400-146777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)"X-OR"(RA)
DESCRIPTION: The 8-bit contents of RB are EXCLUSIVE-ORed with the 8-bit contents of RA and placed in RA. The content of RB is unchanged. The status bit flags are updated. XBF causes the condition code flags, except C, to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
(XBF ONLY) Z: set if the byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

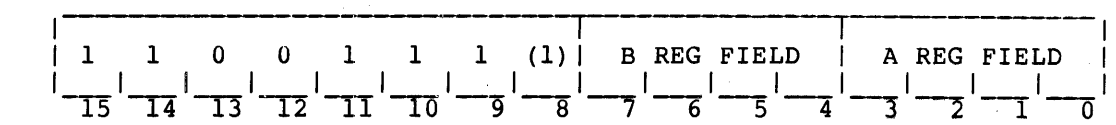
ASSEMBLY MNEMONIC: XBF RDSTL,RSRCL ;RDSTL"X-OR"RSRCL, RES IN RSRCL

ASSEMBLED OCTAL: 146544

BEFORE								AFTER							
(RDSTL) = 201								(RDSTL) = 201							
(RSRCL) = 201								(RSRCL) = 000							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

XW (XWF)

EXCLUSIVE-OR WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 147000-147377 (147400-147777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)"X-OR"(RA+1:RA)
DESCRIPTION: The 16-bit contents of RB+1:RB are EXCLUSIVE ORed with the 16-bit contents of RA+1:RA and placed in RA+1:RA. The contents of RB+1:RB are unchanged. The status bit flags are updated. XWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(XWF ONLY) Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: XRW RDST,RSRC ;RDST"X-OR"RSRC, RESULT IN RSRC

ASSEMBLED OCTAL: 147144

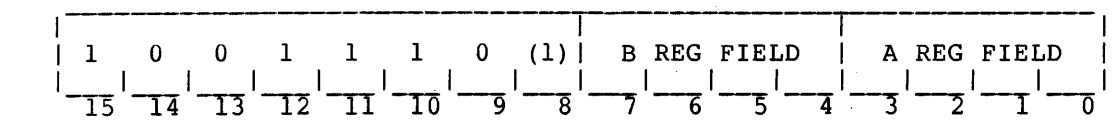
BEFORE	AFTER
(RDST) = 125252	(RDST) = 125252
(RSRC) = 052525	(RSRC) = 000000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	0	0	0

OCB (OCBF)

ONES COMPLEMENT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 116000-116377 (116400-116777)
MICROCYCLES: 1
OPERATION: (RA) <-- $\overline{(RB)}$
DESCRIPTION: The ones complement of the 8-bit content of RB is placed in RA. The content of RB is unchanged. The status bit flags are updated. OCBF causes the condition code flags to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
(OCBF ONLY) Z: set if the byte result = 0; cleared otherwise
V: not affected
C: cleared

EXAMPLE:

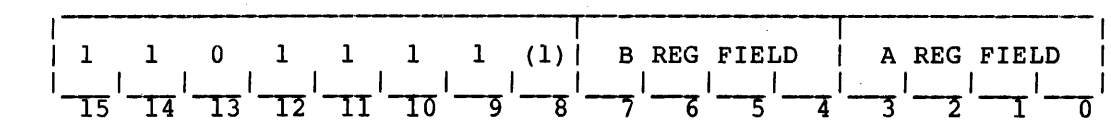
ASSEMBLY MNEMONIC: OCB RDSTL,RSRCL ;ONES COMP OF RDSTL INTO RSRCL

ASSEMBLED OCTAL: 154144

BEFORE								AFTER							
(RDSTL) = 001								(RDSTL) = 001							
(RSRCL) = 000								(RSRCL) = 376							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0

OCW (OCWF)

ONES COMPLEMENT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 117000-117377 (117400-117777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- $\overline{(RB+1:RB)}$
DESCRIPTION: The ones complement of the 16-bit contents of RB+1:RB are placed in RA+1:RA when B is even. When B is odd, the ones complement of the contents of RB with its high-order bit extended through 8 high-order bits is placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. OCWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(OCWF ONLY) Z: set if the word result = 0; cleared otherwise
V: not affected
C: cleared

EXAMPLE:

ASSEMBLY MNEMONIC: OCWF RDST,RSRC ;ONES COMP OF RDST INTO RSRC

ASSEMBLED OCTAL: 157544

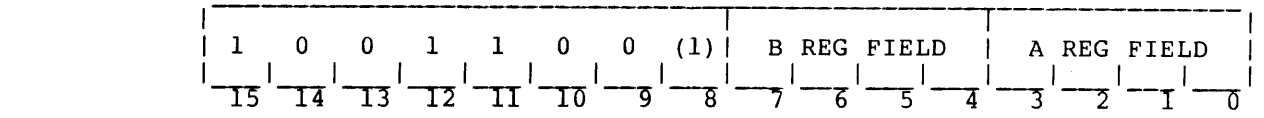
BEFORE	AFTER
(RDST) = 000001	(RDST) = 000001
(RSRC) = 000000	(RSRC) = 177776

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	1	0	0	0

TCB (TCBF)

TWOS COMPLEMENT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 114000-114377 (114400-114777)
MICROCYCLES: 1
OPERATION: (RA) <-- $\overline{(RB)}$+1
DESCRIPTION: The twos complement of the 8-bit content of RB is placed in RA. The content of RB is unchanged. The status bit flags are updated. TCBF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (TCBF ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: TCB RDSTL,RSRCL ;TWO'S COMP OF RDSTL INTO RSRCL

ASSEMBLED OCTAL: 134144

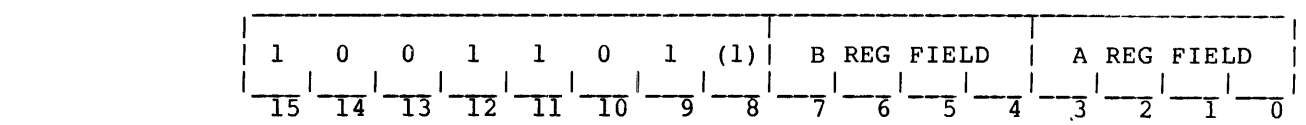
BEFORE	AFTER
(RDSTL) = 001	(RDSTL) = 001
(RSRCL) = 000	(RSRCL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	1	0	0	0

TCW (TCWF)

TWO'S COMPLEMENT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 115000-115377 (115400-115777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- $\overline{(RB+1:RB)}$+1
DESCRIPTION: The two's complement of the 16-bit contents of RB+1:RB is placed in RA+1:RA when B is even. When B is odd, the two's complement of the contents of RB with its high-order bit extended through 8 high-order bits is placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. TCWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(TCWF ONLY) Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: TCW RDST,RSRC ;TWO'S COMP OF RDST INTO RSRC

ASSEMBLED OCTAL: 154544

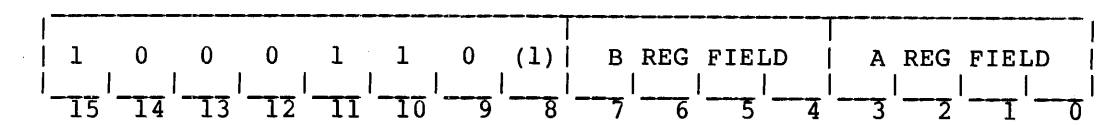
BEFORE		AFTER	
(RDST) = 000001		(RDST) = 000001	
(RSRC) = 000000		(RSRC) = 177777	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		1 0 0 0 1 0 0 0	

4.3.1.4 Shift Microinstructions - The Shift Microinstructions are divided into 8 right shift and 8 left shift operations. Those shift operations which execute "With Carry" implement 8 or 16 bit shift registers using the C condition code flag as the bit that is shifted in.

The microinstruction descriptions are as follows:

SLB (SLBF)

SHIFT LEFT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 106000-106377 (106400-106777)
MICROCYCLES: 1
OPERATION: (RA) <-- 2(RB)
DESCRIPTION: The 8-bit contents of RB are shifted left one bit position and placed in RA. The content of RB is unchanged. A zero is inserted into the vacated low-order bit postion. The status bit flags are updated. SLBF causes the condition code flags to be updated, with the C condition code flag receiving the high-order shifted-off bit.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
(SLBF ONLY) Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: SLB RDSTL,RSRCL ;SHIFT RDSTL LEFT, INTO RSRCL
;IGNORE CARRY

ASSEMBLED OCTAL: 106144

BEFORE

(RDSTL) = 010

(RSRCL) = 000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1

AFTER

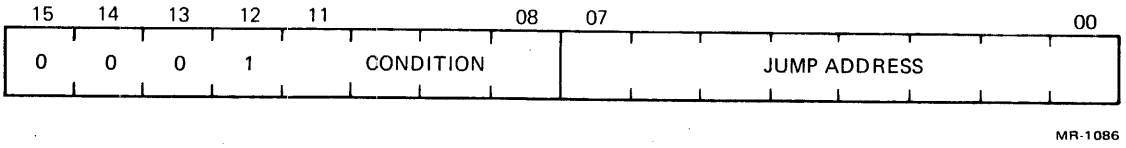
(RDSTL) = 010

(RSRCL) = 020

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

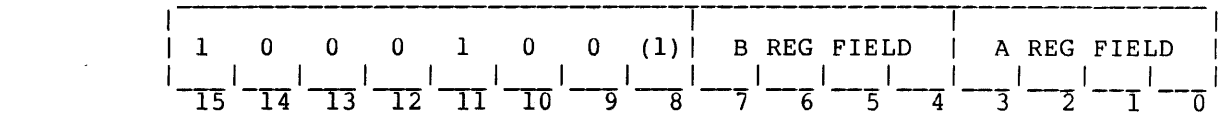
Conditional Jump Microinstruction Format

Figure 4-3



SLBC (SLBCF)

SHIFT LEFT BYTE WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 104000-104377 (104400-104777)
MICROCYCLES: 1
OPERATION: (RA) <-- 2(RB)+C
DESCRIPTION: The 8-bit contents of RB are shifted left one bit position and placed in RA. The content of the C condition code flag is placed in the vacated low-order bit postion. The content of RB is unchanged. The status bit flags are updated. SLBCF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: (SLBCF ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

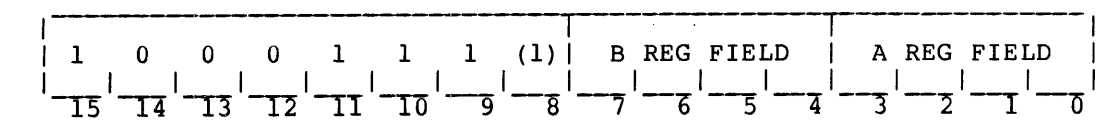
ASSEMBLY MNEMONIC: SLBC RDSTL,RSRCL ;SHIFT RDSTL LEFT, INTO RSRCL

ASSEMBLED OCTAL: 104144

BEFORE								AFTER							
(RDSTL) = 010								(RDSTL) = 010							
(RSRCL) = 000								(RSRCL) = 021							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SLW (SLWF)

SHIFT LEFT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 107000-107377 (107400-107777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- 2(RB+1:RB)
DESCRIPTION: The 16-bit contents of RB+1:RB are shifted left one bit position and placed in RA+1:RA when B is even. When B is odd, the contents of RB with its high-order bit extended through 8 high-order bits is shifted left one bit position and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. SLBF causes the condition code flags to be updated, with the C flag receiving the high-order shifted-off bit.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: (SLWF ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the WORD result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

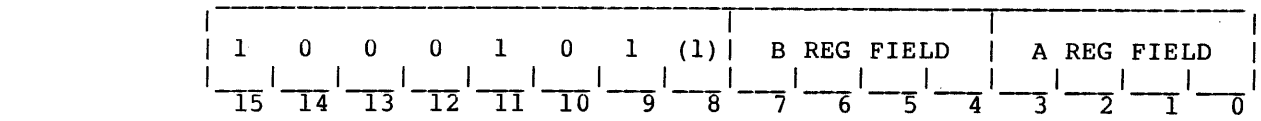
ASSEMBLY MNEMONIC: SLW RDST,RSRC ;SHIFT RDST LEFT, INTO RSRC
;IGNORE CARRY

ASSEMBLED OCTAL: 107044

BEFORE								AFTER							
(RDST) = 010000								(RDST) = 010000							
(RSRC) = 010000								(RSRC) = 020000							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SLWC (SLWCF)

SHIFT LEFT WORD WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 105000-105377 (105400-105777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- 2(RB+1:RB)+C
DESCRIPTION: The 16-bit contents of RB+1:RB are shifted left one bit position and placed in RA+1:RA when B is even. When B is odd, the contents of RB with its high-order bit extended through 8 high-order bit positions is shifted left one bit position and placed in RA+1:RA. The content of the C condition code flag is placed in the vacated low-order bit postion. The content of RB+1:RB is unchanged. The status bit flags are updated. SLWCF causes the condition code flags to be updated, with the C flag receiving the high-order shifted-off bit.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(SLWCF ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

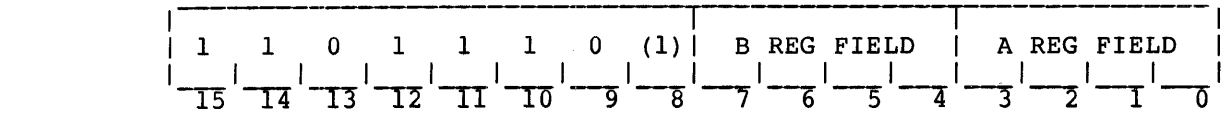
ASSEMBLY MNEMONIC: SLWC RDST,RSRC ;SHIFT RDST LEFT, INTO RSRC

ASSEMBLED OCTAL: 105144

BEFORE		AFTER	
(RDST) = 010000		(RDST) = 010000	
(RSRC) = 010000		(RSRC) = 020001	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		0 0 0 0 0 0 0 0	

SRB (SRBF)

SHIFT RIGHT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 156000-156377 (156400-156777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)/2
DESCRIPTION: The 8-bit contents of RB are shifted right one bit position and placed in RA. The content of RB is unchanged. The status bit flags are updated. SRBF causes the condition code flags to be updated, with the C flag receiving the low-order shifted-off bit.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: (SRBF ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: not affected
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

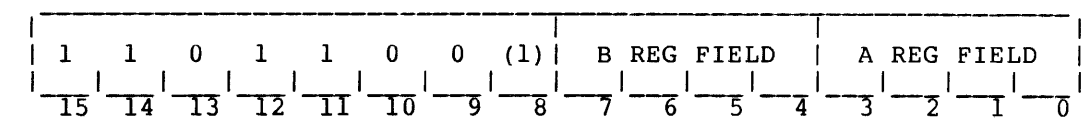
```
ASSEMBLY MNEMONIC:  SRB RDSTH,RSRCH  ;SHIFT RDSTH RIGHT, INTO RSRCH
                                      ;IGNORE CARRY

ASSEMBLED OCTAL:    156165
```

BEFORE		AFTER	
(RDSTH) = 010		(RDSTH) = 010	
(RSRCH) = 000		(RSRCH) = 004	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		0 0 0 0 0 0 0 1	

SRBC (SRBCF)

SHIFT RIGHT BYTE WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 154000-154377 (154400-154777)
MICROCYCLES: 1
OPERATION: (RA) <-- (C:(RB))/2
DESCRIPTION: The 8-bit contents of RB are shifted right one bit position and placed in RA. The content of the C condition code flag is placed in the vacated high-order bit postion. The content of RB is unchanged. The status bit flags are updated. SRBCF causes the condition code flags to be updated, with the C flag receiving the low-order shifted-off bit.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
(SRBCF ONLY) Z: set if the byte result = 0; cleared otherwise
V: not affected
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

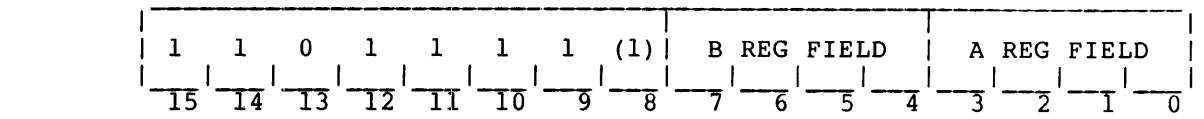
ASSEMBLY MNEMONIC: SRBC RDSTL,RSRCL ;SHIFT RDSTL RIGHT, INTO RSRCL

ASSEMBLED OCTAL: 154144

BEFORE								AFTER							
(RDSTL) = 010								(RDSTL) = 010							
(RSRCL) = 000								(RSRCL) = 204							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SRW (SRWF)

SHIFT RIGHT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 157000-157377 (157400-157777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)/2
DESCRIPTION: The 16-bit contents of RB+1:RB are shifted right one bit position and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. SRWF causes the condition code flags to be updated, with the C flag receiving the low-order shifted-off bit. A and B must both be odd since shifting starts with the left byte.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: (SRWF ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: not affected
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC:
```
SRWF RDSTH,RSRCH  ;SHIFT RDST RIGHT, INTO RSRC
                  ;IGNORE CARRY
```

ASSEMBLED OCTAL: 157565

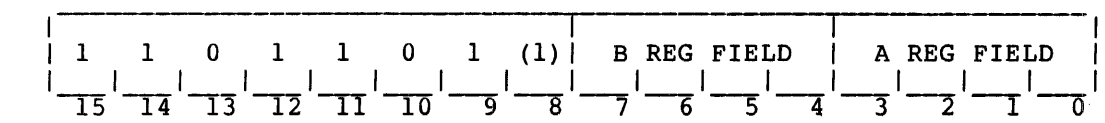
BEFORE	AFTER
(RDST) = 000001	(RDST) = 000001
(RSRC) = 000000	(RSRC) = 000000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	1	0	1

SRWC (SRWCF)

SHIFT RIGHT WORD WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 155000-155377 (155400-155777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (C:(RB+1:RB))/2
DESCRIPTION: The 16-bit contents of RB+1:RB are shifted right one bit position and placed in RA+1:RA. The content of the C condition code flag is placed in the vacated high-order bit postion. The content of RB+1:RB is unchanged. The status bit flags are updated. SRWCF causes the condition code flags to be updated, with the C flag receiving the low-order shifted-off bit. A and B must both be odd since shifting starts with the left byte.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: set if shifted-off bit = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(SRWCF ONLY) Z: set if the word result = 0; cleared otherwise
V: not affected
C: set if shifted-off bit = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: SRWC RDSTH,RSRCH ;SHIFT RDST RIGHT, INTO RSRC

ASSEMBLED OCTAL: 155144

BEFORE	AFTER
(RDST) = 000001	(RDST) = 000001
(RSRC) = 000000	(RSRC) = 100000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	1

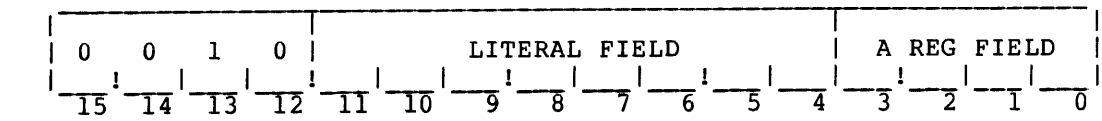
NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	1	0	0	1

4.3.1.5 Arithmetic Microinstructions - The Arithmetic Microinstructions consist mainly of operations which either add or subtract values expressed in twos complement representation. One microinstruction facilitates decimal arithmetic (CAD).

The microinstruction descriptions are as follows:

AL

ADD LITERAL



OPCODE: 02XXXX
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)+(LITERAL FIELD)
DESCRIPTION: The 8-bit contents of the literal field, MI<11:4>, are added to the 8-bit contents of RA and loaded into register RA. The NB, ZB, C4, and C8 status bit flags are updated. The contents of the literal field are not changed.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES:
N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

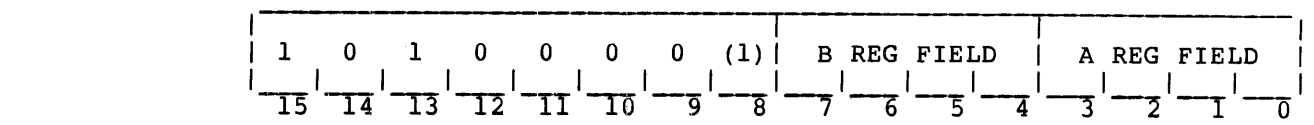
ASSEMBLY MNEMONIC: AL 200,RBAL ;ADD 200 TO RBAL

ASSEMBLED OCTAL: 024202

BEFORE								AFTER							
(RBAL) = 210								(RBAL) = 020							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

AB (ABF)

ADD BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 120000-120377 (120400-120777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)+(RA)
DESCRIPTION: The sum of the 8-bit contents of RB and RA are placed in RA. The content of RB is unchanged. The status bit flags are updated. ABF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (ABF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: ABF RDSTL,RSRCL ;ADD RDSTL TO RSRCL, RES IN RSRCL

ASSEMBLED OCTAL: 160544

BEFORE		AFTER	
(RDSTL) = 001		(RDSTL) = 001	
(RSRCL) = 001		(RSRCL) = 002	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0	

4.2.3 Literal Microinstruction Format

The Literal Microinstruction Format is shown in Figure 4.4. These microinstructions provide 8-bit constants in the microprogram. There are 5 microinsructions in this group and the octal opcode values (MI<15:12>) in the range 02 to 06. An 8-bit literal field, (MI<11:3>), is one operand. The other operand is specified by the A Register Field (MI<3:0>) which supports direct or indirect addressing of any 8-bit byte register through the A Read/Write port. All literal format microinstructions require 1 microcycle for execution.

4.2.4 Register Microinstruction Format

Most of the LSI-11 microinstructions belong to the Register Microinstruction Format which is illustrated in Figure 4-5. The opcode field occupies 8 bits, MI<15:8>, and opcode values range from 07 to 37 (octal). Most register format microinstructions unconditionally update the status bit flags relevant to the particular operation. The condition code flags are selectively updated by opcode choice, the odd opcode value (in general) affecting the condition code flags in addition to the status bit flags.

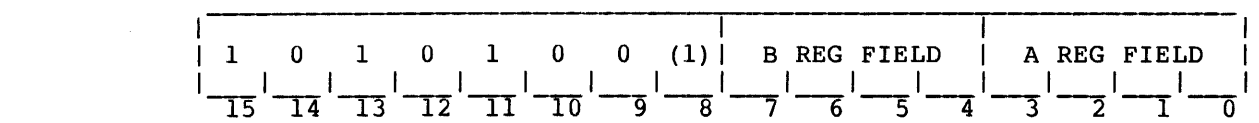
4.2.4.1 Byte and Word Microinstructions - All register format microinstructions belonging to the data manipulation and the data access classifications are either byte or word microinstructions. The byte/word distinction is a direct consequence of the fact that the register file access ports are 8-bits wide. This 8-bit width is also shared by the ALU inputs and output. A data manipulation microinstruction which produces a single 8-bit byte as its result can be completed in one microcycle. An example is the Add Byte microinstruction which forms the binary addition of the bytes addressed by the B and A register fields and places the result in the register designated by the A field. The microprocessor Data chip simultaneously presents the two 8-bit operands to the ALU and restructures the A register port path to load the ALU output when the result is formed.

In the case of a word microinstruction, the 8-bit register port and ALU width requires a 2-microcycle sequence to complete processing of 16-bit word operands. Carry or borrow information which needs to be transmitted between byte operations is stored in the ALU status bit flags. The ALU status bit flags are unconditionally updated by each byte operation. The Add Word microinstruction is the 16-bit, 2-microcycle counterpart of the Add Byte microinstruction. Since the B and A register fields of the microinstructions each point to a single 8-bit byte within the register file, special techniques are used to form the 16-bit operands for word microinstructions. These techniques are explained in the next section.

There are several microinstructions which produce a word operand but still fall into the byte classification. This is because the Data chip organization allows the required 16-bit path to be simultaneously formed from two 8-bit paths. An example is the Write microinstruction, which is able to simultaneously present the contents of two independently specified registers to the data address buffers, and thus to the WDAL Data chip outputs.

ABC (ABCF)

ADD BYTE WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 124000-124377 (124400-124777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)+(RA)+C
DESCRIPTION: The sum of the 8-bit contents of RB and RA plus C are placed in RA. The content of RB is unchanged. The status bit flags are updated. ABCF causes the condition code flags to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
ABCF (ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

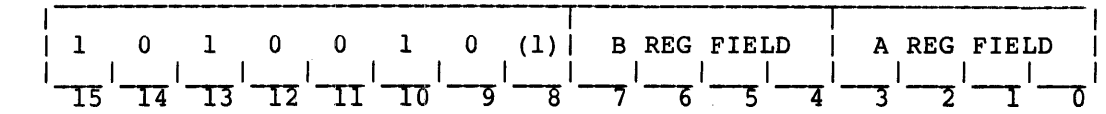
ASSEMBLY MNEMONIC: ABCF RDSTL,RSRCL ;ADD RDSTL TO RSRCL, RES IN RSRCL

ASSEMBLED OCTAL: 164544

BEFORE		AFTER	
(RDSTL) = 001		(RDSTL) = 001	
(RSRCL) = 001		(RSRCL) = 003	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		0 0 0 0 0 0 0 0	

CAB (CABF)

CONDITIONALLY ADD BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 122000-122377 (122400-122777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RB)+(RA), IF C = 1
DESCRIPTION: The sum of the 8-bit contents of RB and RA are placed in RA, if the C flag is a 1. If the C flag is not a 1, the add operation will not take place and RA will remain unchanged. The content of RB is unchanged. The status bit flags are updated. CABF causes the condition code flags to be updated.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(CABF ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

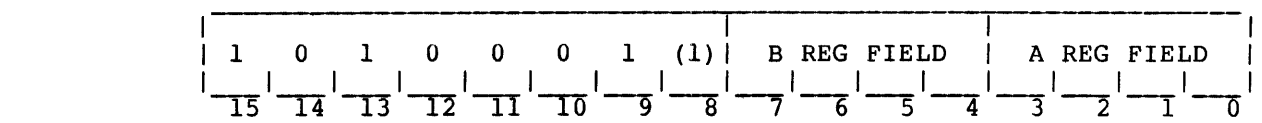
EXAMPLE:

```
ASSEMBLY MNEMONIC:  CAB RDSTL,RSRCL ;ADD RDSTL TO RSRCL, RES IN RSRCL
                                     ;(IF C PREVIOUSLY SET)
ASSEMBLED OCTAL:    162144
```

BEFORE								AFTER							
(RDSTL) = 001								(RDSTL) = 001							
(RSRCL) = 001								(RSRCL) = 001							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AW (AWF)

ADD WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 121000-121377 (121400-121777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+(RA+1:RA)
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB and RA+1:RA is placed in RA+1:RA when B is even. When B is odd, the sum of the contents of RB with its high order bit extended through 8 high order bits and the contents of RA+1:RA is placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. AWF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (AWF ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: AW RDST,RSRC ;ADD RDST TO RSRC, SUM IN RSRC

ASSEMBLED OCTAL: 161144

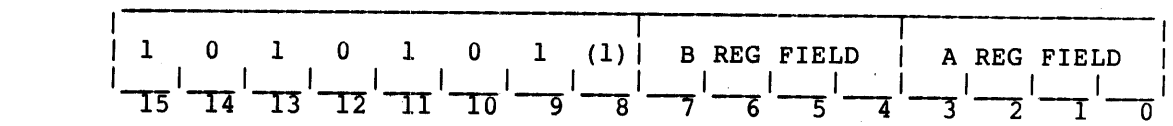
BEFORE	AFTER
(RDST) = 000377	(RDST) = 000377
(RSRC) = 000001	(RSRC) = 000400

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	0	1	1	0	0	0	0

AWC (AWCF)

ADD WORD WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 125000-125377 (125400-125777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+(RA+1:RA)+C
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB and RA+1:RA plus C is placed in RA+1:RA when B is even. When B is odd, the sum of the contents of RB with its high order bit extended through 8 high order bits and RA+1:RA plus C is placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. AWCF causes the condition code flags to be updated.

STATUS BITS:
NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (AWCF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

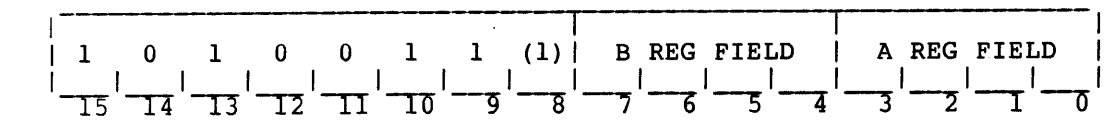
ASSEMBLY MNEMONIC: AWCF RDST,RSRC ;ADD RDST TO RSRC, SUM IN RSRC

ASSEMBLED OCTAL: 165544

BEFORE		AFTER	
(RDST) = 000377		(RDST) = 000377	
(RSRC) = 000001		(RSRC) = 000401	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		0 0 1 1 0 0 0 0	

CAW (CAWF)

CONDITIONALLY ADD WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 123000-123377 (123400-123777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+(RA+1:RA), IF C=1
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB and RA+1:RA is placed in RA+1:RA when B is even. When B is odd, the sum of the contents of RB with its high order bit extended through 8 high order bits and RA+1:RA is placed in RA+1:RA if the value of the C flag is 1. The contents of RB+1:RB remain unchanged. The status bit flags are updated. CAWF causes the condition code flags to be updated. If the C flag is not a one, the add operation will not take place and no flag or register contents will be changed.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(CAWF ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

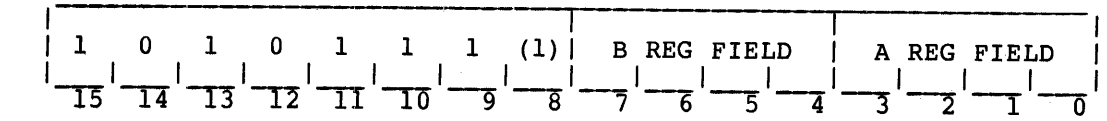
ASSEMBLY MNEMONIC: CAW RDST,RSRC ;ADD RDST TO RSRC, SUM IN RSRC
;(IF C PREVIOUSLY SET)

ASSEMBLED OCTAL: 163144

BEFORE		AFTER	
(RDST) = 000377		(RDST) = 000377	
(RSRC) = 000001		(RSRC) = 000001	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 0		0 0 1 1 0 0 0 0	

CAWI (CAWIF)

CONDITIONALLY ADD WORD ON ICC (UPDATE CONDITION CODE FLAGS)



OPCODE: 127000-127377 (127400-127777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RB+1:RB)+(RA+1:RA), IF ICC = 1
DESCRIPTION: The sum of the 16-bit contents of RB+1:RB and RA+1:RA is placed in RA+1:RA if the ICS flag is a 1, when B is even. When B is odd,the sum of the contents of RB with its high order bit extended through 8 high order bits and RA+1:RA is placed in RA+1:RA if the ICS flag is a 1. The contents of RB+1:RB remain unchanged. The status bit flags are updated. CAWIF causes the condition code flags to be updated. If the C flag is not a one, the add operation will not take place and no flag or register contents will be changed.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (CAWIF ONLY) N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: CAWIF RDST,RSRC ;ADD RDST TO RSRC, SUM IN RSRC
;(IF ICC PREVIOUSLY SET)

ASSEMBLED OCTAL: 167544

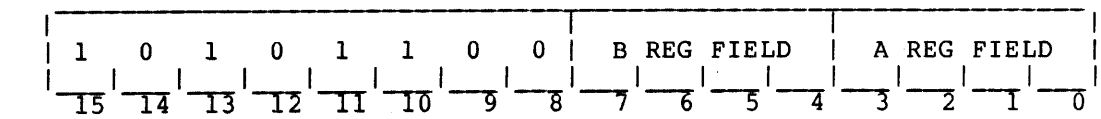
BEFORE : ICC = 1	AFTER
(RDST) = 177777	(RDST) = 177777
(RSRC) = 000001	(RSRC) = 000001

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
1	0	0	1	1	0	0	0

CAD

CONDINTIONALLY ADD DIGITS



OPCODE: 126000-126377
MICROCYCLES: 1
OPERATION: (RA<3:0>) <-- (RB<3:0>)+(RA<3:0>), IF C4 = 0,
(RA<7:4>) <-- (RB<7:4>)+(RA<7:4>), IF C8 = 0

DESCRIPTION: This microinstruction divides the designated register operands into two digits of 4 bits each, bits <7:4> and bits <3:0>. Status bit flag C4 must be 0 for the lower digit to add and C8 must be 0 for the higher digit to add. The contents of RB are unchanged. The status bit flags are updated in all cases. A carry out of bit position 3 will not effect the higher digit.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

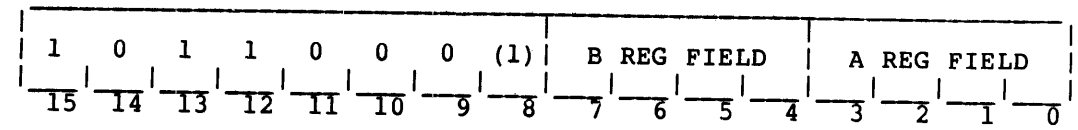
ASSEMBLY MNEMONIC: CAD RDSTL,RSRCL ;ADD RDSTL TO RSRCL,
;DIGIT SUM(S) IN RSRCL

ASSEMBLED OCTAL: 166144

BEFORE								AFTER							
(RDSTL) = 021								(RDSTL) = 021							
(RSRCL) = 021								(RSRC) = 041							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

SB (SBF)

SUBTRACT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 130000-130377 (130400-130777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)-(RB)
DESCRIPTION: The 8-bit contents of RB are subtracted from the 8-bit contents of RA and placed in RA. The contents of RB are unchanged. The status bit flags are updated. SBF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (SBF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C:set if there is a borrow; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC:
```
SB RDSTL,RSRCL  ;SUB RDSTL FROM RSRCL,
                ;DIF IN RSRCL
```

ASSEMBLED OCTAL: 170164

BEFORE

(RDSTL) = 377

(RSRCL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

AFTER

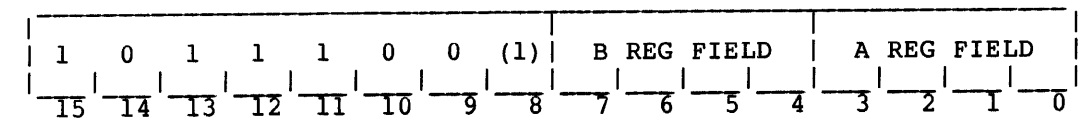
(RDSTL) = 377

(RSRCL) = 000

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	0	0	0

SBC (SBCF)

SUBTRACT BYTE WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 134000-134377 (134400-134777)
MICROCYCLES: 1
OPERATION: (RA) <-- (RA)-(RB)-C
DESCRIPTION: The 8-bit contents of RB minus the C flag are subtracted from the 8-bit contents of RA and placed in RA. The contents of RB are unchanged. The status bit flags are updated. SBCF causes the condition code flags to be updated.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (SBCF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if there is a borrow; cleared otherwise

EXAMPLE:

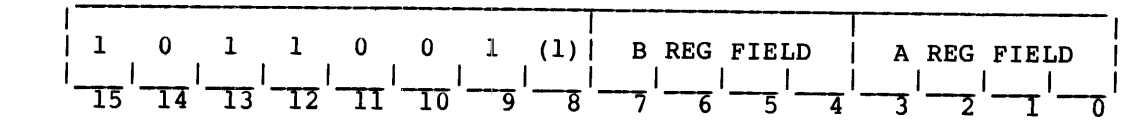
ASSEMBLY MNEMONIC:
```
SBCF RDSTL,RSRCL  ;SUB RDSTL, CARRY FROM
                  ;RSRCL, DIF IN RSRCL
```

ASSEMBLED OCTAL: 134544

BEFORE		AFTER	
(RDSTL) = 377		(RDSTL) = 377	
(RSRCL) = 002		(RSRCL) = 374	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		0 0 0 0 0 0 0 0	

SW (SWF)

SUBTRACT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 131000-131377 (131400-131777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RA+1:RA)-(RB+1:RB)
DESCRIPTION: The 16-bit contents of RB+1:RB are subtracted from the 16-bit contents of RA+1:RA and placed in RA+1:RA when B is even. When B is odd, the contents of RB with its high order bit extended 8 high order bits is subtracted from RA+1:RA and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. SWF causes the condition code flags to be updated.

STATUS BITS:
NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (SWF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if there is a borrow; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC:
```
SW RDST,RSRC  ;SUB RDST FROM RSRC,
              ;DIF IN RSRC
```

ASSEMBLED OCTAL: 131144

BEFORE

(RDST) = 000001

(RSRC) = 000000

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

AFTER

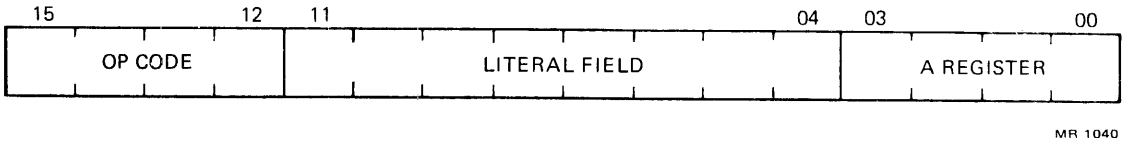
(RDST) = 000001

(RSRC) = 177777

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

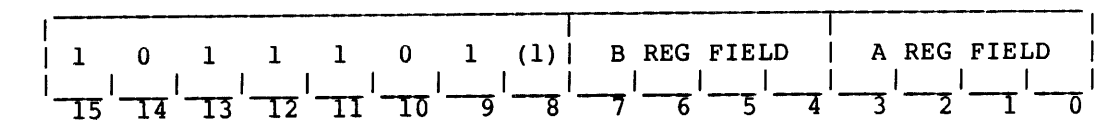
Literal Microinstruction Format

Figure 4-4



SWC (SWCF)

SUBTRAC WORD WITH CARRY (UPDATE CONDITION CODE FLAGS)



OPCODE: 135000-135377 (135400-135777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (RA+1:RA)-(RB+1:RB)-C
DESCRIPTION: The 16-bit contents of RB+1:RB minus the value of the C flag subtracted from the 16-bit contents of RA+1:RA and placed in RA+1:RA when B is even. When B is odd,the contents of RB with its high order bits minus the value of the C flag is subtracted from RA+1:RA and placed in RA+1:RA. The content of RB+1:RB is unchanged. The status bit flags are updated. SWCF causes the condition code flags to be updated.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(SWCF ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if there is a borrow; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: SWC RDST,RSRC ;SUB RDST, CARRY FROM RSRC,
;RESULT IN RSRC

ASSEMBLED OCTAL: 135144

BEFORE		AFTER	
(RDST) = 000001		(RDST) = 000001	
(RSRC) = 000000		(RSRC) = 177776	
NB ZB C4 C8 N Z V C		NB ZB C4 C8 N Z V C	
0 0 0 0 0 0 0 1		1 0 0 0 1 0 0 0	

4.3.2 Data Access Microinstructions

The Data Access Microinstructions provide the only means for transferring data into or out of the micromachine. The 16-bit transfer path is formed by the microprocessor Data chip WDAL connections which are bidirectionally buffered to the 16 LSI-11 system bus lines, BDAL <15:00>.

The LSI-11 machine input operation (DATI) is implemented by a Read microinstruction followed by an Input microinstruction. Similarly, the machine output operation (DATO) is implemented by a Write-Output sequence of microinstructions. These two basic sequences are available in seveval variations to accomodate Byte transfers (DATOB) and read-modify write operations (DATIO, DATIOB). Further variations allow for special handling of interrupt transactions.

It is the data access group of microinstructions (only) which activate the microprocessor control chip I/O control signal lines. The lines which carry control information to the LSI-11 system bus interface logic are WSYNC, WWB, WDIN, WDOUT, and WIACK. The REPLY and BUSY inputs to the Control chip are monitored during a data access operation. The logic circuitry which interfaces these signals to the LSI-11 system bus is discussed in The Microcomputer Handbook.

These microinstructions are organized as four basic families of Read, Input, Write and Output. Further sequencing and timing details are presented in Chapter 5.

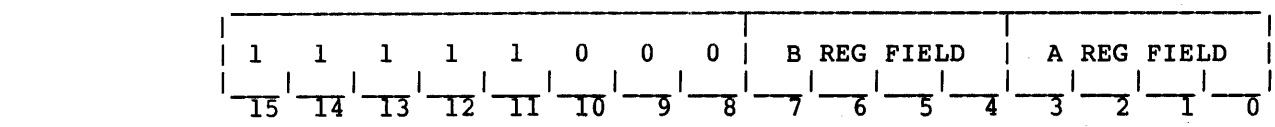
4.3.2.1 Read Microinstructions - The Read Microinstruction group consists of 6 microinstructions, all of which place a system bus device address on the data address lines. To facilitate sequential addressing, four microinstructions automatically increment the registers containing the address arguments. Because the A and B register ports may be read simultaneously, all read microinstructions place both the upper and lower Byte of the device address on BDAL <15:00> at the same time. All Read microinstructions except for those which increment an address word execute in one microcycle. However, before execution is initiated, the control chip checks REPLY H and BUSY H at PH3 to verify that no other system bus operations are still in progress. Because of the requirment to check REPLY H and BUSY H during PH3, the address information is not placed on the bus until the following microcycle.

The address information is placed on WDAL <15:00> during PH1 and is cancelled at the end of PH4, thus making the address valid for one microcycle. All read microinstructions assert WSYNC during PH2 of the microcycle in which the address becomes valid. The Read Acknowledge (RA) microinstruction asserts WIACK simultaneously with WSYNC. In all cases WDIN is asserted during the PH4 which follows removal of the address data. None of the read group microinstructions effect the condition code flags.

The microinstruction descriptions are as follows:

R

READ



OPCODE: 174000-174377
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The contents of RB:RA are unchanged and the flags are unaffected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The read microinstruction is the first part of a DATI, DATIO, or DATIOB system bus cycle.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

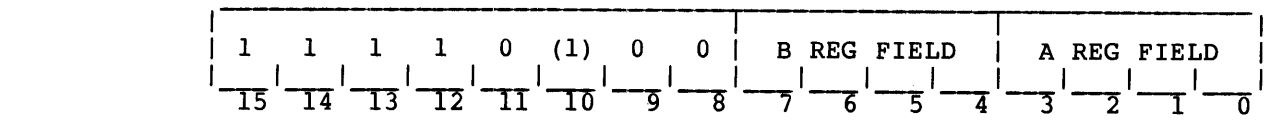
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: R     RBAH,RBAL  ;START DATI, ADDRESS IN RBA
                   IW    TG8,RSRC   ;DATA INTO RSRC
```

RIB1 (RIB2)

READ AND INCREMENT BYTE BY ONE (BY TWO)



OPCODE: 170000-170377 (172000-172377)
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
(RA) <-- (RA)+1
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The content of RB is unchanged but the content of RA is incremtented by 1. RIB2 causes the content of RA to be incremented by 2. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The RIB1(2) microinstruction is the first part of a DATI, DATIO, or DATIOB system bus cycle.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

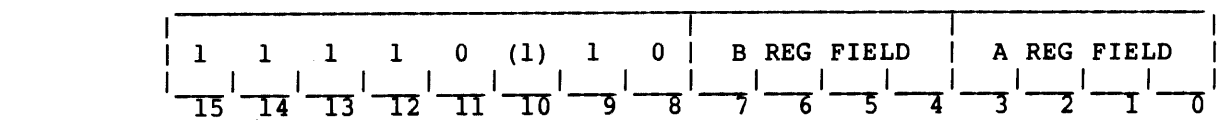
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC:
```
RIB2  RBAH,RBAL  ;START DATI, ADDRESS IN RBA
IW    TG8,RSRC   ;DATA INTO SOURCE REGISTER
```

RIW1 (RIW2)

READ AND INCREMENT WORD BY ONE (BY TWO)



OPCODE: 171000-171377 (173000-173777)
MICROCYCLES: 2 (MIN)
OPERATION: (DAL) <-- (RB:RA)
(RB:RA) <-- (RB:RA)+1 (or +2)

DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The 16-bit word content of registers RB:RA is incremented by 1. RIW2 causes the 16-bit word content of registers to be incremented by 2. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The RIW1(2) microinstruction may be the first part of a DATI, DATIO, or DATIOB system bus cycle.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

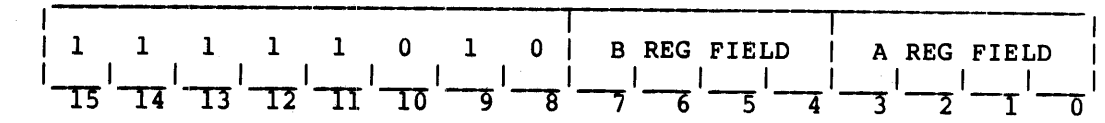
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: RIW2  RBAH,RBAL  ;START DATI, ADDRESS IN RBA
                   IW    TG8,RSRC   ;DATA INTO SOURCE REGISTER
```

RA

READ ACKNOWLEDGE



OPCODE: 175000-175377
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The contents of RB:RA are unchanged and the flags are unaffected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The read acknowledge microinstruction is the first part of an input operation executed with the WIACK line asseted HIGH.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

4.3.2.2 Input Microinstructions - A member of the Input Microinstruction group completes the basic DATI or Read-Input operation. WDIN is asserted during PH4 following the removal of address information from the bus. This signal is interfaced to the system bus as BDIN L, which informs the addressed device to place its data on the bus for input to the microprocessor. In order for the input microinstruction to be executed to completion, WDINH and REPLY H must be asserted during PH3. Since only the A port register is capable of writing register contents, input word microinstructions require 2 microcycles to execute.

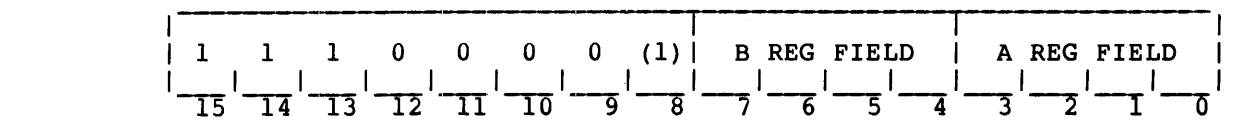
In some instances it is desirable to have the input operations execute to completion independent of the state of the REPLY H signal. The input status microinstructions execute immediately without being preceded by a read operation and without checking the REPLY H signal.

It should be noted that of all the possible Input Microinstructions, only the Input Word (IW) Microinstruction can load the translation register (when the B register field contains the proper code). The B field argument of the Input Byte (IB) and Input Word (IW) microinstruction can also specify a Read-Modify-Write operation. All of the Input Microinstructions are available in a flag-affecting version.

The microinstruction descriptions are as follows:

IB (IBF)

INPUT BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 160000-160377 (160400-160777)

MICROCYCLES: 1 (MIN)
OPERATION: (RA) <-- (DAL<15:8> or DAL<7:0>)
DESCRIPTION: The 8-bit byte from the system bus data lines is placed in register RA. IBF causes the condition code flags to be updated. The Read-Input Data Access operation is terminated unless MI<6> is a 1 which causes a Read-Modify-Write operation requiring termination by an Output operation. This microinstruction will not execute until the REPLY signal has been received from the I/O device.

The input options are specified by the B argument.

(B) = 0 Upper Byte, DAL<15:8>
(B) = 1 Lower Byte, DAL<7:0>
(B) = 2 Upper Byte if M(0)=1, Lower Byte if M(0)=0
(B) = 3 Lower Byte if M(0)=1, Upper Byte if M(0)=0
(B) = 4 Upper Byte, DAL<15:8>, RMW
(B) = 5 Lower Byte, DAL<7:0>, RMW
(B) = 6 Upper Byte if M(0)=1, Lower Byte if M(0)=0, RMW
(B) = 7 Lower Byte if M(0)=1, Upper Byte if M(0)=0, RMW

NOTE: M(0) is defined as the least significant bit of the address transferred to the DAL by the previous Read microinstruction.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

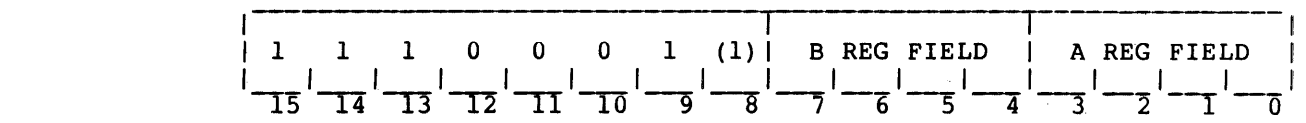
CONDITION CODES: N: set if byte result < 0; cleared otherwise
(IBF ONLY) Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: IB  UB,RSRCL  ;INPUT UPPER BYTE INTO RSRCL
```

IW (IWF)

INPUT WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 161000-161377 (161400-161777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (DAL<15:0>)
DESCRIPTION: The 16-bit word from the system bus data lines is placed in registers RA+1:RA. IWF causes the condition code flags, except C, to be updated. The Read-Input Data Access operation is terminated unless MI<6> is a 1 which allows a Read-Modify-Write. The loading of the translation register and the G register is controlled by MI<5> and MI<4>, respectively, as indicated below. This microinstruction will not execute until the REPLY signal has been received from the I/O device. The lower byte is loaded before the upper byte.

The input options are specified by the B argument.

(B) = 0 Load Designated Registers Only
(B) = 1 Load TR, DAL<6:4> TO G, Set ICC Flag
(B) = 2 Load TR, DAL<8:6> TO G, Set ICC Flag
(B) = 3 Load TR, Set ICC Flag
(B) = 4 READ-MODIFY-WRITE (RMW)
(B) = 5 Load TR, DAL<6:4> TO G, Set ICC Flag, RMW
(B) = 6 Load TR, DAL<8:6> TO G, Set ICC Flag, RMW
(B) = 7 Load TR, Set ICC Flag, RMW

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

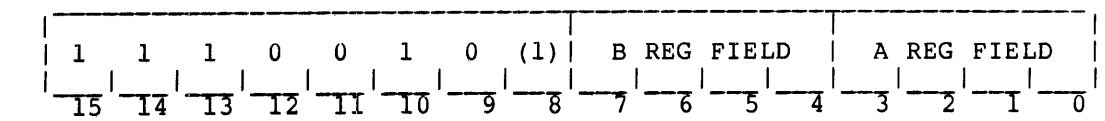
CONDITION CODES: N: set if byte result < 0; cleared otherwise
(IWF ONLY) Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: IW  TG8,RSRCL  ;INPUT WORD INTO RSRCL
                                  ;LOAD TR, LOAD <8:6> TO G
```

ISB (ISBF)

INPUT STATUS BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 162000-162377 (162400-162777)
MICROCYCLES: 1
OPERATION: (RA) <-- (DAL<15:8>) (or DAL<7:0>)
DESCRIPTION: The 8-bit byte from the system bus data lines is placed in register RA. ISBF causes the condition code except C, flags to be updated. This microinstruction will execute regardless of the state of the REPLY or BUSY signals.

The input options are specified by the B argument.

(B) = 0 or 4 Upper Byte, DAL<15:8>
(B) = 1 or 5 Lower Byte, DAL<7:0>
(B) = 2 or 6 Upper Byte if M(0)=1, Lower Byte if M(0)=0
(B) = 3 or 7 Lower Byte if M(0)=1, Upper Byte if M(0)=0

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

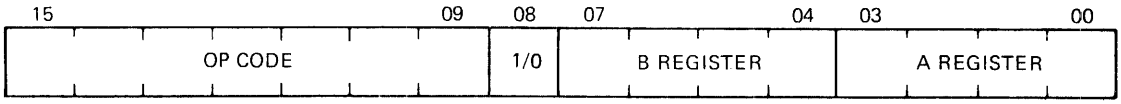
CONDITION CODES: N: set if byte result < 0; cleared otherwise
(ISBF ONLY) Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: ISB UB,RSRCL ;INPUT UPPER BYTE INTO RSRCL

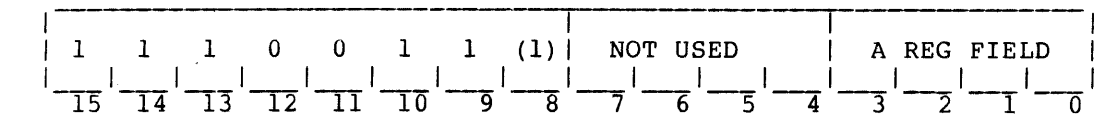
Register Microinstruction Format

Figure 4-5



ISW (ISWF)

INPUT STATUS WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 163000-163377 (163400-163777)
MICROCYCLES: 2
OPERATION: (RA+1:RA) <-- (DAL<15:0>)
DESCRIPTION: The 16-bit word from the system bus data lines is placed in registers RA+1:RA. ISWF causes the condition code flags, except C, to be updated. This microinstruction will execute regardless of the state of the BUSY or REPLY signals. The lower byte is loaded before the upper byte.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: (ISWF ONLY)
N: set if byte result < 0; cleared otherwise
Z: set if byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

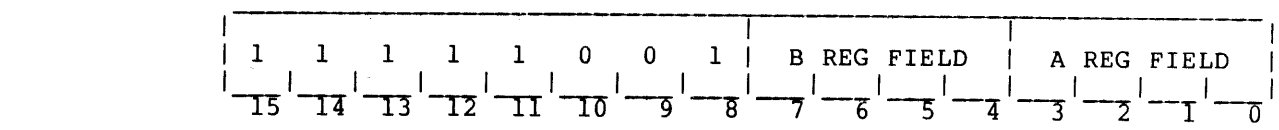
```
ASSEMBLY MNEMONIC: ISWF     ,RSRCL  ;INPUT WORD INTO RSRCL
                                    ;LOAD TR, LOAD <8:6> TO G
```

4.3.2.3 Write Microinstructions - The Write Microinstructions are the first part of the Write-Output sequence which implements the DATO (Data-Output) transactions. Because the A and B register ports may be read simultaneously, Write Microinstructions place both the upper and lower byte of the device address on BDAL <15:00> at the same time. All Write Microinstructions, except for those which increment an address word, execute in one microcycle. However, before execution is initiated, the Control chip checks REPLY H and BUSY H at PH3 to verify that no other system bus operations are in progress. Because of the requirement to check REPLY H and BUSY H during PH3, the address information cannot be placed on the bus until the following microcycle. The address information is placed on BDAL <15:00> during PH1 and is cancelled at the end of PH4, thus making the address valid for one microcycle. All Write Microinstructions assert WSYNC during PH2 of the microcycle in which the address is valid. The Write Acknowledge (WA) microinstruction asserts WIACK H simultaneously with WSYNC.

The microinstruction descriptions are as follows:

W

WRITE



OPCODE: 174400-174777
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The contents of RB:RA are unchanged and the flags are unaffected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The write microinstruction is the first part of a DATO or DATOB system bus cycle.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

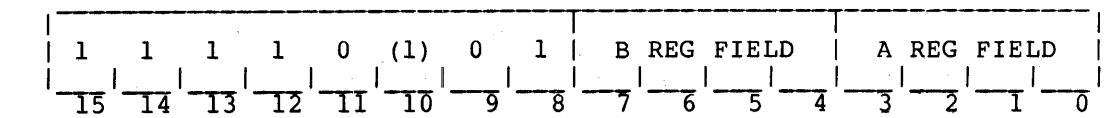
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: W      RBAH,RBAL   ;START DATO, ADDRESS IN RBA
                   OW     RDSTH,RDSTL ;DATA FROM SOURCE REGISTER
```

WIB1 (WIB2)

WRITE AND INCREMENT BYTE BY ONE (BY TWO)



OPCODE: 170400-170777 (172400-172777)
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
(RA) <-- (RA)+1 (or+2)

DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The content of RB is unchanged but the content of RA is incremented by 1. WIB2 causes the content of RA to be incremented by 2. The flags are not affected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The WIB1(2) microinstruction is the first part of a DATO or DATOB system bus cycle.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry = 1;cleared otherwise
C8: set if carry out = 1; cleared otherwise

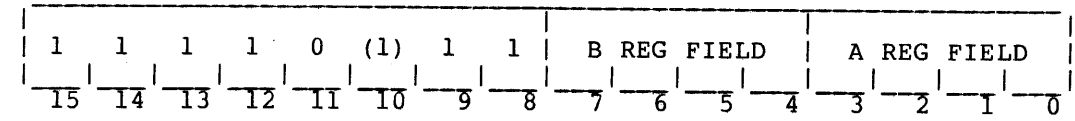
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC:
```
WIB2  RBAH,RBAL   ;START DATO, ADDRESS IN RBA
OW    RDSTH,RDSTL ;DATA INTO SOURCE REGISTER
```

WIW1 (WIW2)

WRITE AND INCREMENT WORD BY ONE (BY TWO)



OPCODE: 171400-171777 (173400-173777)
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
(RB:RA) <-- (RB:RA)+1 (or+2)

DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The 16-bit word content of registers RB:RA is incremented by 1. WIW2 causes the 16-bit word content of the registers to be incremented by 2. The flags are not affected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The WIW1(2) microinstruction is the first part of a DATO or DATOB system bus cycle.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry = 1;cleared otherwise
C8: set if carry out = 1; cleared otherwise

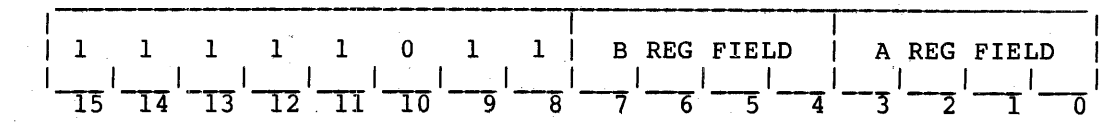
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: WIB2  RBAH,RBAL   ;START DATO, ADDRESS IN RBA
                   OW    RDSTH,RDSTL ;DATA FROM SOURCE REGISTER
```

WA

WRITE ACKNOWLEDGE



OPCODE: 175400-175777
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as a system bus address. The contents of RB:RA are unchanged and the flags are unaffected. This microinstruction will execute only when the BUSY and REPLY lines are tested low at PH3. The write acknowledge microinstruction is the first part of an input operation executed with the WIACK line asseted HIGH. However, the circuitry on the LSI-11 module does not allow BIACK L to be asserted for this "acknowledge" microinstruction.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

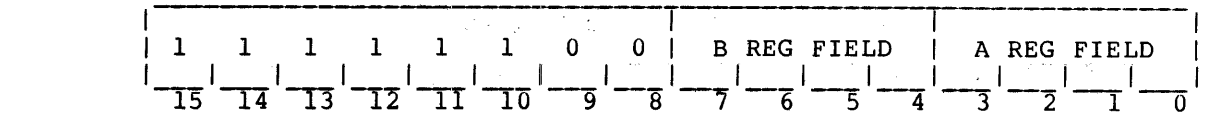
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

4.3.2.4 Output Microinstructions - An Output Microinstruction is used to complete the basic DATO or Write/Output operation, or to complete a Read-Modify-Write operation (DATIO or DATIOB). WDOUT is asserted by an Output Microinstruction during PH1 following the removal of address information from the bus. This signal is interfaced to the system bus as BDOUT L, which causes the addressed device to accept the data placed on the bus by the microprocessor Data chip. In order for an Output Microinstruction to be completed, REPLY H must be asserted by the addressed device during PH3. Both register A and B ports provide for reading or accessing register contents, so for Output Word and Output Status Word Microinstructions, all 16-bits can be output during one microcycle. In some instances it is desirable to complete an output operation independent of the state of the REPLY H signal. The Output Status Microinstructions execute immediately without being preceded by a Write Microinstruction and without checking the REPLY H signal.

The microinstruction descriptions are as follows:

OB

OUTPUT BYTE



OPCODE: 176000-176377
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as output data. The WDOUT signal is asserted. The content of registers RB:RA is not changed. Because of the 16-bit system bus data path, B must equal A so that the same byte is placed in both positions. The OB microinstruction will not execute to completion until the REPLY signal has been received.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

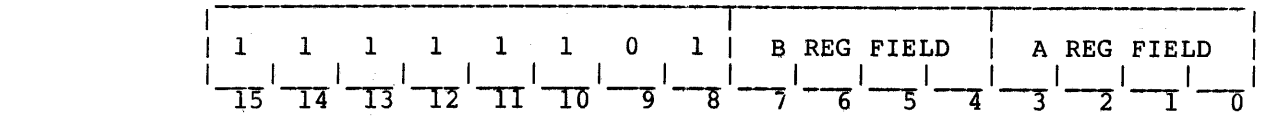
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: OB RDSTL,RDSTL ;OUTPUT BYTE FROM RDSTL

OW

OUTPUT WORD



OPCODE: 176400-176777
MICROCYCLES: 1 (MIN)
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as output data. The WDOUT signal is asserted. The content of registers RB:RA is not changed. The OW microinstruction will not execute to completion until the REPLY signal has been received.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

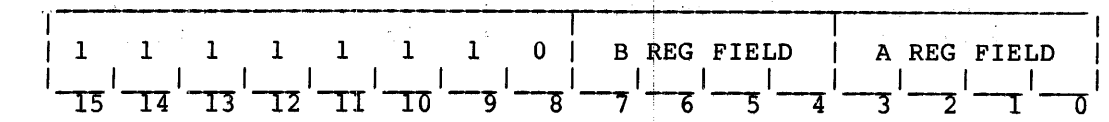
CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: OW RDSTH,RDSTL ;OUTPUT WORD FROM RDST

OS

OUTPUT STATUS



OPCODE: 177000-177377
MICROCYCLES: 1
OPERATION: (DAL) <-- (RB:RA)
DESCRIPTION: The 16-bit contents of RB:RA are placed on the system bus as output data. The content of registers RB:RA is not changed. The OS microinstruction will execute to completion regardless of the state of the BUSY OR REPLY signals. I/O circuitry interfaced to the system bus may be sychronized with this operation by means of the TTL Control bits, MI<23:18>.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: OS RDSTH,RDSTL ;OUTPUT WORD FROM RDST

4.3.3 Microprogram Control Microinstructons

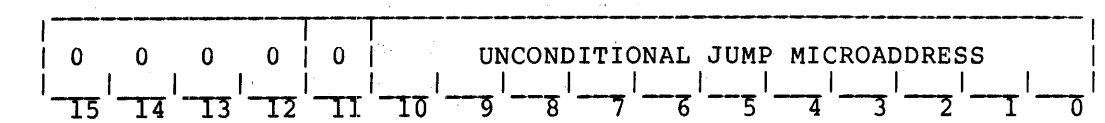
The Microprogram Control Microinstructions control or modify the microinstruction flow. The only available control methods not covered by this group are (1) the RSVC bit (Read Next Instruction/Interrupt and Trap Service) and (2) the microprocessor Control chip RESET line.

4.3.3.1 Jump and Return Microinstructions - This group contains all microinstructions which alter the microinstruction flow (both conditionally and unconditionally). The Modify Microinstruction (MI) - Jump (JMP) sequence allows the microprogrammer to determine the jump microaddress from data in one of the micromachine registers.

The microinstruction descriptions are as follows:

JMP

JUMP (UNCONDITIONALLY)



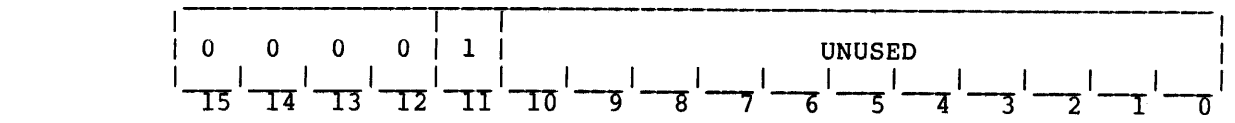
OPCODE: 000000-003777
MICROCYCLES: 2
OPERATION: (LC) <-- (MI<10:0>)
DESCRIPTION: The 11-bit microaddress stored in this microinstruction is placed in the Control Chip Location Counter. The next microinstruction to be executed will be fetched from the new location. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Translations programmed in the translation array are ignored. No other register or flag contents are changed.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

RFS

RETURN FROM SUBROUTINE



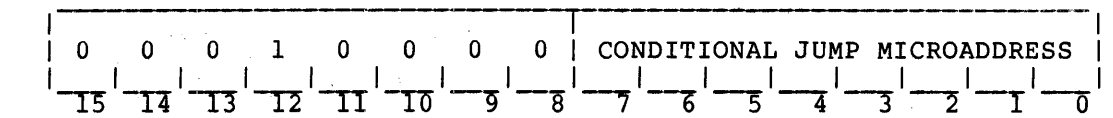
OPCODE: 004000-007777
MICROCYCLES: 2
OPERATION: (LC) <-- (RR<10:0>)
DESCRIPTION: The 11-bit microaddress stored in the return register is placed in the Control chip Location Counter. The Return Register is usually loaded at the time a JMP microinstruction is executed. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Translations programmed in the translation array are ignored. No other register or flag contents are changed.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JZBF

JUMP IF STATUS BIT FLAG ZB IS ZERO



OPCODE: 010000-010377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF ZB = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (ZB=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <7:0> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

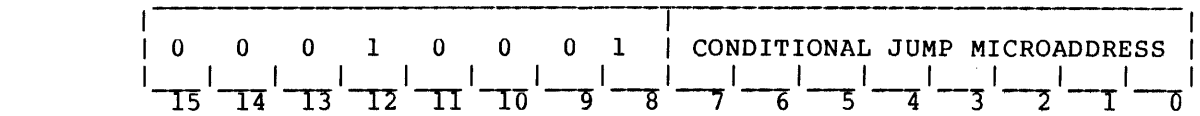
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JZBT

JUMP IF STATUS BIT FLAG ZB IS ONE



OPCODE: 010400-010777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF ZB = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (ZB=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <7:0> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

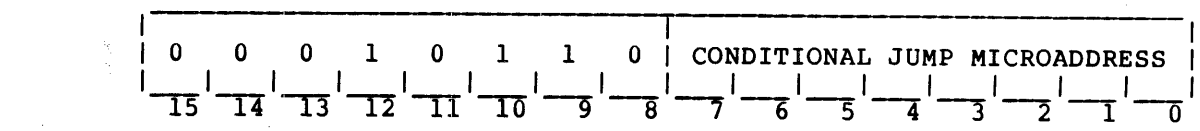
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JNBF

JUMP IF STATUS BIT FLAG NB IS ZERO



OPCODE: 013000-013377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF NB = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (NB=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <7:0> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

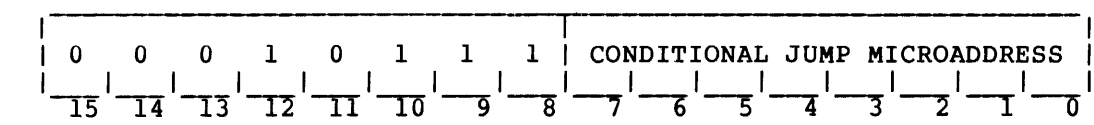
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JNBT

JUMP IF STATUS BIT FLAG NB IS ONE



OPCODE: 013400-013777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF NB = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (NB=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

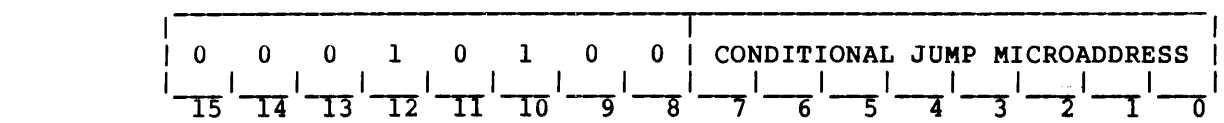
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JIF

JUMP IF INDIRECT CONDITION CODE IS ZERO



OPCODE: 012000-012377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF ICC = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (ICC=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

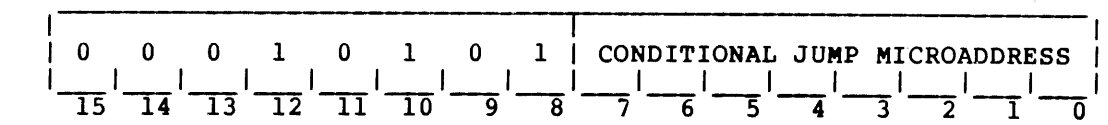
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JIT

JUMP IF INDIRECT CONDITION CODE IS ONE



OPCODE: 012400-012777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF ICC = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (ICC=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

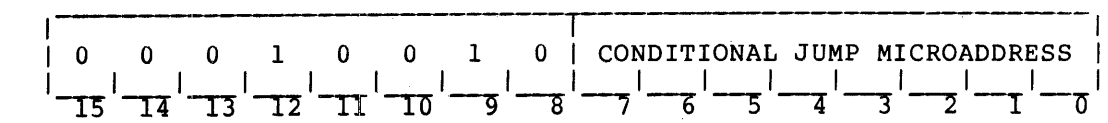
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JC8F

JUMP IF STATUS BIT FLAG C8 IS ZERO



OPCODE: 011000-011377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF C8 = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (C8=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

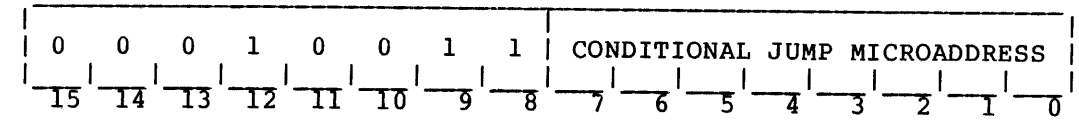
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JC8T

JUMP IF STATUS BIT FLAG C8 IS ONE



OPCODE: 011400-011777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF C8 = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (C8=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

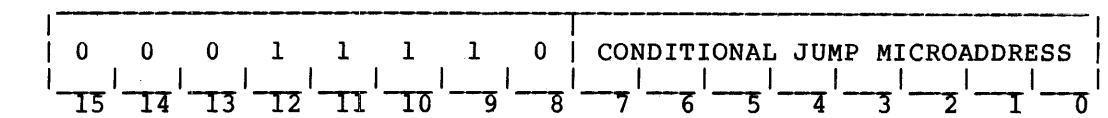
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JNF

JUMP IF CONDITION CODE FLAG N IS ZERO



OPCODE: 017000-017377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF N = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (N=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

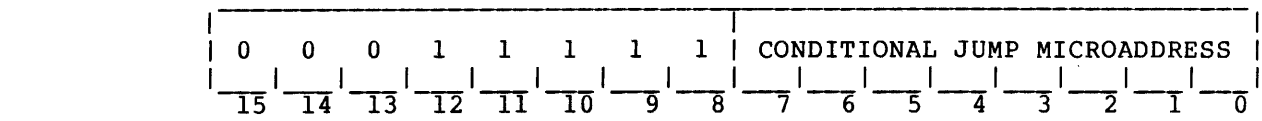
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JNT

JUMP IF CONDITION CODE FLAG N IS ONE



OPCODE: 017400-017777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF N = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (N=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

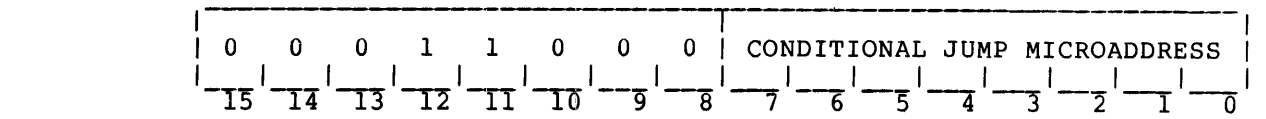
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JZF

JUMP IF CONDITION CODE FLAG Z IS ZERO



OPCODE: 014000-014377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF Z = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (Z=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

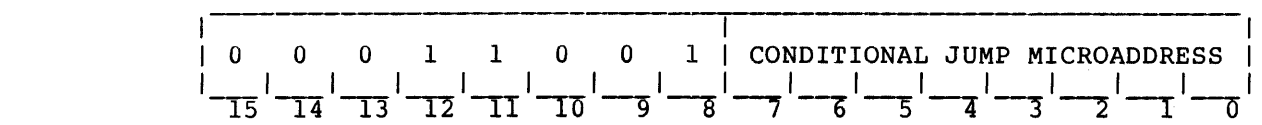
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JZT

JUMP IF CONDITION CODE FLAG Z IS ONE



OPCODE: 014400-014777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF Z = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (Z=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

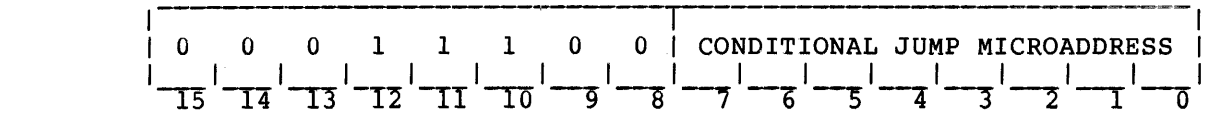
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JVF

JUMP IF CONDITION CODE FLAG V IS ZERO



OPCODE: 016000-016377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF V = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (V=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

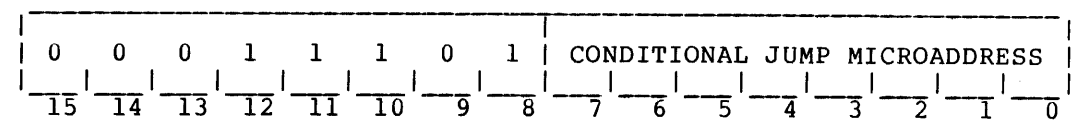
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JVT

JUMP IF CONDITION CODE FLAG V IS ONE



OPCODE: 016400-016777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF V = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (V=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

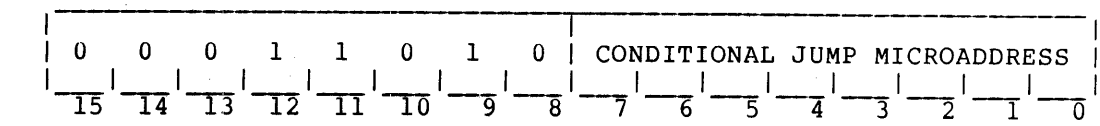
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JCF

JUMP IF CONDITION CODE FLAG C IS ZERO



OPCODE: 015000-015377
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF C = 0
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (C=0) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

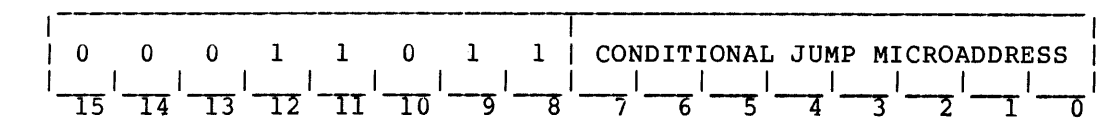
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

JCT

JUMP IF CONDITION CODE FLAG C IS ONE



OPCODE: 015400-015777
MICROCYCLES: 2
OPERATION: (LC<7:0>) <-- (MI<7:0>), IF C = 1
DESCRIPTION: The 8-bit microaddress stored in this microinstruction is placed in the Control chip Location Counter if the condition (C=1) is met. Execution of this microinstruction requires 2 microcycles because the normal microaddress generation sequence is modified. Bits <0:7> of translations programmed in the translation array are ignored. No other register or flag contents are changed.

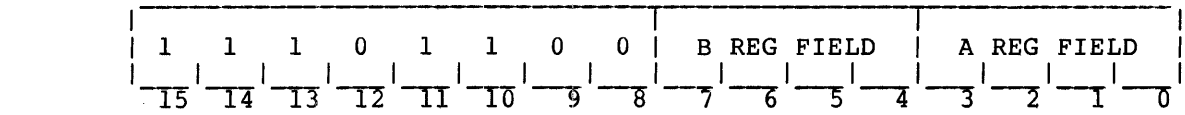
Note that the upper 3 bits of the Location Counter, LC<10:8>, remain unchanged, pointing to a 256-word page from which the next microinstruction is fetched. The page number is determined at the beginning of the first microcycle of execution at the time the microaddress of the next sequential microinstruction is loaded in the Location Counter. This microaddress will be used during microfetch if the condition is not met.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

MI

MODIFY MICROINSTRUCTION



OPCODE: 166000-166377
OPERATION: (MIB<15:0>) <-- (RB:RA), With Next Microinstruction
DESCRIPTION: The 16-bit contents of registers RB:RA are ORed with the contents of the next sequential control store location to form the next microinstruction. During the execution of the MI microinstruction, the contents of registers RB:RA are placed on the Microinstruction Bus simultaneously with the next microinstruction from control store. The content of RB:RA is unchanged.

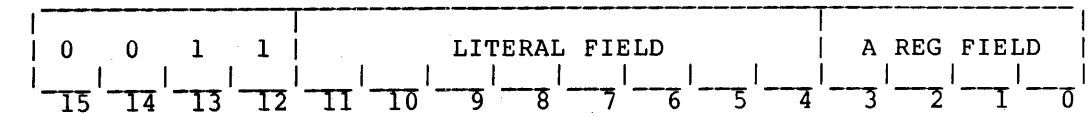
STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

4.3.3.2 Compare and Test Microinstructions - This group consists of five arithmetic compare and five logical test microinstructions.

The microinstruction descriptions are as follows:

COMPARE LITERAL



OPCODE: 03XXXX
MICROCYCLES: 1
OPERATION: (RA)-(LITERAL FIELD), Set Status Bit Flags
DESCRIPTION: The 8-bit contents of the literal field, MI<11:4>, are subtracted from the 8-bit contents of RA and the result sets the NB, ZB, C4, and C8 status bit flags. The contents of the literal field and RA remain unchanged.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES:
N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: CL 200,RBAL ;COMPARE RBAL WITH 200

ASSEMBLED OCTAL: 034002

BEFORE

(RBAL) = 200

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

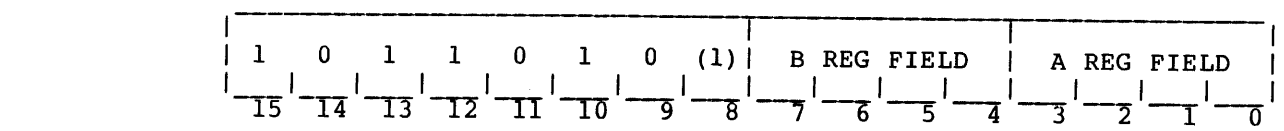
AFTER

(RBAL) = 200

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	0	0	0

CB (CBF)

COMPARE BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 132000-132377 (132400-132777)
MICROCYCLES: 1
OPERATION: (RA-(RB) SETS STATUS BIT FLAGS
DESCRIPTION: The 8-bit contents of RB are subtracted from the 8-bit contents of RA and the result sets status bit flags NB, ZB, C4, and C8. CBF causes the condition code flags to be updated. The contents of the designated registers remain unchanged in all cases.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: (CBF ONLY)
N: set if the word result < 0; cleared otherwise
Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

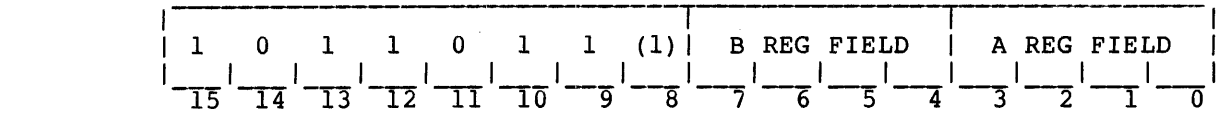
ASSEMBLY MNEMONIC: CB RDSTL,RSRCL ;COMPARE RDSTL WITH RSRCL

ASSEMBLED OCTAL: 132144

BEFORE								AFTER							
(RDSTL) = 000								(RDSTL) = 000							
(RSRCL) = 377								(RSRCL) = 377							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

CW (CWF)

COMPARE WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 133000-133377 (133400-133777)
MICROCYCLES: 2
OPERATION: (RA+1:RA)-(RB+1:RB) SETS STATUS BIT FLAGS
DESCRIPTION: The 16-bit contents of RB+1:RB are subtracted from the 16-bit contents of RA+1:RA and the result sets the NB, ZB, C4, and C8 status bit flags. op code 267 (CWF) causes the condition code flags to be updated. The contents of the desginated registers remain unchanged in all cases.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: set if bit<3> carry out = 1; cleared otherwise
C8: set if carry out = 1; cleared otherwise

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(CWF ONLY) Z: set if the word result = 0; cleared otherwise
V: set if arithmetic overflow; cleared otherwise
C: set if carry out = 1; cleared otherwise

EXAMPLE:

ASSEMBLY MNEMONIC: CWF RDST,RSRC ;COMPARE RDST WITH RSRC

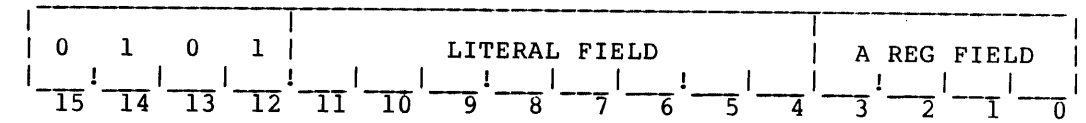
ASSEMBLED OCTAL: 133544

BEFORE	AFTER
(RDST) = 000001	(RDST) = 000001
(RSRC) = 000001	(RSRC) = 000001

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

NB	ZB	C4	C8	N	Z	V	C
0	1	0	0	0	1	0	0

TEST LITERAL



OPCODE: 05XXXX
MICROCYCLES: 1
OPERATION: (RA"AND"(LITERAL FIELD) SETS STATUS BIT FLAGS
DESCRIPTION: The 8-bit contents of the literal field, MI<11:4>, are ANDED with the 8-bit contents of RA and the result sets the NB, ZB, C4, and C8 status bit flags. The contents of the literal field and the designated register remain unchanged.

STATUS BITS: NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: TL 200,RBAL ;"AND" RBAL WITH 200, SET FLAGS

ASSEMBLED OCTAL: 054002

BEFORE

(RBAL) = 377

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

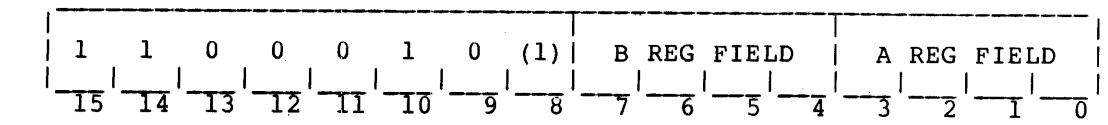
AFTER

(RBAL) = 377

NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	0	0	0	0

TB (TBF)

TEST BYTE (UPDATE CONDITION CODE FLAGS)



OPCODE: 142000-142377 (142400-142777)
MICROCYCLES: 1
OPERATION: (RB)"AND"(RA) SETS STATUS BIT FLAGS
DESCRIPTION: The 8-bit contents of RB are ANDed with the 8-bit contents of RA and the result sets the NB, ZB, C4, and C8 status bit flags. Op code 305 (TBF) causes the condition code flags to be updated. The contents of the designated registers remain unchanged in all cases.

STATUS BITS:
NB: set if byte result < 0; cleared otherwise
ZB: set if byte result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: TBF (ONLY)
N: set if the byte result < 0; cleared otherwise
Z: set if the byte result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

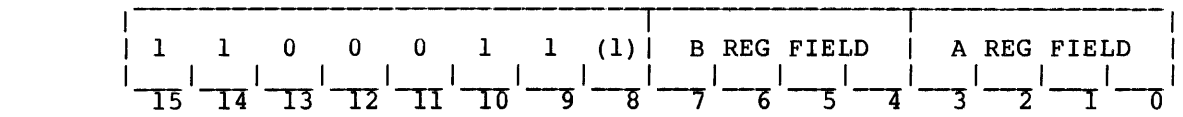
ASSEMBLY MNEMONIC: TBF RDSTL,RSRCL ;"AND" RDSTL WITH RSRCL, SET FLAGS

ASSEMBLED OCTAL: 142544

BEFORE								AFTER							
(RDSTL) = 201								(RDSTL) = 201							
(RSRCL) = 176								(RSRCL) = 176							
NB	ZB	C4	C8	N	Z	V	C	NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

TW (TWF)

TEST WORD (UPDATE CONDITION CODE FLAGS)



OPCODE: 143000-143377 (143400-143777)
MICROCYCLES: 2
OPERATION: (RB+1:RB)"AND"(RA+1:RA) SETS STATUS BIT FLAGS
DESCRIPTION: The 16-bit contents of RB+1:RB are ANDed with the 16-bit contents of RA+1:RA and the result sets the NB, ZB, C4, and C8 status bit flags. TWF causes the condition code flags to be updated. The contents of the designated registers remain unchanged in all cases.

STATUS BITS: NB: set if word result < 0; cleared otherwise
ZB: set if word result = 0; cleared otherwise
C4: not affected
C8: not affected

CONDITION CODES: N: set if the word result < 0; cleared otherwise
(TWF ONLY) Z: set if the word result = 0; cleared otherwise
V: cleared
C: not affected

EXAMPLE:

ASSEMBLY MNEMONIC: TW RDST,RSRC ;"AND" RDST WITH RSRC, SET FLAGS

ASSEMBLED OCTAL: 143144

BEFORE	AFTER
(RDST) = 177777	(RDST) = 177777
(RSRC) = 177777	(RSRC) = 177777

NB	ZB	C4	C8	N	Z	V	C
0	0	0	0	0	0	0	0

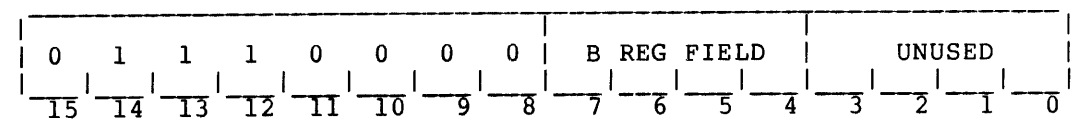
NB	ZB	C4	C8	N	Z	V	C
1	0	0	0	0	0	0	0

4.3.3.3 Miscellaneous Control Microinstructions - This group of microinstructions affects control of the interrupt flags, and the Translation State Register (TSR) contents. Also included here is the No Operation (NOP) microinstruction.

The microinstruction descriptions are as follows:

RI

RESET INTERRUPTS



OPCODE: 070000-070377
MICROCYCLES: 1
OPERATION: I4 <-- COMPLEMENT OF (MI<4>),
I5 <-- COMPLEMENT OF (MI<5>),
I6 <-- COMPLEMENT OF (MI<7>)

DESCRIPTION: The Control chip internal interrupts are reset under control of the B register field contents. The A register field is unused. The status bit and condition code flags are not affected.

(B) = 01, RESET I4 TRACE TRAP
(B) = 02, RESET I5 COMPLEMENT OF PSW<7>
(B) = 04, RESET I6 EXTERNAL INTERRUPT TEST
(B) = 10, UNUSED

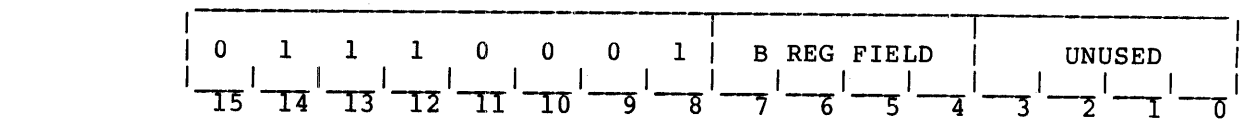
The B register field may reset any combination of the internal interrupts.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

SI

SET INTERRUPTS



OPCODE: 070400-070777
MICROCYCLES: 1
OPERATION: I4 <-- (MI<4>),
I5 <-- (MI<5>),
I6 <-- (MI<7>)

DESCRIPTION: The Control chip internal interrupts are set under control of the B register field contents. The A register field is unused. The status bit and condition code flags are not affected.

(B) = 01, SET I4 TRACE TRAP
(B) = 02, SET I5 COMPLEMENT OF PSW<7>
(B) = 04, SET I6 EXTERNAL INTERRUPT TEST
(B) = 10, UNUSED

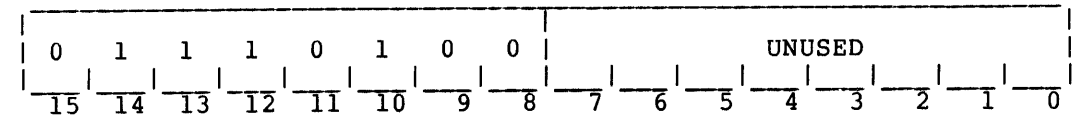
The B register field may set any combination of the internal interrupts.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

RTSR

RESET TRANSLATION STATE REGISTER



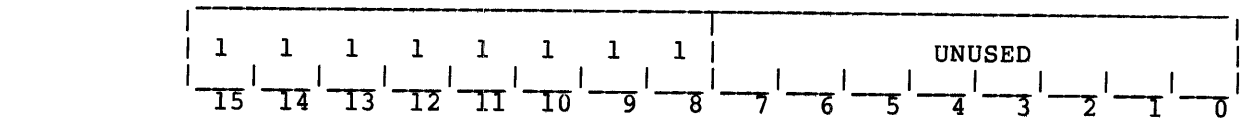
OPCODE: 072000-072377
MICROCYCLES: 1
OPERATION: (TSR) <-- 0
DESCRIPTION: Execution of this microinstruction resets the three bits in the translation state register (TSR). The TSR should always contain 0 when returning control back to the LSI-11 machine cycle. Under normal conditions, the TSR will contain 0 when control is passed to user control store.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

NOP

NO OPERATION



OPCODE: 177400-177777
MICROCYCLES: 1
OPERATION: NO OPERATION
DESCRIPTION: Execution of this microinstruction causes no change to the register file or flag contents, nor does it cause a change in the microinstruction flow. It is useful for inserting a one-microcycle delay in microprogram execution.

STATUS BITS: NB: not affected
ZB: not affected
C4: not affected
C8: not affected

CONDITION CODES: N: not affected
Z: not affected
V: not affected
C: not affected

EXAMPLE:

```
ASSEMBLY MNEMONIC: NOP          ;NO OPERATION

ASSEMBLED OCTAL:   177400

                    BEFORE                      AFTER

                      NO CHANGE   ONE MICROCYCLE DELAY
```

CHAPTER 5

MICROPROGRAMMING LSI-11 SYSTEM BUS TRANSACTIONS

5.1 GENERAL

The LSI-11 processor module uses a microprogramable microprocessor chip set to emulate a PDP-11. Since the LSI-11 bus is more complex than the microprocessor I/O structure, additional logic (external to the chip set), implements the LSI-11 bus. LSI-11 bus I/O operations are easy to microprogram, but this external logic must be understood to efficiently microprogram LSI-11 bus transactions. the Circuit Schematic diagram of the LSI-11 CPU should be referenced for the discussion to follow.

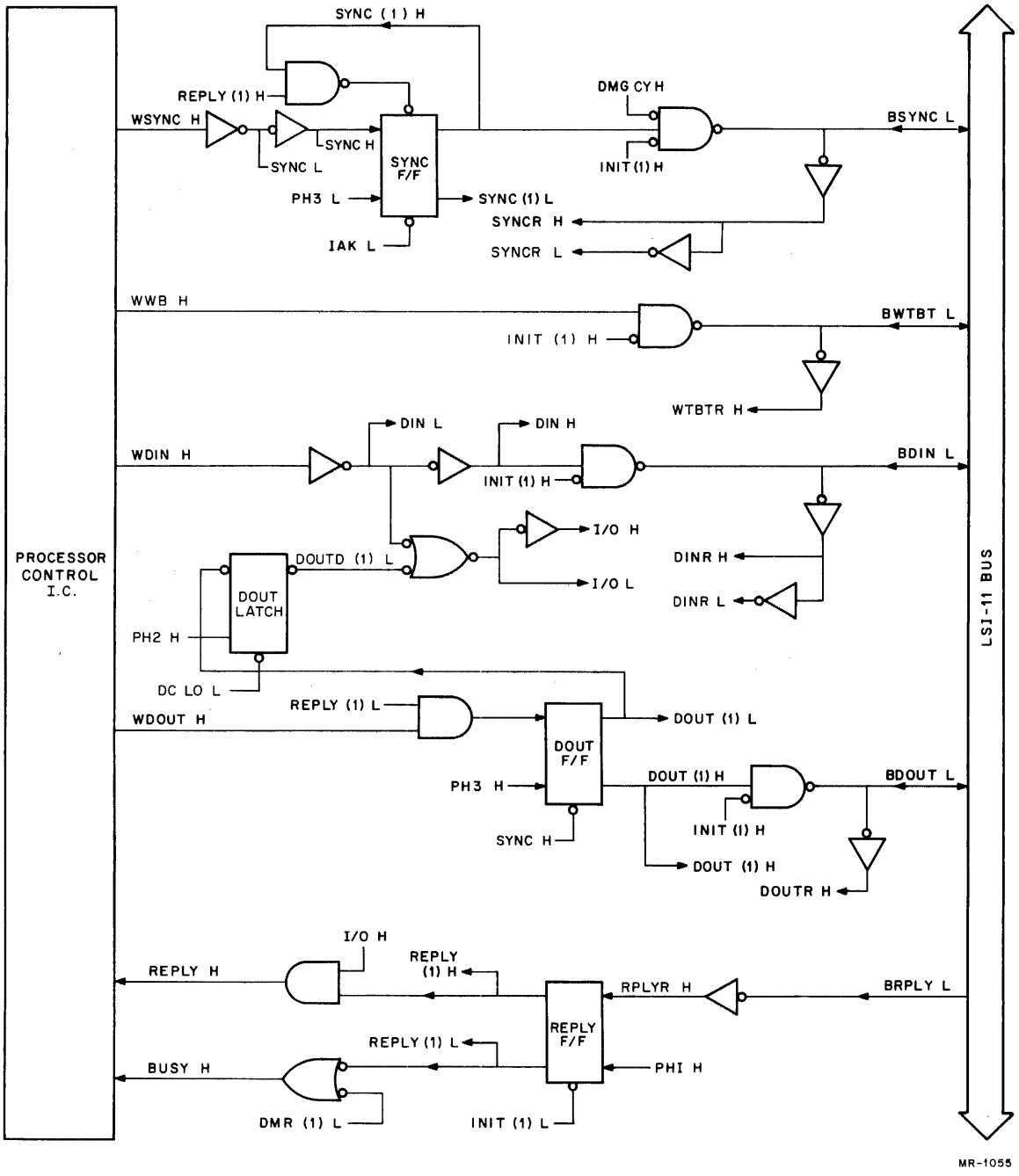
5.2 LSI-11 SYSTEM BUS INTERFACE LOGIC OVERVIEW

The LSI-11 processor module contains logic to interface the microprocessor to the LSI-11 system bus. This logic performs two functions: (1) it modifies the sequencing and timing of control signals which appear at the Control chip, and (2) it provides an electrical buffer between the MOS LSI microprocessor and the TTL system bus. Further discussion of this material is contained in The Microcomputer Handbook. Figure 5-1, Bus I/O Control Signal Logic and Figure 5-2, Interrupt Control and Reset Logic, should be referenced for this section.

This logic overview lists each control signal first in its original form and then in its final form. For example, the microprocessor originates WSYNC H which becomes the system bus signal BSYNC L.

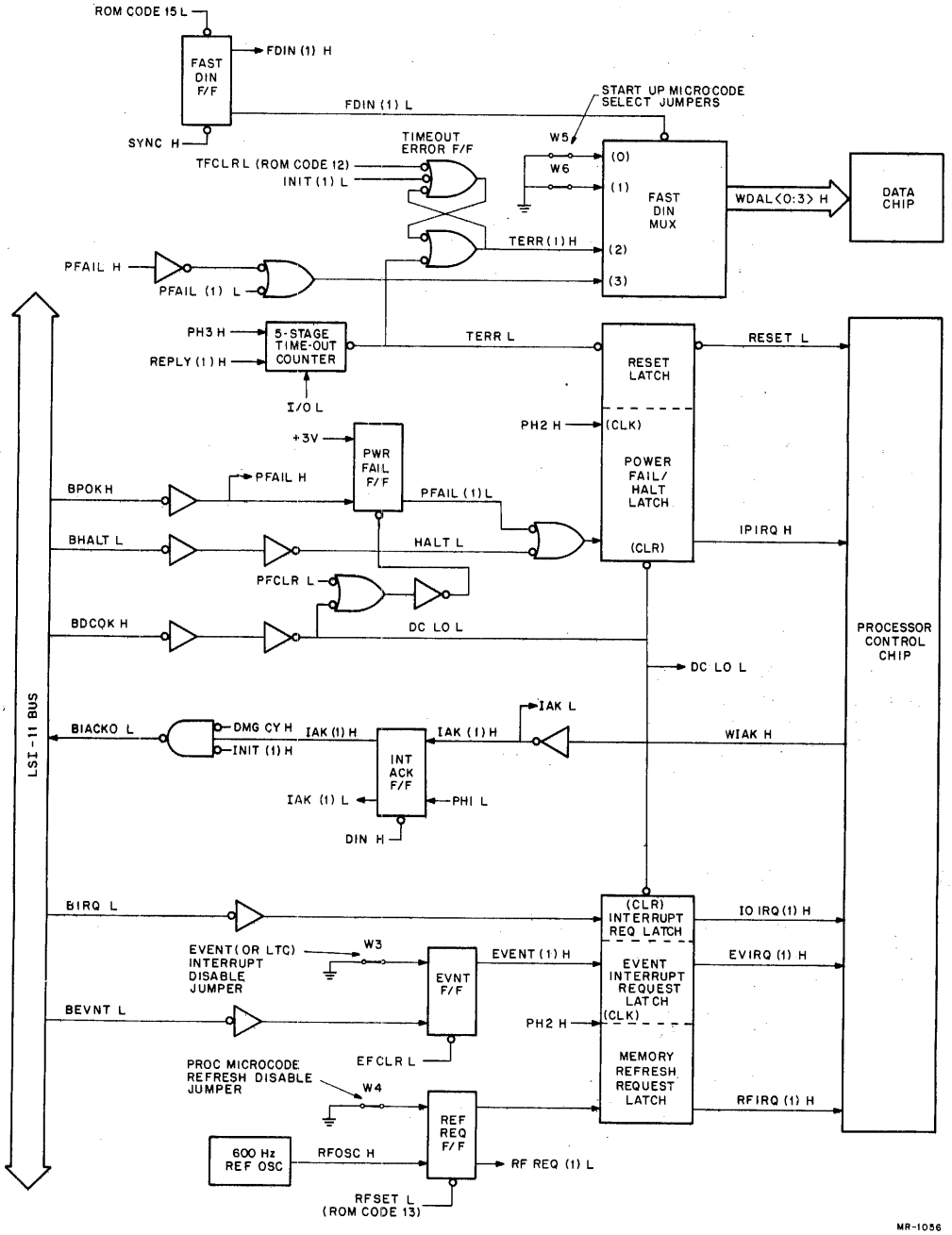
Bus I/O Control Signal Logic

Figure 5-1



Interrupt Control and Reset Logic

Figure 5-2



5.2.1 WSYNC H - BSYNC L

WSYNC H is asserted by the Control chip as part of the execution of any Read or Write microinstruction. In all cases, this signal appears at the beginning of PH2 following the microcycle in which the microinstruction was executed. The interface logic circuitry provides the following 3 functions:

-DELAYS the appearance of WSYNC H as BSYNC L until after PH3 by means of the SYNC Flip Flop (clocked by the trailing edge of PH3 L).

-MAINTAINS BSYNC L in the asserted state until after BRPLY L has been terminated by the slave device at the end of the transaction. This serves to keep the device recognition signal (on the device) stable until the device has completed its role in the bus transaction.

-INHIBITS the appearance of WSYNC H as BSYNC L during interrupt operations. The two acknowledge microinstructions, RA and WA, cause the Control chip to assert WSYNC H simultaneously with WIAK H. The WIAK H signal forces the SYNC F/F to the reset state which inhibits BSYNC L.

5.2.2 WWB H - BWTBT L

WWB H is asserted by the Control chip during the microcycle which places address information on the bus. This signal appears immediately as BWTBT L (delayed only by the inverting buffer propagation delay).

5.2.3 WDIN H - BDIN L

WDIN H is asserted by the Control chip during the microcycle following the transmission of address information. All Read microinstructions cause this signal to be asserted at the beginning of PH2. WDIN H appears immmediately as BDIN L (delayed only by two invertors and the inverting buffer propagation delay).

5.2.4 WDOUT H - BDOUT L

WDOUT H is asserted by the Control chip at the beginning of the microcycle which follows transmission of address information. All Output microinstructions assert this signal asserted at the beginning of PH1. WDOUT H is also asserted as part of the DATIO bus transaction. This logic provides the following 4 functions:

-DELAYS the appearance of WDOUT H as BDOUT L until after PH2, by means of the DOUT F/F (which is clocked by the leading edge of PH3).

-INHIBITS the appearance of BSYNC L until BRPLY has been passive long enough for the REPLY F/F to have been reset. The reset state of the REPLY F/F must be ANDed with WDOUT H in order for the DOUT F/F to be set. This mechanism inhibits data from being placed on the bus until all slave device action has ended.

-CANCELS BDOUT L when the REPLY F/F has been set by the receipt of BRPLY L. BRPLY L indicates that the data output by the processor has been accepted by the addressed device.

-INHIBITS the setting of the DOUT F/F when WSYNC H is passive.

5.2.5 BRPLY L - REPLY H

BRPLY L is asserted by the addressed device when data is placed on the bus, in the case of an input operation, or data has been accepted from the bus, in the case of an output operation. Propagation delays along the system bus and response delays in the slave device cause BRPLY L to be asynchronous with the microinstruction cycle. This logic provides tne following 3 functions:

-DELAYS the appearance of the reply signal until the beginning of each new microcycle. This is accomplished by clocking the REPLY F/F with PH1 H.

-ENABLES the set state of the REPLY F/F to appear as REPLY L for input to the Control chip only when a valid I/O operation is in progress (BSYNC L is active).

-CANCELS the BDOUT L signal by negating the signal REPLY (1) which was ANDed with WDOUT H to set the DOUT F/F. This mechanism accelerates release of the system bus by the addressed device in the case of an output transaction.

5.2.6 BRPLY L - BUSY H

BRPLY L also affects the BUSY H input to the Control chip. BUSY H is asserted when the REPLY F/F is set and is not dependent on any enabling action. The BUSY H input is checked by the Control chip during PH3 of both Read and Write microinstructions. If BUSY H is asserted, execution is suspended and the pending bus transfer is delayed until the system bus becomes free. BUSY H is also asserted by the DMA arbitration circuitry when another master device has control of the LSI-11 bus.

5.2.7 WIAK H - BIACK H

WIAK H is asserted by the Control chip during execution of either Acknowledge microinstruction, RA or WA. This signal is asserted at the beginning of PH2 (simultaneously with WSYNC). Because BSYNC L does not take part in LSI-11 system bus interrupt transactions, the WSYNC H - BSYNC L interface logic inhibits BSYNC L. The interface logic associated with the Control chip WIAK H output provides the following 2 functions:

-DISABLES the INT ACK F/F (interrupt acknowledge) until after the appearance of WDIN H.

-DELAYS WIAK H (as BIACK H) until one microcycle after PH2 of the assertion of BDIN L.

These two functions assure that BDIN L arrives at the interrupting device before BIACK H. Note that the Write Acknowledge microinstruciton will never produce a BIACK H signal because WDIN H is not asserted as part of a write operation.

5.3 THE DATA-INPUT (DATI) OPERATION

5.3.1 DATI Operation, Minimum Execution Time

Figure 5-3 illustrates the control signal action relevant to a DATI operation which executes with minimum possible delay. The microinstruction sequence presented is Read, Input Word, Next Microinstruction:

```
R       Rb,Ra              ; ADDRESS DEVICE, SIGNAL DIN
IW      Rb,Ra              ; INPUT LOW AND HIGH BYTE
Next Microinstruction      ;
```

The individual events which occur in the DATI operation are discussed with the aid of Figure 5-3, which costains the microcycle reference numbers.

Microcycle 1 Read

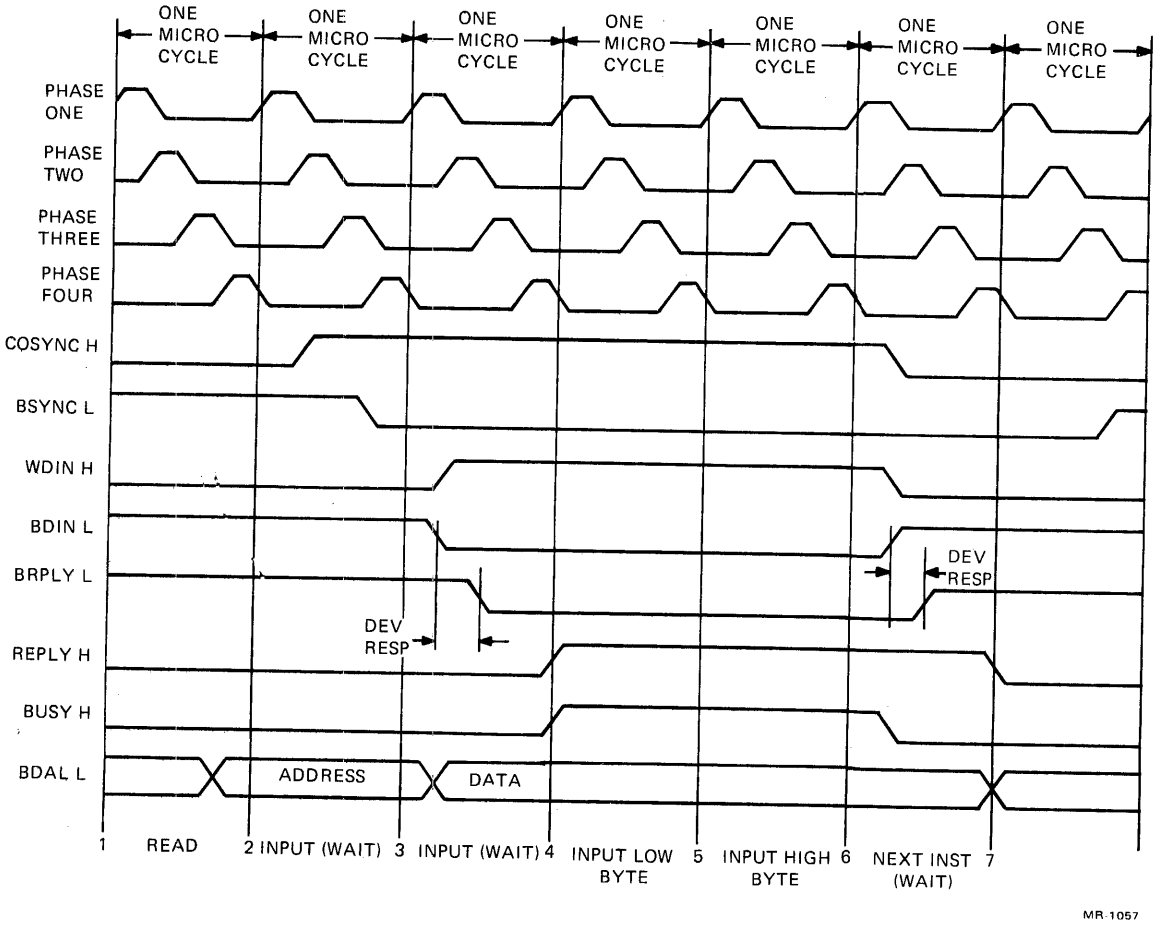
The execution of the Read microinstruction (R) in the first microcycle causes no change in the Control chip. The microinstruction executes to completion because REPLY H and BUSY H were not asserted during PH3.

Microcycle 2 Input Word (Wait)

Because the Read microinstruction executed to completion during Microcycle 1, the address information is placed on BDAL<15:00> at the beginning of PH1. WSYNC H is asserted at PH2, causing BYSNC L to be asserted at the beginning of PH4. Since the addressed device can only respond to BDIN L, which has not yet been asserted, the Input Word microinstruction check of REPLY H during PH3 fails and initiates the Wait state.

Minimum DATI Cycle

Figure 5-3



Microcycle 3 Input Word (Wait)

Because the Read microinstruction was completed during Microcycle 1, WDIN H is asserted at the beginning of PH2. The assertion of BDIN L follows immediately. In response to BDIN L, the addressed device places data on BDAL<15:00> and asserts BRPLY L to the processor. However, the REPLY F/F is not clocked until PH1 and REPLY H remains passive (which in turn maintains the Wait state).

To assure minimum delay in the execution of the DATI operation illustrated here, BRPLY L must be received by the processor in time to set the REPLY F/F at the beginning of Microcycle 4. Otherwise, the Wait state will continue throughout Microcycle 4 and the next opportunity to set the REPLY F/F will not occur until the beginning of Microcycle 5.

Microcycle 4 Input Word (Input Low Byte)

The asserted state of BRPLY L sets the REPLY F/F at the beginning of PH1. Since REPLY H is now active, the Data chip stores the low byte in its designated register.

Microcycle 5 Input Word (Input High Byte)

Since the PH3 check of REPLY H is still true, the Data chip stores the high byte in its designated register (determined by complementing the low-order bit of the A register field). No control signals are changed by either the processor or the addressed device.

Microcycle 6 Next Microinstruction (Possible Wait)

Because the Input Word microinstruction executed to completion during Microcycle 5, BDIN L goes passive at the end of PH 1. This causes the addressed device to (1) make BRPLY L go passive, and (2) remove its data from BDAL<15:00>. Since the SYNC F/F is locked into its Set state due to the set state of the REPLY F/F, BSYNC L remains asserted.

If the Next Microinstruction belongs either to the Read or Write group, the state of BUSY H is checked and a Wait state will be initiated. Any other microinstruction will execute. To assure minimum delay in the execution of the DATI operation illustrated here, BRPLY L must go passive at the processor in time to reset the REPLY F/F at the beginning of Microcycle 7. Otherwise the next opportunity to set the REPLY F/F will not occur until the beginning of Microcycle 8, and a possible Wait state initiated during Microcycle 6 will continue at least into Microcycle 8.

Microcycle 7 Next Microinstruction

Since BRPLY L was negated by the addressed device during Microcycle 6, the REPLY F/F is clocked to the reset state at the beginning of PH1, which cancels BUSY H. The reset state of the REPLY F/F also allows the SYNC F/F to be clocked to the reset state at the end of PH3, which makes BSYNC L passive.

5.3.2 DATI Operation, Delayed Execution Time

Figure 5-4 illustrates the control signal action for a DATI operation in which execution is delayed at 3 points. The delays occur during the Read microinstruction and in the response of the addressed device to both the assertion and negation of BDIN L. The microinstruction sequence presented here is Read, Input Word, Next Microinstruction:

```
R       Rb,Ra              ; ADDRESS DEVICE, SIGNAL DTN
IW      Rb,Ra              ; INPUT LOW AND HIGH BYTE
Next Microinstruction      ;
```

The individual events which occur in the DATI operation are discussed with the aid of Figure 5-4, which contains the Microcycle reference numbers.

Microcycle 1 Read (Wait)

The BUSY H input to the Control chip is asserted during the entire microcycle causing the PH3 test performed by the READ microinstruction to fail. The Wait state is initiated during PH3. Other control signal actions during microcycle 1 which relate to the conclusion of a previous I/O or DMA transaction have been omitted from the figures.

Microcycle 2 Read

Since the BUSY H signal goes passive prior to PH, the BUSY H test is passed, the Wait state is terminated and the Read microinstruction executes to completion.

Microcycle 3 Input Word (Wait)

Because the Read microinstruction completed during Microcycle 2, the address information is placed on BDAL<15:0> at the beginning of PH1. WSYNC H is asserted at PH2, causing BSYNC L to be asserted at the beginning of PH4. Since the addressed device can only respond to BDIN L, which has not yet been asserted, the Input Word microinstruction check of REPLY H during PH3 fails and initiates the Wait state.

Microcycle 4 Input Word (Wait)

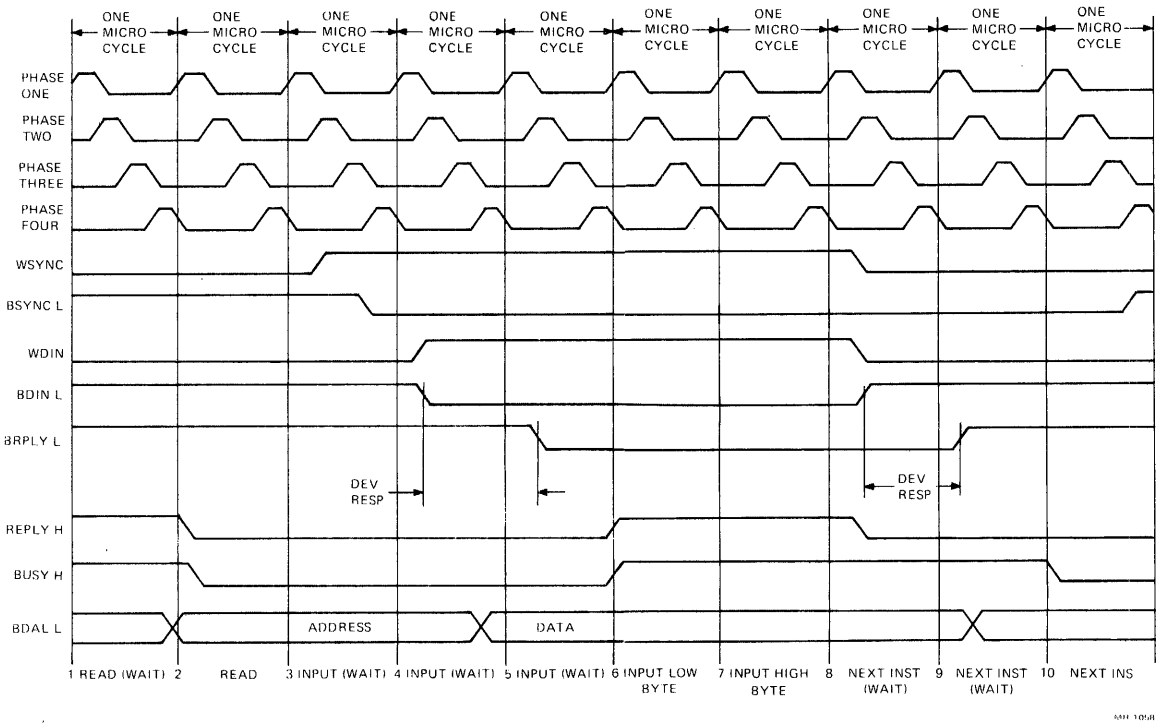
Because the Read microinstruction completed during Microcycle 2, WDIN H is asserted at the beginning of PH2. The assertion of BDIN L follows immediately. Since there is no change in the state of REPLY H, the Wait state is maintained.

Microcycle 5 Input Word (Wait)

In response to BDIN L asserted during Microcycle 4, the addressed device asserts BRPLY L and places data on BDAL<15:00>. Since there is no change in the state of REPLY H, the Wait state is maintained.

Delayed DATI Cycle

Figure 5-4



If BRPLY L had been asserted during Microcycle 4, REPLY H would have become active during Microcycle 5. As illustrated, the additional delay in device response adds one full microcycle to the execution of the DATI operation.

Microcycle 6 Input Word (Input Low Byte)

The asserted state of BRPLY L sets the REPLY F/F at the beginning of PH1. Since REPLY H becomes active, the Wait state is terminated during the PH3 REPLY H check and the Data chip stores the low byte in its designated register.

Microcycle 7 Input Word (Input High Byte)

Since the PH3 test of REPLY H again passes, the Data chip stores the high byte in its designated register (determined by complementing the low order bit of the A register field). No other control signals are changed by either the processor or the addressed device.

Microcycle 8 Next Microinstruction (Possible Wait)

Because the Input Word microinstruction completed during Microcycle 7, both WSYNC H and WDIN H are made passive at the end of PH1. Since WDIN H must be asserted to enable REPLY H during a DATI operation, REPLY H also goes passive. BDIN L goes passive immediately, which informs the addressed device to remove its data from BDAL<15:00>. Since the SYNC F/F is locked into its set state due to the set state of the REPLY F/F, BSYNC L remains asserted.

If the next microinstruction belongs either to the Read or Write microinstruction group, the state of REPLY H and BUSY H is tested and a new Wait state is initiated which delays the beginning of the next I/O operation. Any other microinstruction will execute.

Microcycle 9 Next Microinstruction (Possible Wait)

In response to BDIN L negated during Microcycle 8, the addressed device negates BRPLY L and removes its data from BDAL<15:00>. Since there is no change in the state of BUSY H the possible WAIT state would be maintained.

Microcycle 10 Next Microinstruction

Since BRPLY L was negated by the addressed device during Microcycle 9, the REPLY F/F is clocked to the reset state at the beginning of PH1, which cancels BUSY H. The reset state of the REPLY F/F also allows the SYNC F/F to be clocked to the reset state at the end of PH3, which makes BSYNC L passive.

Device Response Note

If BRPLY L had been negated during Microcycle 8, BUSY H would become passive during Microcycle 9. As illustrated, the additional delay in device response adds one full microcycle to the execution of the DATI operation.

5.3.3 DATI Microprogramming Summary

The DATI operation, implemented by the Read-Input microinstruction sequence, allows for variable delays in addressed device response to the assertion and negation of BDIN L. In any conventional machine configuration the device responses will likely not be the minimums discussed in section 5.3.1. To make better use of what would otherwise be Wait-induced microprocessor idle time, data manipulation microinstructions may be inserted between the Read and Input microinstructions. The time required to execute the inserted data manipulation microinstructions should optimatlly be close to, but less than, the worst case maximum device response to the assertion of BDIN L, so the LSI-11 bus will not be held busy for abnormal lengths of time. In this way the microprogram will execute with minimized or even zero overall delay. No difficulty is encountered if addressed device response causes REPLY H to be asserted before completion of the inserted data manipulation microinstruction sequence is completed.

Once the Input microinstruction has completed, the addressed device removes data and negates BRPLY L. The microproessor may be best used at this point by executing microinstrucitons which do not check the status of either REPLY H or BUSY H.

To summarize, the delay in execution of a micrprogram containing a DATI operation may be minimized if:

1. The microinstruction immediately preceding the Read microinstruction does not cause BUSY H or REPLY H to be asserted.

2. The addressed device responds to the assertion of BDIN L in minimum time.

3. The addressed device responds to the negation of BDIN L in minimum time.

4. The microinstruction immediately following the Input microinstruction does not check REPLY H or BUSY H.

As an alternative to condition 2 above, unavoidable delays may be utilized by inserting Data Manipulation microintructions between the Read and Input microinstructions.

5.4 THE DATA-OUTPUT (DATO) OPERATION

5.4.1 DATO Operation, Minimum Execution Time

Figure 5-5 illustrates the control signal action for a DATO operation which executes with minimum delay. The microinstruction sequence presented is Write, Output Word, Next Microinstruction:

```
W       Rb,Ra              ; ADDRESS DEVICE, SIGNAL DOUT
OW      Rb,Ra              ; OUTPUT WORD
Next Microinstruction      ;
```

The individual events which occur in the DATO operations are discussed with the aid of Figure 5-5, which contains the Microcycle reference numbers.

Microcycle 1 Write

The execution of the Write microinstruction in the first microcycle causes no change in the control signals. The microinstruction executes to completion because REPLY H and BUSY H are unasserted during PH3.

Microcycle 2 Output Word (Wait)

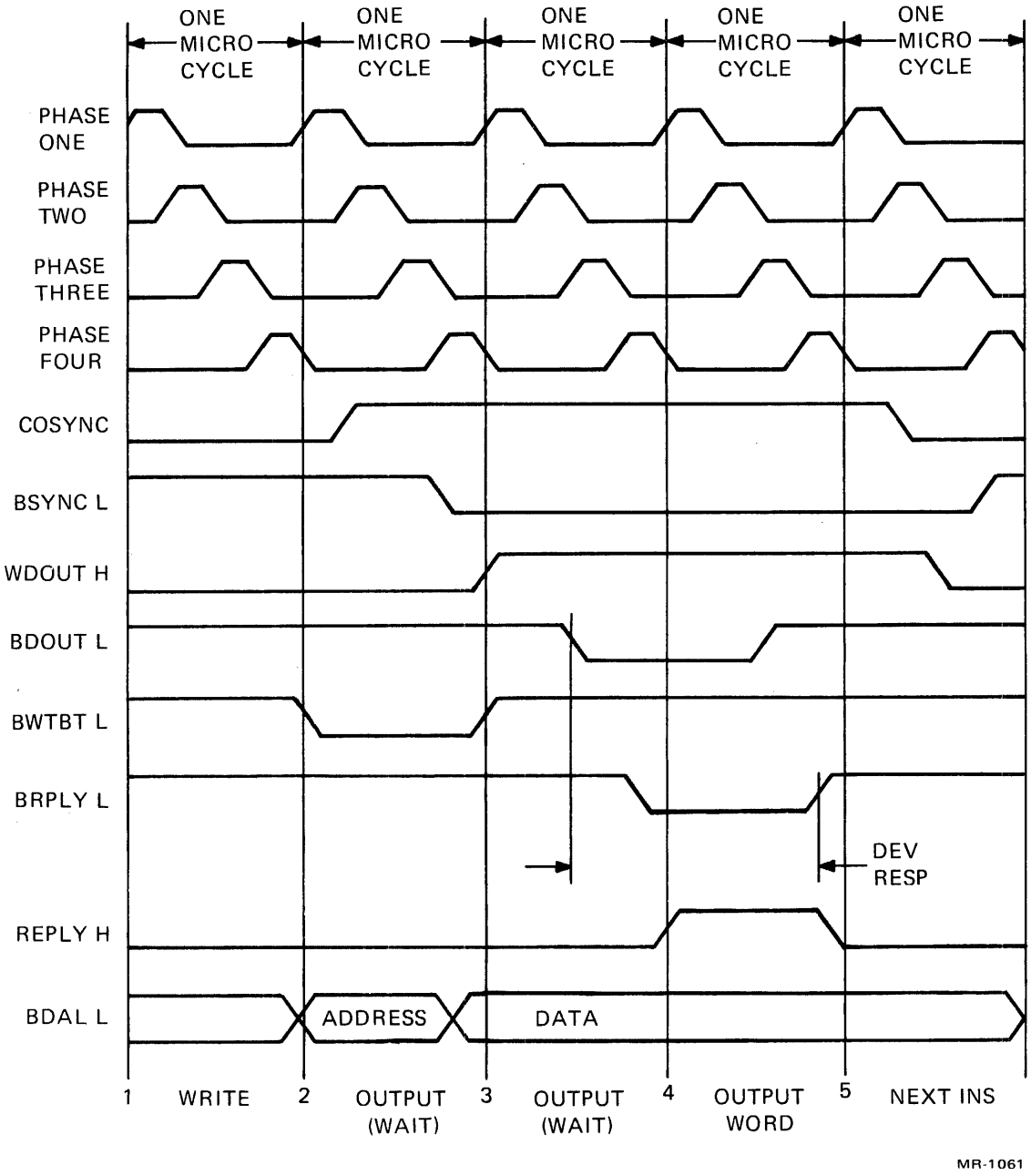
Because the Write microinstruction completed during Microcycle 1, the address information is placed on BDAL<15:00> at the beginning of PH1. The WWB H signal is asserted at the beginning of PH1, indicating a byte operation. The assertion of BWTBT L follows immediately. WSYNC H is asserted at PH2, causing BSYNC L to be asserted at the beginning of PH4. Since the addressed device can only respond to BDOUT L, which has not yet been asserted, the Output Word microinstruction check of REPLY H during PH3 fails and initiates the WAIT state.

Microcycle 3 Output Word (Wait)

Because the Output Word microinstruction executed up to the failing test of REPLY H during Microcycle 2, WDOUT H is asserted at the beginning of PH1. The REPLY F/F is in the reset state during this microcycle, which allows the asserted WDOUT H to set the DOUT F/F at the beginning of PH3. When the DOUT F/F is set, BDOUT L is asserted. Since this example contains the Output Word microinstruction, WWB H and subsequently BWTBT L are negated at the beginning of PH1. BWTBT L remains asserted past this point only in the case of an Output Byte operation. Also during PH1, the Data chip simultaneously places two bytes (a full word) of data on BDAL<15:00>. In response to the assertion of BDOUT L, the addressed device accepts the data output by the processor and returns BRPLY L. However, the REPLY F/F is only clocked at PH1, so REPLY H remains passive which maintains the Wait state.

Minimum DATO Cycle

Figure 5-5



To assure minimum delay in the execution of the DATO operation illustrated here, BRPLY L must be received by the processor in time to set the REPLY F/F at the beginning of Microcycle 4. Otherwise, the Wait state will continue throughout Microcycle 4 and the next opportunity to set the REPLY F/F will not occur until the beginning of Microcycle 5.

Microcycle 4 Output Word

The asserted state of BRPLY L sets the REPLY F/F at the beginning of PH1. Since REPLY H became active during PH3, the Wait state is terminated at the PH3 REPLY H test and the Output Word microinstruction executes to completion. The set state of the REPLY F/F also negates the DOUT F/F input, and it is clocked to the reset state at the beginning of PH3 which negates BDOUT L and cancels REPLY H. Note that the Control chip determines the state of REPLY H at the beginning of PH3, thus recognizing the addressed device response before REPLY H is canceled via the resetting of the DOUT F/F. In response to the negation of BDOUT L, the addressed device negates BRPLY L.

To assure minimum delay in the execution of the DATO operation illustrated, BRPLY L must go passive at the processor in time to reset the REPLY F/F at the beginning of Microcycle 5. Otherwise the next opportunity to reset the REPLY F/F will not occur until the beginning of Microcycle 6.

Microcycle 5 Next Instruction

Since BRPLY L was negated during Microcycle 4, the REPLY F/F is clocked to the reset state at the beginning of PH1, thus cancelling BUSY H. Because the Output Word microinstruction completed during Microcycle 4, WSYNC H is made passive at the beginning of PH2 and WDOUT H is made passive at the beginning of PH4. The Data chip removes the output word from BDAL<15:00> at the end of PH4. SYNC L is clocked to the passive state at the end of PH3.

Any type of microinstruction may be executed during Microcycle 5, since the REPLY F/F was reset at the beginning of PH1, thus negating BUSY H.

5.4.2 DATO Operation, Delayed Execution Time

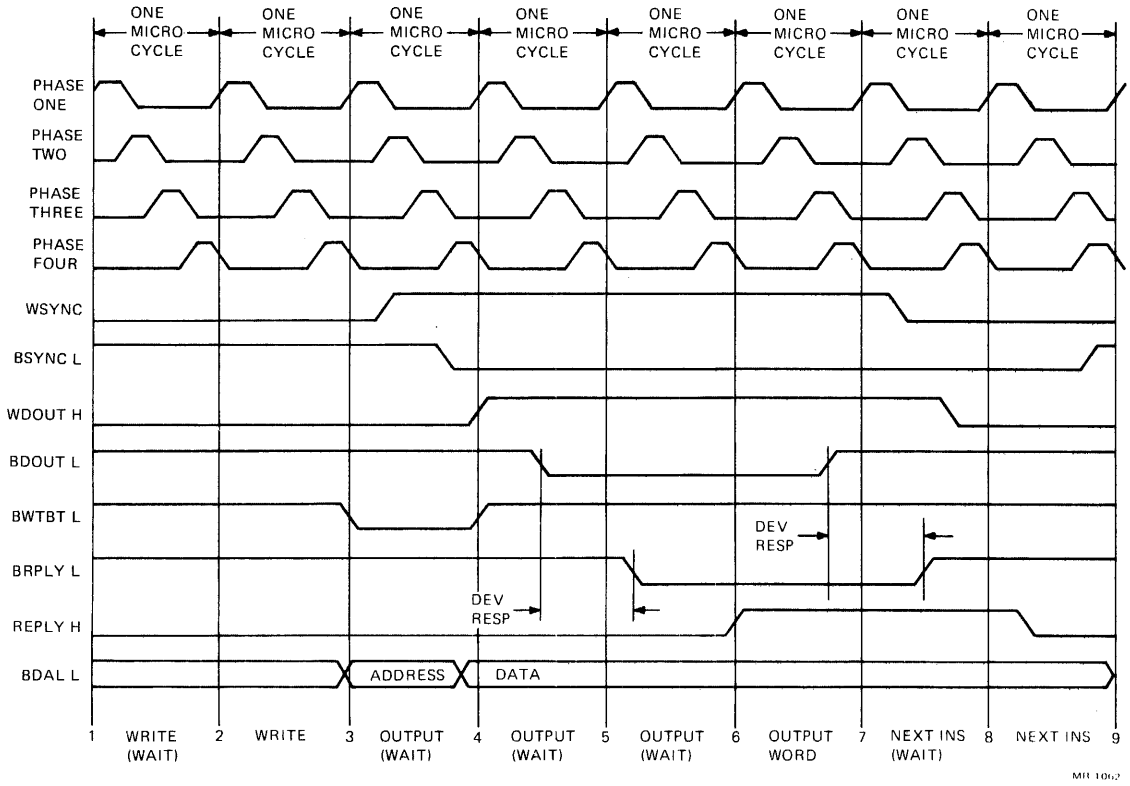
Figure 5-6 illustrates the control signal action for a DATO operation in which execution is delayed at 3 points. The delays occur during the execution of the Write microinstruciton and in the response of the addressed device to both the assertion and negation of BDOUT L. The microinstruction sequence presented here is Write, Output Word, Next Microinstruction:

```
W       Rb,Ra              ; ADDRESS DEVICE, SIGNAL DOUT
OW      Rb,Ra              ; OUTPUT WORD
Next Microinstruction      ;
```

The individual events which occur in the DATI operation are discussed with the aid of Figure 5-6 which contains the microcycle reference numbers.

Delayed DATO Cycle

Figure 5-6



Microcycle 1 Write (Wait)

The BUSY H input to the Control chip (not shown in the timing diagram) is asserted during the entire microcycle which causes the PH3 test performed by the Write microinstruction to fail. The Wait state is initiated during PH3. Other control signal actions during Microcycle 1 which relate to the conclusion of a previous I/O or DMA transaction have been omitted from the figures.

Microcycle 2 Write

Since the BUSY H signal goes passive prior to PH3, the BUSY H test is passed, the Wait state is terminated and the Write microinstruction executes to completion.

Microcycle 3 Output Word (Wait)

Because the Write microinstruction completed during Microcycle 2, the address information is placed on BDAL<15:00> at the beginning of PH1.

The WWB H signal is asserted at the beginning of PH1 which indicates a byte operation. The assertion of BWTBT L follows immediately.

WSYNC H is asserted at PH2, causing BSYNC L to be asserted at the beginning of PH4. Since the addressed device can only respond to BDOUT L, which has not yet been asserted, the Output Word microinstruction check of REPLY H during PH3 fails and initiates the Wait state.

Microcycle 4 Output Word (Wait)

Because the Output Word microinstruction executed until the failing test of REPLY H during microcycle 3, WDOUT H is asserted at the beginning of PH1. The REPLY F/F is in the reset state during this microcycle allowing the asserted WDOUT H to set the DOUT F/F at the beginning of PH3. When the DOUT F/F is set, BDOUT L is asserted. Since this example contains the Output Word microinstruction, WWB H and thus BWTBT L are negated at the beginning of PH1. BWTBT L remain asserted past this point only in the case of a Output Byte operation. Also during PH1, the Data chip simultaneously places two bytes (a full word) of data on BDAL<15:00>. Since REPLY H is unchanged in this microcycle, the Wait state is maintained.

Microcycle 5 Output Word (Wait)

In response to BDOUT L, the addressed device accepts the data on BDAL<15:00> and returns BRPLY L to the processor. However, the REPLY F/F is only clocked at PH1 and REPLY H remains passive which maintains the Wait state.

If BRPLY L had been asserted during Microcycle 4, REPLY H would have become active during Microcycle 5. As illustrated, the additional delay in device response adds one full microcycle to the execution of the DATO operation.

Microcycle 6 Output Word

The asserted state of BRPLY L sets the REPLY F/F at the beginning of PH1. Since REPLY H became active, the Wait state is terminated at the PH3 REPLY H check and the Output Word microinstruction executes to completion. The set state of the REPLY F/F also negates the DOUT F/F input, and it is clocked to the reset state at the beginning of PH3 which negates BDOUT L and cancels REPLY H. Note that the Control chip determines the state of REPLY H at the beginning of PH3, thus recognizing the response of the addressed device before REPLY H is cancelled via the resetting of the DOUT F/F.

Microcycle 7 Next Microinstruction (Possible Wait)

Because the Output Word microinstruction completed during Microcycle 6, WSYNC H is made passive at the beginning of PH2 and WDOUT H is made passive at the beginning of PH4. The Data chip removes the output word from BDAL<15:00> at the end of PH4. In response to the negation of BDOUT L, the addressed device negates BRPLY L. However, since the REPLY F/F is clocked at PH1, it remains in the set state which asserts BUSY H.

If the Next Microinstruction belongs either to the Read or Write group, the state of BUSY H is checked and a new Wait state will be initiated which delays the beginning of the next I/O operation. Any other microinstruction will execute immediately.

If BRPLY L had been negated during Microcycle 6, the REPLY F/F would have been reset at the beginning of Microcycle 7. As illustrated, the additional delay in device response adds one full microcycle to the execution of the DATO operation.

Microcycle 8 Next Microinstruction

Since BRPLY L was negated during Microcycle 7, the REPLY F/F is clocked to the reset state at the beginning of PH1, thus cancelling BUSY H. The reset state of the REPLY F/F also allows the SYNC F/F to be clocked to the reset state which negates BSYNC L.

If the microinstruction in Microcycle 7 caused the microprocessor to initiate the WAIT state, the microinstruction will now execute. Otherwise microinstruction execution proceeds normally.

5.4.3 DATO Microprogramming Summary

The DATO operation, as implemented by the Write-Output microinstruction sequence, allows for variable delays in addressed device response to both the assertion and negation of BDOUT L. However, there is an important difference between the DATI and DATO operations regarding delay handling. To optimize microprocessor performance in the DATO context, the Output microinstruction must immediately follow the Write microinstruction. This is necessary because BDOUT L is a function of the Output microinstruction. In contrast, BDIN L is a function of a Read microinstruction. Therefore, there is no opportunity (in the DATO context) to make use of idle time caused by the delay of the addressed device to respond to the assertion of BDOUT L.

Once BRPLY L is received, indicating that the addressed device has stored the output data, the LSI-11 system bus interface logic proceeds immediately to negate BDOUT L. This action accelerates the negation of BRPLY L by the addressed device. The microinstructions which follow the Output microinstruction should not be of the type which check REPLY H or BUSY H. Otherwise the resulting Wait state(s) will cause microprocessor idle time.

To summarize, the delay in the execution of a microprogram containing a DATO operation may be minimized if:

1. The microinstruction immediately preceeding the Write microinstruction does not cause BUSY H or REPLY H to be asserted.

2. The Write microinstruction is immediately followed by an Output microinstruction.

3. The addressed device responds to the assertion of BDOUT L in minimum time.

4. The addressed device responds to the negation of BDOUT L in minimum time.

5. The microinstruction immediately following the Output microinstruction does not check REPLY H or BUSY H.

5.5 THE DATA-INPUT-OUTPUT (DATIO) OPERATION

5.5.1 DATIO Operation, Minimum Execution Time

Figure 5-7 illustrates the control signal action relevant to a DATIO operation which executes with minimum delay. The microinstruction sequence presented is Read, Input Word, Modify the data, Output Word, Next Microinstruction:

```
R       Rb,Ra              ; ADDRESS DEVICE, SIGNAL DIN
IW      Rb,Ra              ; INPUT LOW AND HIGH BYTES
Modify the data            ; MODIFY DATA
OW      Rb,Ra              ; SIGNAL DOUT, OUTPUT WORD
Next Microinstruction      ;
```

The individual events which occur in the DATIO operation are discussed with the aid of Figure 5-7, which contains the Microcycle reference numbers.

Microcycle 1 Read

The execution of the Read microinstruction in the first microcycle causes no change in the control signals. The microinstruction executes to completion because REPLY H and BUSY H (not shown) are unasserted at PH3.

Microcycle 2 Input Word (Wait)

The Control chip determines that a Read-Modify-Write operation is to be performed by means of the argument contained in the B register field of the Input Word microinstruction. It stores this information internally and will not conclude the DATIO operation until an Output microinstruction is successfully completed.

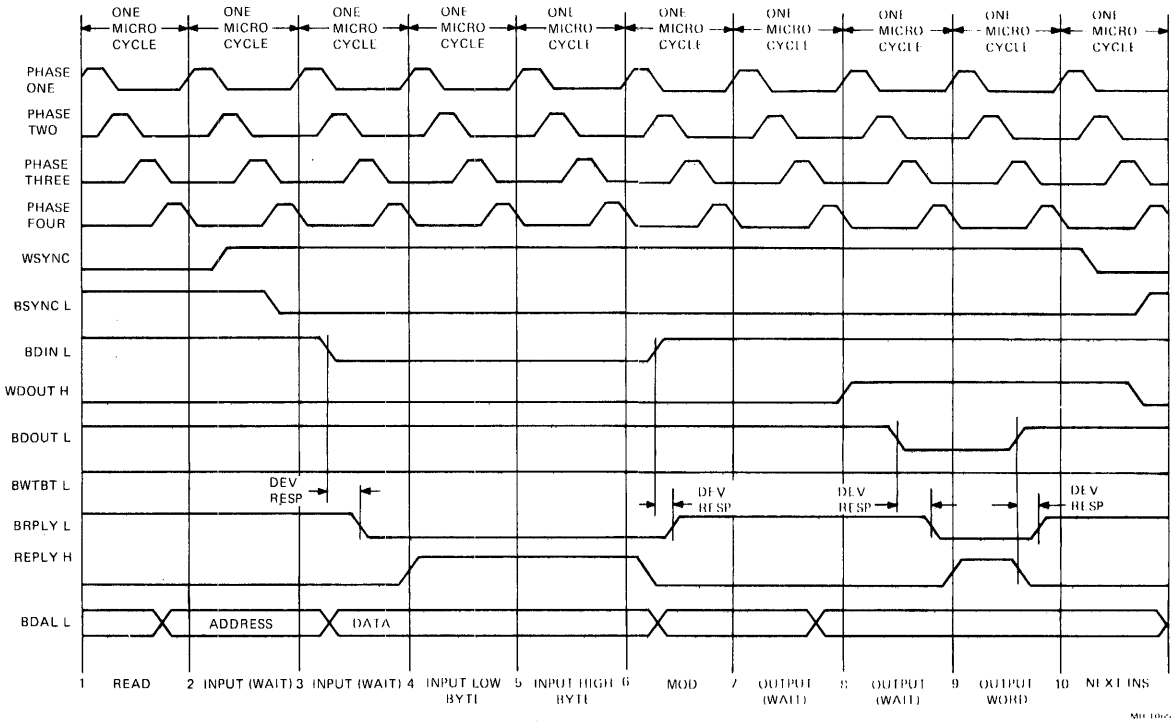
Because the Read microinstruction completed during Microcycle 1, the address information is placed on BDAL<15:00> at the beginning of PH1. WSYNCH is asserted at PH2, causing BSYNC L to be asserted at the beginning of PH4. Since the addressed device can only respond to BDIN L (during the input portion), which has not yet been asserted, the Input Word microinstruction test of REPLY H at PH3 fails and the Wait state is initiated.

Microcycle 3 Input Word (Wait)

Because the Read microinstruction completed during Microcycle 1, WDIN H (not shown) is asserted at the beginning of PH2. The assertion of BDIN L follows immediately. In response to BDIN L, the addressed device places data on BDAL <15:00> and returns BRPLY L to the processor. However, the REPLY F/F is not clocked until PH1 and REPLY H remains passive which maintains the Wait state.

Minimum DATIO Cycle

Figure 5-7



Microcycle 4 Input Word (Input Low Byte)

The asserted state of BRPLY sets the REPLY F/F at the beginning of PH1. Since REPLY H becomes active, the Wait state is terminated during the PH3 REPLY H check and the Data chip stores the low byte in a designated register.

Microcycle 5 Input Word (Input High Byte)

Since the PH3 check of REPLY H is again passed, the Data chip stores the high byte in a designated register (determined by complementing the low-order bit of the A register field). No control signals are changed by either processor or the addressed device.

To assure the minimum delay in the input portion of the DATIO operation, BRPLY L must be received at the processor in time to set the REPLY F/F at the beginning of Microcycle 4. Otherwise the Wait state will continue throughout Microcycle 4 and the next opportunity to set the REPLY F/F will not occur until the beginning of Microcycle 5.

Microcycle 6 Modify Data

Because the Input Word microinstruction completed during Microcycle 5, WDIN H (not shown) and BDIN L are negated at the end of PH1. The negation of WDIN H cancels REPLY H. The Read-Modify-Write operation maintains WSYNC H in the asserted state. The addressed device responds to the negation of BDIN L by removing data from BDAL<15:00> and negating BRPLY L.

This example of the DATIO operation allows one microcycle for modification of the data retrieved by the Input Word microinstruction. Normal usage would probably require manipulation of the entire data word which would necessitate at least 2 microcycles. Since none of the data manipulation microinstructions check the REPLY H or BUSY H signals, execution can proceed without delay.

Microcycle 7 Output Word (Wait)

Since BRPLY L was negated by the addressed device during Microcycle 6, the REPLY F/F is clocked into the reset state at the beginning of PH1, which cancels BUSY H (not shown). Execution of the Output Word microinstruction proceeds up to the PH3 check of REPLY H. REPLY H has been cancelled and the Wait state is initiated.

Microcycle 8 Output Word (Wait)

Because the Output Word microinstruction executed up to the failing check of REPLY H during Microcycle 7, WDOUT H is asserted at the beginning of PH1. BDOUT L is asserted at the beginning of PH3.

Since this example contains the Output Word microinstruction, WWBH and BWTBT L are not asserted. In the case of an Output Byte microinstruction, the assertion of BWTBT L would begin at PH1 of Microcycle 8 and end at PH1 of Microcycle 9. Also, as a result

of the partial execution of the Output Word microinstruction, the Data chip simultaneously places two bytes (a full word) of data on BDAL<15:00>.

In response to the assertion of BDOUT L, the addressed device accepts the data output by the processor, and asserts BRPLY L. However, the REPLY F/F is not clocked until PH1 and REPLY H remains passive which maintains the Wait state.

Microcycle 9 Output Word

The asserted state of BRPLY L sets the REPLY F/F at the beginning of PH1. Since REPLY H becomes active, the Wait state is terminated at the PH3 REPLY H check and the Output Word microinstruction executes to completion. The set state of the REPLY F/F also negates the DOUT F/F input and it is clocked to the reset state at the beginning of PH3, which negates BDOUT L and also cancels REPLY H. Note that the Control chip determines the state of REPLY H at the beginning of PH3, thus recognizing the addressed device response before REPLY H is cancelled via the resetting of the DOUT F/F.

Microcycle 10 Next Microinstruction

The negation of BRPLY L causes the REPLY F/F to be reset at the beginning of PH1 which in turn negates BUSY H (not shown). Because the Output Word microinstruction completed during Microcycle 9, WSYNC H is made passive at the beginning of PH2 and WDOUT H is made passive at the beginning of PH4. The Data chip removes the output word from BDAL<15:00> at the end of PH4. BSYNC L is clocked to the passive state at the end of PH3.

Any type of microinstruction may be executed during Microcycle 10, since the REPLY F/F was reset at the beginning of PH1, thus negating BUSY H.

5.5.2 DATIO Microprogramming Summary

The DATIO operation, as implemented by the Read-Input-Modify-Output microinstruction sequence, allows variable delays in addressed device response to assertion and negation of both BDIN L and BDOUT L. The optimizing considerations relevant here are a combination of those discussed in the DATI and DATO contexts.

To summarize, the delay in the execution of a microprogram containing a DATIO operation may be minimized if:

1. The microinstruction immediately preceding the Read microinstruction does not cause BUSY H or REPLY H to be asserted.

2. The addressed device responds to the assertion of BDIN L in minimum time.

3. The addressed device responds to the negation of BDINL in minimum time.

4. The addressed device responds to the assertion of BDOUT L in minimum time.

5. The addressed device responds to the negation of BDOUT L in minimum time.

6. The microinstruction immediately following the Output microinstruction does not check REPLY H or BUSY H.

As an alternative to condition 2 above, unavoidable delays may be utilized by inserting data manipulation microinstructions.

5.6 THE INTERRUPT OPERATION

Figure 5-8 illustrates the control signal action relevant to an interrupt operation. The microinstruction sequence presented is Read Acknowledge, Input Word, Next Microinstruction.

```
RA      Rb,Ra              ; SIGNAL DIN, IACK
IW      Rb,Ra              ; INPUT VECTOR LOW AND HIGH BYTE
Next Microinstruction      ;
```

The individual events which occur in the interrupt transaction are discussed with the aid of Figure 5-8, which contains the Microcycle reference numbers.

Microcycle 1 Read Acknowledge

The execution of the Read Acknowledge microinstruction in the first microcycle causes no change in the control signals. The microinstruction executes to completion because REPLY H and BUSY H are unasserted during PH3.

Microcycle 2 Input Word (Wait)

Because the Read Acknowledge microinstruction completed during Microcycle 1, the contents of the designated registers are placed on BDAL<15:00> at the beginning of PH1. Since no address information is required in the LSI-11 system bus interrupt transaction, any register(s) may be designated. WSYNC H and WIAK H are asserted at the beginning of PH2. The assertion of WIAK H holds the SYNC F/F in the reset state thus blocking the assertion of BSYNC L, as would otherwise occur at the beginning of PH4. BSYNC L is inhibited because it is not required in the LSI-11 system bus interrupt transaction.

The interrupting device only returns BRPLY L in response to BIACK H, which has not yet been asserted. Therefore, the Wait state is initiated at the beginning of PH3.

Microcycle 3 Input Word (Wait)

Because the Read Acknowledge microinstruction executed to completion during Microcycle 1, WDIN H is asserted at the beginning of PH2. The assertion of BDIN L follows immediately and is used to stablize the priorities in the interrupting device. Since REPLY H remains unasserted, the Wait state is maintained.

Microcycle 4 Input Word (Wait)

With the assertion of WDIN H during Microcycle 3, the INT ACK F/F is no longer locked in the reset state therefore the INT ACK F/F is clocked to the set state on the trailing edge of PH1 (and asserts BIACK H). In response to the assertion of BIACK H, the interrupting device places its device vector on BDAL<15:00> and asserts BRPLY L. However, the REPLY F/F is not clocked until PH1 of Microcycle 5. REPLY H remains passive which maintains the Wait state.

MICROPROGRAMMING LSI-11 SYSTEM BUS TRANSACTIONS

5.6.1 Interrupt Operation Microprogramming Summary

The interrupt operation, as implemented by the Read Acknowledge-Input microinstruction sequence, allows for variable delays in addressed device response to both the assertion and negation of BIACK H. BDIN L also takes part in the interrupt operation. The timing relationships between BIACK H and BDIN L is established by the LSI-11 system bus interface logic. The return of BRPLY L and the device vector is under control of BIACK H, not BDIN L, as is normally the case. Because WDIN H must be asserted before the INT ACK F/F can be set, the Write Acknowledge microinstruction does not assert BIACK H on the LSI-11 Bus.

To summarize, the delay in the execution of a microprogram containing an interrupt operation may be minimized if:

1. The microinstruction immediately preceding the Read Acknowledge microinstruction does not cause REPLY H or BUSY H to be asserted.

2. The interrupting device responds to the assertion of BIACK H in minimum time.

3. The interrupting device responds to the negation of BIACK H in minimum time.

4. The microprogram immediately following the Input microinstruction does not check BUSY H or REPLY H.

As an alternative to 2 above, unavoidable delays may be utilized by inserting data manipulation microinstructions. However, due to the largely unpredictable nature of the interrupt operation, it is not expected that the freedom to perform data manipulation between execution of the Read Acknowledge and Input microinstructions would be of any general use.

CHAPTER 6

THE LSI-11 WRITABLE CONTROL STORE

6.1 GENERAL

The LSI-11 Writable Control Store option consists of a single quad height (8.5x10 inch) printed circuit module (M8018), a WCS cable/plug assembly and an LSI-11 CPU.

The Writable Control Store module is a 1024 x 24 bit microcode RAM for the LSI-11 CPU which enables user specified machine instructions to be added to the standard LSI-11 (PDP-11) instruction set. The module also contains Trace RAM logic to facilitate microprogram development and 2 additional TTL Control bits in each microcode word (not found in an LSI-11 MICROM) that are available at the backplane for high speed control applications. The microaddress range that the WCS responds to is determined by setting an 8 wide DIP switch on the module.

The WCS cable/plug assembly connects the WCS module to the microinstruction bus on the LSI-11 CPU (see Figure 6-1).

The WCS option can be utilized only with M7264-YC LSI-11 CPU.

6.2 THE WRITABLE CONTROL STORE MEMORY

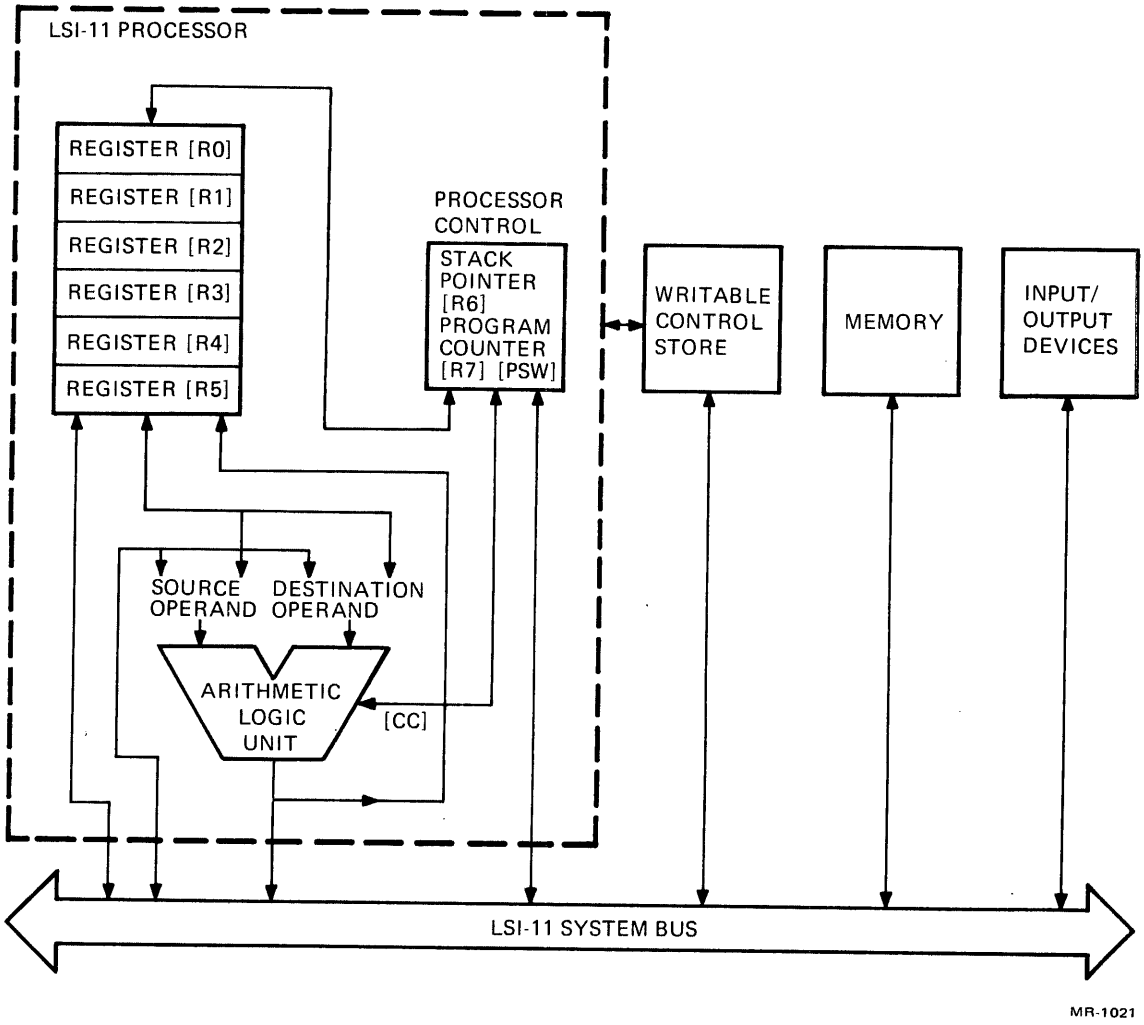
The memory of the Writable Control Store module (1024 24-bit memory words) is implemented by 24 1024 x 1 bit high speed static semiconductor memory devices. Since the memory is static, no refreshing is required. However, the semiconductor memory is volatile, so the control store must be reloaded after each power up.

6.2.1 Control Store Microword Organization

The microword organization is determined by the function of individual bit fields and by the access timing. As shown in Figure 6-2, the 24 bit WCS microword is composed of the standard 22-bit microinstruction, (MI<21:0>) and 2 additional bits (MI<23:22>) which function as extended TTL control bits.

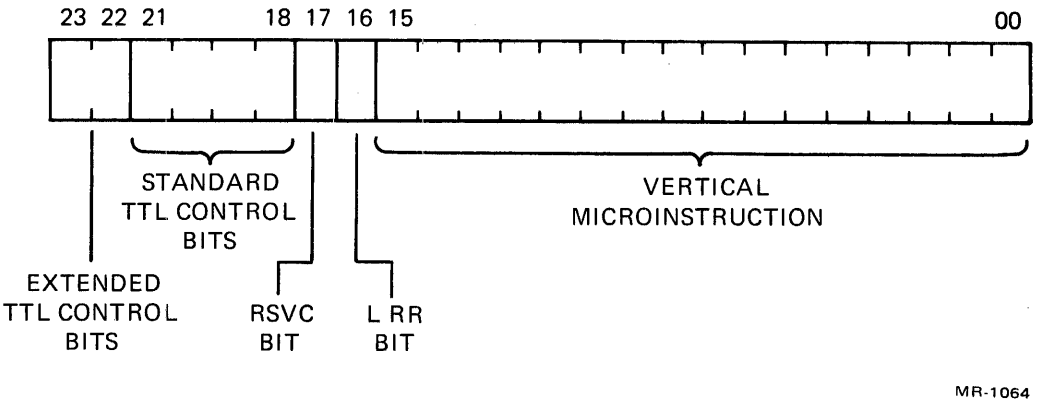
LSI-11 Processor-Writable Control Store Interconnection

Figure 6-1



Standard 22-Bit Microword Plus Extended Control Bits

Figure 6-2



6.2.1.1 Standard 22-Bit Microword MI<21:0> - This microword is composed of 4 functionally distinct and independent fields, as illustrated in Figure 6-2. The three lower fields, MI<17>, MI<16>, MI<15:0> are read by the microprocessor Control and Data chips during the microfetch operation at microcycle PH1. The upper field, MI<21:18> is accessed by the Special Control Function Logic on the CPU module at microcycle PH3. These 4 bits are then decoded to provide TTL compatible control signals synchronized with the vertical microinstruction.

For the standard 22-bit microword, the Writable Control Store memory is functionally identical to a MICROM with the following exceptions:

1. It does not precharge the MIB lines. Note that all MICROMs on the CPU unconditionally perform precharge.

2. MI<21:18> are asserted by the WCS during PH2 and PH3 as compared to a MICROM which asserts MI<21:18> during PH1, PH2 and PH3.

6.2.1.2 Extended TTL Control Bits MI<23:22> - The WCS memory provides storage for two additional control bits not found in an LSI-11 MICROM, called the Extended TTL Control bits. Both bits (MI<23:22>) appear at the WCS module fingers (AE1 and AF1) for user access via backplane connection. The highest bit, MI<23>, is also used by the microaddress trace RAM on the WCS module. These Extended TTL Control bits are synchronized with micromachine operations.

6.2.2 Control Store Microaddressing Modes

The WCS module contains an 8 wide DIP switch which determines the relationship of the WCS memory to the LSI-11 microaddress space. There are four address modes which are of general use and the switch positions for each mode are illustrated below:

	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
MODE 1	ON	OFF	OFF	ON	OFF	ON	OFF	-
MODE 2	OFF	ON	OFF	ON	OFF	OFF	ON	-
MODE 3	OFF	OFF	ON	ON	OFF	ON	OFF	-
MODE 4	ON	OFF	OFF	OFF	ON	ON	OFF	-

The address modes are explained as follows:

MODE 1 The WCS memory responds to microaddresses 2000-3777 (octal). Note that MICROMs 0 and 1 contain the PDP-11 and console ODT microcode and respond to microaddresses 0000-1777 (octal).

MODE 2 This is the paging mode where the 1024 WCS memory locations are treated as two 512 microword pages. Either page can respond to control store addresses in the 3000-3777 (octal) range. The WCS page from which a microword is accessed is determined by the WCS page logic. After this logic is initialized (by asserting the reset bit in the control/status register, CSR<15>), all microaddresses point to WCS page 0

(the initial page). The WCS pages are swapped immediately after a microinstruction containing MI<21:18> = 07 (octal) is accessed in control store and subsequent control store microwords are accessed from page 1 (the toggled page). A second occurance of 07 in MI<21:18> causes microwords to be read from page 0 again.

MODE 3 The WCS memory responds to microaddresses 2000-3777 (octal). This is similar to MODE 1 except that WCS memory pages are physically swapped on the module. MODE 3 provides diagnostic capability to execute a microprogram out of different physical memory pages. Note that loading and accessing WCS memory is identical to MODE 1.

MODE 4 The WCS memory responds to microaddresses 0000-1777 (octal). Normally, this mode is not used because LSI-11 MICROMs 0 and 1 respond to the same microaddresses.

In the modes described above, the WCS memory is loaded from the LSI-11 system bus via WCS RAM addresses 0000-1777 (octal). Figure 6-3 illustrates the relationship between the WCS RAM addresses and the responding microaddresses for each of the four modes. Note that page 0 is always accessed via the system bus in the WCS RAM address range 0000-0777 (octal) and similarly that page 1 is always accessed in the WCS RAM address range 1000-1777 (octal).

6.3 WRITABLE CONTROL STORE MEMORY ACCESS

The memory on the WCS module may be accessed in two ways: (1) by the microprocessor chip set via the microinstruction bus (Read only) and (2) by the LSI-11 processor via the LSI-11 system bus (Read/Write) using normal PDP-11 instructions.

6.3.1 Microinstruction Bus Access

Figure 6-4 illustrates the LSI-11 micromachine configuration which contains both Writable Control Store and MICROMs for microprogram storage. The first half of the illustration, Figure 6-4-1, appeared earilier in Chapter 3. The second half of the illustration, Figure 6-4-2, shows the interconnections to the WCS module. In addition to bus interconnections at both the machine and micromachine level, the WCS module also connects to 2 of the processor module clock phases, PH2 H and PH4 H. These two connections are contained within the Microinstruction Bus interconnect cable.

Note that a Writable Control Store memory access by the microinstruction bus is Read Only. There are no microinstructions which alter control store memory via the microinstruction bus.

WCS RAM Address-Microaddress Relationship

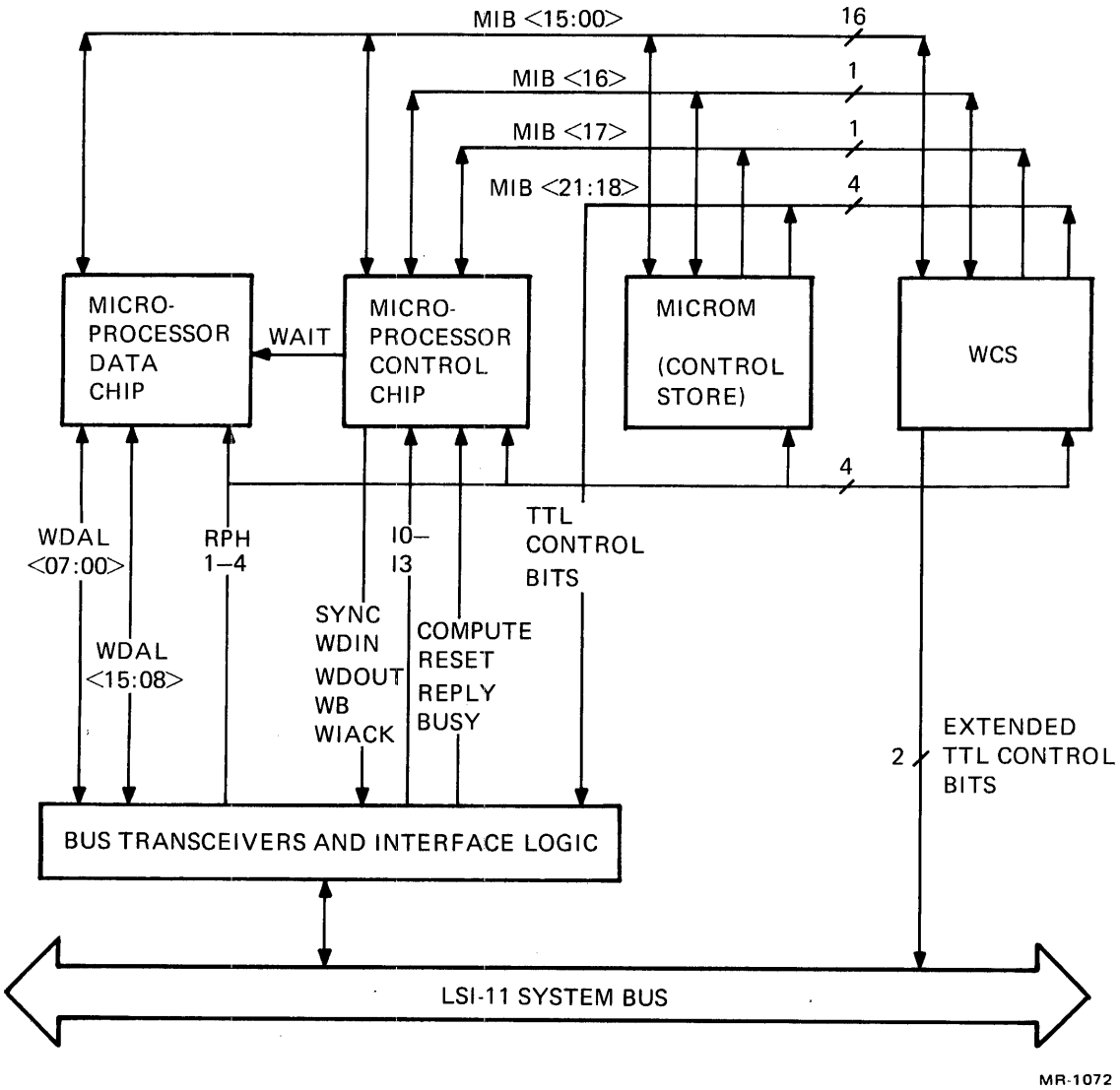
Figure 6-3

	WCS RAM <0000–0777>	WCS RAM <1000–1777>
MODE1	MIB<2000–2777>	MIB<3000–3777>
MODE2	MIB<3000–3777> (INITIAL PAGE)	MIB<3000–3777> (TOGGLED PAGE)
MODE3	MIB<2000–2777>	MIB<3000–3777>
MODE4	MIB<0000–0777>	MIB<1000–1777>

MR-1071

WCS Microinstruction Bus and System Bus Interconnection

Figure 6-4



The 24 signal paths in the microinstruction bus interconnect cable are listed in Figure 6-5. MIB<10:00> carries the microaddress, MADR<10:00>, to the WCS during the control store addressing phase of the microcycle (PH2). The WCS then responds, during the control store access phase, by asserting the microinstruction word on MIB<21:0>. Note that the MIB<10:0> lines are time-multiplexed for address and data information. During the first microcycle of a multiple cycle microinstruction, a control store disable function (CSD) is performed with MIB<16>. If MIB<16> is asserted by the Control chip during PH3, WCS (or MICROM) response during the next microcycle is disabled. The highest 2 bits of the WCS microword, the Extended TTL Control Bits, do not appear in Figure 6-5 since they are not utilized by the LSI-11 CPU module and consequently are not carried by the MIB interconnection cable.

6.3.1.1 Control Store Access Timing - The control store access timing for the WCS module is shown in Figure 6-6. As noted in the figure, the WCS module does not perform precharge of any Microinstruction Bus lines. This function is performed by every MICROM connected to the Microinstruction Bus. It is important also to note that TTL control bits, MI<21:18>, are asserted (active low) from the beginning of PH2 through the end of PH3 whereas the lower bits of the control store, MI<17:00> remain asserted during PH1 only.

During the first microcycle of a multiple microcycle operation, MIB<16> is asserted by the Control chip to disable the control store response during the following microcycle. This is shown in Figure 6-7.

6.3.1.2 Extended TTL Control Bit Timing - The timing diagrams illustrated in Figure 6-6 and 6.7 also show the timing for the Extended TTL Control bits MI<23:22>. These bits are latched on the WCS module and are available at the backplane (pins AE1 and AF1) as described below:

1. If a single cycle microinstruction is executed, the Extended TTL Control bits are valid from rising PH1 of the microinstruction to the next rising PH1. Note that multiple single cycle microinstructions with the same Extended TTL Control bit(s) asserted will produce a continuously asserted control signal.

2. If a multiple cycle (2 or more cycles) microinstruction is executed, the Extended TTL Control Bits are valid from rising PH1 of the first microcycle to rising PH3 of that cycle and are cleared to "0" for any subsequent microcycles of that microinstruction.

Microinstruction Bus Access Functions

Figure 6-5

SIGNAL	ADDRESS CYCLE	ACCESS CYCLE
MIB<01>	MADR <01>	MI<01>
MIB<02>	MADR<02>	MI<02>
MIB<03>	MADR<03>	MI<03>
MIB<04>	MADR<04>	MI<04>
MIB<05>	MADR<05>	MI<05>
MIB<06>	MADR<06>	MI<06>
MIB<07>	MADR<07>	MI<07>
MIB<08>	MADR<08>	MI<08>
MIB<09>	MADR<09>	MI<09>
MIB<10>	MADR<10>	MI<10>
MIB<11>		MI<11>
MIB<12>		MI<12>
MIB<13>		MI<13>
MIB<14>		MI<14>
MIB<15>		MI<15>
MIB<16>	CSD <16>	MI<16>
MIB<17>		MI<17>
MIB<18>		MI<18>
MIB<19>		MI<19>
MIB<20>		MI<20>
MIB<21>		MI<21>
PH2 H		WCS TIMING
PH4 H		WCS TIMING

MR-1073

THE LSI-11 WRITABLE CONTROL STORE

6.3.2 LSI-11 System Bus Access

The Writable Control Store RAM memory is Read/Write only when accessed by the LSI-11 system bus. The WCS interface logic supports only Programmed I/O data transfers via the system bus interface registers.

6.4 THE MICROADDRESS TRACE RAM

The Writable Control Store option has a 16-word microaddress trace RAM to aid with microprogram debugging. The trace hardware is normally controlled by the operator under MODT (see Chapter 7).

6.4.1 Microaddress Trace RAM Operation

The hardware portion of the microaddress trace RAM consists of a 16-word RAM which is continuously loaded with the last microaddress asserted on the Microinstruction Bus by the microprocessor Control chip. The RAM, therefore, contains the last 16 microaddresses presented to the WCS (or MICROM) including the disabled cycle(s) of a multiple cycle microinstruction. To stop a microaddress trace for examination, the operator (via MODT) sets MI<23> (one of the extended TTL bits) of the microword that is to be the last microinstruction in the Trace RAM. After this microinstruction is fetched by the microprocessor Control chip, the WCS hardware inhibits the microaddress trace RAM clock from any further operation. Once the clock is inhibited, the trace RAM contents remain unchanged. The operator may then access the buffer to read the traced microaddresses in the order of the actual execution.

The Trace RAM is implemented with three 16x4-bit random access memories addressed by a 4-bit counter. The counter is normally clocked continuously, thus providing a recirculating, increasing address sequence (modulo 16) to the RAM memories. When the microprogram trace has been halted, each of the stored microaddresses can be accessed.

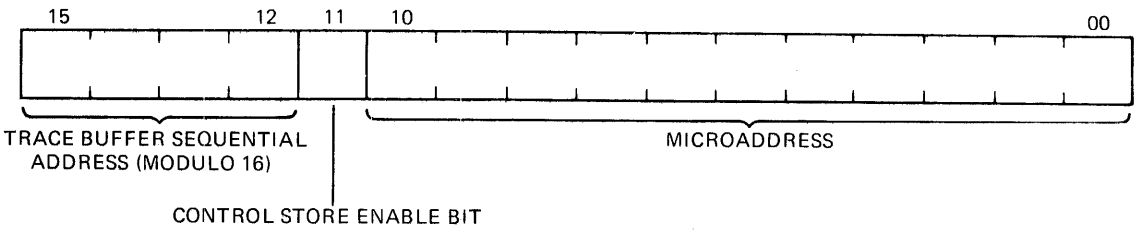
The format of a Trace RAM word is shown in Figure 6-8.

6.4.2 WCS Enable Bit

In addition to providing a record of the last 16 microaddresses on the MIB, the Trace RAM also indicates whether the WCS responded to the microaddress. The RAM stores an extra bit, the WCS Enable Bit (bit <11>), along with each microaddress. This bit is a "0" when the WCS was disabled during the microfetch and set to "1" when the WCS was enabled. This is useful in determining microprogram execution delays incurred during data access (system bus I/O) operations.

Microaddress Trace RAM Word Format

Figure 6-8



6.5 LSI-11 SYSTEM BUS INTERFACE

The WCS interface to the LSI-11 bus is similar to that of other I/O devices which support only Programmed I/O transactions. The device address format is a modification of the general format presented earlier in Chapter 2. The primary features of the WCS interface registers are that there is a single control/status register and that the two other registers provide the 24-bit access path for the WCS microcode RAM. In addition, one register is multiplexed to provide access to the microaddress Trace RAM. WCS interface register formats are illustrated in Figure 6-9. The register addresses on the LSI-11 system bus are 177540 through 177545.

6.5.1 WCS Control/Status Register

The specific functions of the WCS control/status register are explained below according to the related bit-fields:

CSR<15> Read/Write

This bit functions as the WCS Reset. When set to a "1", the control store memory paging logic is reset to page 0, and the Trace RAM address is set to a "0". Subsequently clearing CSR<15> to a "0" enables the paging logic and the counter that supplies the Trace RAM address.

CSR<14> Read/Write

This bit functions as the Trace RAM Examine bit. It should be set to "0" for normal control store operation. When it is set to a "1", the interface register located at 177542 may be read to access a Trace RAM word. The Trace RAM word is Read only .

CSR<13> Read/Write

This bit functions as the Examine Toggle for the Trace RAM address counter and is toggled to examine sequential trace words. Its use is further explained later in this chapter.

CSR<12> Read/Write

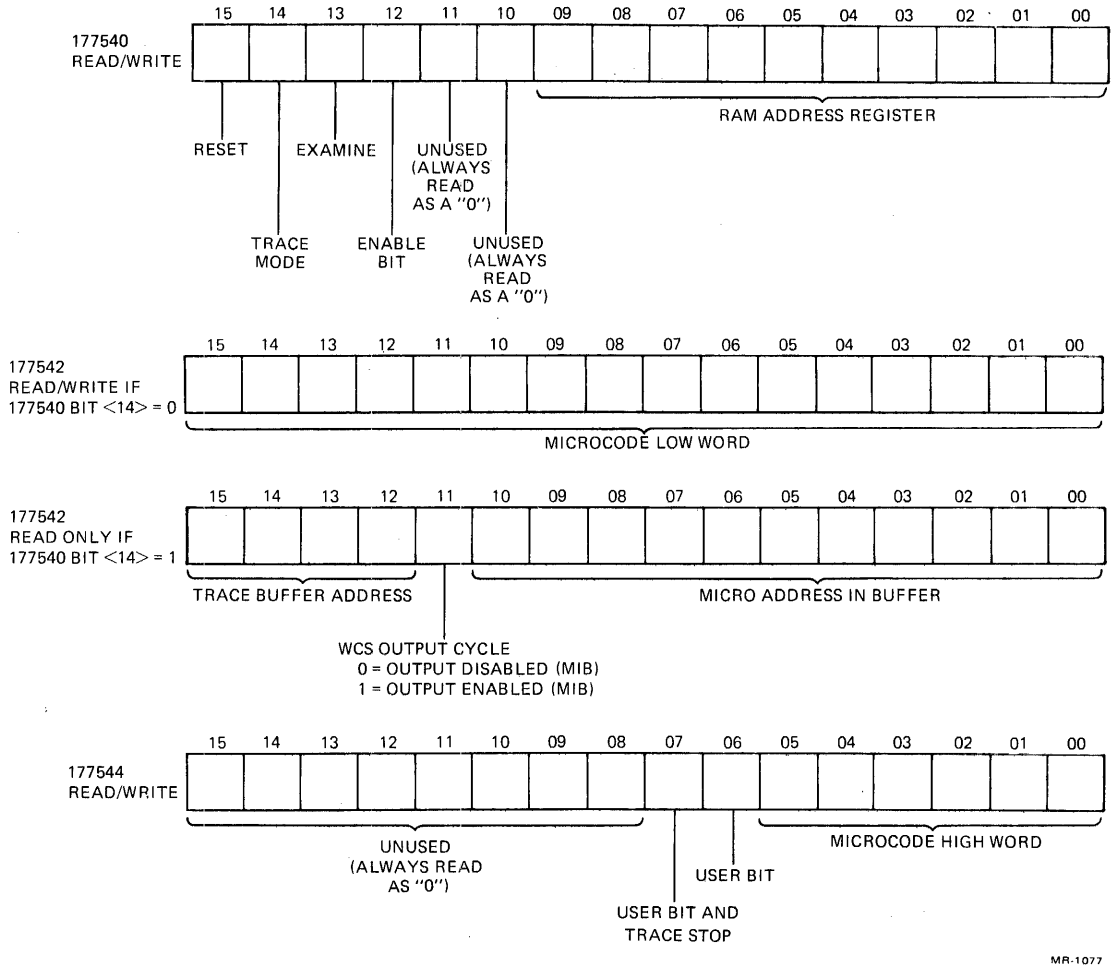
This bit functions as the Writable Control Store Enable bit. When it is set to a "0" the WCS module cannot respond to the microinstruction bus. However, the WCS RAM can then be accessed by the LSI-11 bus. When CSR bit <12> is set to a "1", the WCS module responds to microaddresses in its switch-selected range; and the WCS RAM cannot be altered by the LSI-11 bus.

CSR<11:10> Read Only

These bits are unused and always read as "0".

WCS Interface Register Bit Assignments

Figure 6-9



CSR<9:0> Read/Write

When CSR<12> is a "0", the WCS module is not enabled. This field is then the 10-bit WCS RAM address and is Read/Write. CSR <0> is the low-order bit and CSR <9> is the high order bit.

When CSR<12> is a "1", the WCS module is enabled. This field is then Read Only and contains the microaddress of the actual bus cycle of the PDP-11 instruction (or Console ODT routine) that is used to access the control/status register.

6.5.2 WCS Memory Access Registers

When CSR<12> is a "0", the WCS module is not enabled to respond to the MIB. WCS RAM access by the LSI-11 bus is enabled. The WCS module device registers, which also function as WCS RAM data access registers, are 177542 and 177544, as shown in Figure 6-9. The lower 16 bits of the microword, MI<15:0>, are accessed via location 177542. The higher 8 bits, MI<23:16> are accessed via location 177544. Note that bits <15:8> of location 177544 are unused and are read as "0". The address of the WCS RAM microword accessed via these registers is determined by CSR<9:0> (only when CSR<12> is a "0").

Both WCS memory access registers support Read/Write access only when the WCS is disabled (CSR<12>=0). The Write mode, implemented by an LSI-11 DATO operation, allows WCS memory to be loaded. The Read mode, implemented by an LSI-11 DATI operation, allows the WCS memory contents to be examined.

When the WCS module is enabled (CSR<12>=1), both WCS registers (177542 and 177544) are Read Only, and the data which is read will depend on the type of PDP-11 instruction (or console ODT) used to access the register. This is due to the fact that, when enabled, the WCS module is being addressed only by the MIB, and the microaddress present on the MIB determines the WCS RAM location that is accessed.

6.5.3 Microaddress Trace Register

When bit <14> of the control/status register is set to 1, location 177542 no longer functions as a memory access register, but as a Trace RAM access register, as shown in Figure 6-9. The specifications of this register are explained according to the related bit fields:

Bits <10:0> This field contains an 11-bit microaddress (Bit 0 is the low-order microaddress bit). The value read from Bits <10:0> is the value that appeared on the microinstruction bus (at microaccess time) while the microprogram was being traced.

Bit <11> This bit is the value of the WCS Enable Bit which is stored in the trace buffer along with each microaddress. A value of 0 indicates that WCS response was disabled for the microcycle.

Bits <12:15> This bit field contains the 4-bit (modulo 16) address of the trace buffer memory. The counter value is made

available in the trace word to aid in dumping the buffer contents in the proper order of occurrence.

6.5.3.1 Microaddress Trace RAM Dump Algorithm - Proper operation of the microaddress trace RAM hardware requires the following steps:

1. The trace RAM is initialized (by asserting CSR<15>) prior to execution of microcode.

2. The last microinstruction to be traced contains a value of 1 in MI<23>.

After the trace RAM is initialized, the trace RAM is continually updated to contain the 16 most recent microaddresses placed on the microinstruction bus by the microprocessor Control chip (at PH2). The RAM contents are frozen when the microinstruction containing MI<23>=1 is accessed including the address of that microinstruction. The RAM contents may be dumped by setting CSR<14> to a "1" and then subsequently accessing the trace RAM contents by toggling CSR<13>. The algorithm used is illustrated in Figure 6-10. Note that the RAM memory address (Trace bits <15:12>) must be saved before the Toggle store loop is entered to provide a reference for any subsequent tests. Also note that every CSR Write operation must contain a "1" in bit <14> to maintain the Trace Examine Enable and count down the Trace RAM address counter. The final 2 events in the flow chart initialize the Page Logic and the Trace RAM and re-enable the WCS.

6.6 WRITABLE CONTROL STORE MODULE DESCRIPTION

A Block Diagram of the WCS module is illustrated in Figure 6-11 and should be referred to for the following Circuit Description.

6.6.1 Clock Generation

The clock generation circuit receives and buffers 2 of the 4 TTL microcycle clock phases (PH2 H and PH4 H) from the LSI-11 processor module. These clock signals are connected to the WCS module via the microinstruction bus interconnect cable. Note that only LSI-11 CPU modules of etch revision F (CS Rev Y) or later have the necessary clock signals available at the empty MICROM socket, E75 (alternately used for the KEV-11 option). The M7264-YC module satisfies these requirements.

The clock generation circuit then derives 2 clock signals from its inputs, namely, PH1 H and PH23 H. PH23 H is asserted high continuously during microcycle phases 2 and 3 and is used to enable the standard TTL control bits to the LSI-11 CPU. PH1 H enables the output of MI<17:0> to the CPU.

6.6.2 Control Store Memory and Microaddress Multiplexer

The control store memory is implemented with 24 RAM integrated circuits each having a 1-bit by 1024 organization. The control store page organization is established by the microaddress multiplexer in conjunction with the paging logic.

6.6.3 LSI-11 System Bus Interface

The system bus interface is implemented with integrated bus transceivers (DC005) and a protocol logic circuit (DC004). The register selected to be read (of the 4 possible registers) is determined by the read back multiplexer. The Extended TTL Control Bits are available on the backplane at pin locations which are normally spare (AE1 and AF1). Note that the WCS module does not respond to bus address 177546 (since this is the address assigned to the BDV11 and KPV11 options).

6.6.4 Microinstruction Bus Interface

The microinstruction bus interface is implemented with special integrated circuits which interface the MOS logic levels of the Microinstruction Bus to the TTL levels on the WCS module. Only 12 receivers are implemented (MIB<10:0> and MIB<16>), and since 22 bits of the stored microword are returned via the microinstruction bus, 22 MIB drivers are implemented. The timing for the MIB driver is determined by the output enable circuitry. Control store output is enabled only when:

1. An appropriate microaddress has been received,
2. The WCS module is enabled (CSR<12>=1), and
3. When the control store disable bit (MIB<16>) was not asserted by the Control chip on the previous cycle (PH3).

6.6.5 Microaddress Trace RAM

The microaddress trace facility is implemented by three 16x4 bit RAM memories, one 4-bit counter (which addresses the memories), and associated clocking logic. Toggling CSR <15> to a "1" and then back to a "0" initializes the counter and restores the clock input (PH2) to the counter. The 16 microaddress RAM locations produced by this configuration are loaded from the output of the microaddress multiplexer, MADR<10:0>. Bit <11> is loaded with a "1" when the WCS is enabled to respond to the MIB or a "0" otherwise. In this way the stored Trace RAM data reflects the control store address placed on the MIB by the microprocessor Control chip and whether the WCS responded. The RAM buffer contents are examined from the system bus interface via the read back multiplexer.

6.7 WRITABLE CONTROL STORE HARDWARE SPECIFICATIONS

The items contained in this section provide a quick reference for module hardware details.

6.7.1 Dimensions

The KUV11-AA LSI-11 Writable Control Store option consists of (1) a standard quad height, 8.5 by 10 inch, multilayer printed circuit board (M8018) with signal etch on both sides and 2 inner layers (VCC and GND) and (2) a Microinstruction Bus Interconnect Cable/Plug assembly.

6.7.2 Power Requirements

The only power supply voltage required by the WCS module is +5 volts. Connection to the +5 volt supply as well as ground return is established through the module finger/backplane interconnection. The supply voltage tolerance is + or - 5% and the current drawn from the +5 volt supply is 3.A typical (7.34A worst case).

6.7.3 LSI-11 System Bus Backplane Pin Assignment

Figure 6-12 lists the WCS module (M8018) backplane pin assignments.

6.7.4 Microinstruction Bus Connector Pin Assignment

Figure 6-13 contains a table listing the pin assignments for the microinstruction bus interconnect cable/plug assembly. All unlisted numbers have no connection at either end of the cable assembly. The pin assignments are the same at both the processor module end and the WCS module end of the cable. However, because of two series matching registers (pins 24,25) on the end of the cable marked "CPU", (inside the plug assembly), the cable plugs are NOT interchangeable and the end marked "CPU" must be plugged into E75 of the LSI-11 CPU. A continuity check of each signal path will produce a low resistence reading (less than 1 ohm) except for pins 24 and 25. These paths carry the microcycle clock phases and contain a 100 Ohm series resistence on each path.

LSI-11 System Bus Backplane Pin Assignments

Figure 6-12

AE1	MI<22> (SROM 4 H)	BH2	BDAL04 L
AF1	MI<23> (SROM 5 H)	BJ2	BDAL05 L
AJ1	GND	BK2	BDAL06 L
AM1	GND	BL2	BDAL07 L
AT1	GND	BM2	BDAL08 L
AA2	+5 V	BN2	BDAL09 L
AC2	GND	BP2	BDAL10 L
AE2	BDOUT L	BR2	BDAL11 L
AF2	BRPLY L	BS2	BDAL12 L
AH2	BDIN L	BT2	BDAL13 L
AJ2	BSYNC L	BU2	BDAL14 L
AK2	BWTBT L	BV2	BDAL15 L
AM2	BIAKI L (JUMPERED TO AN2)	CJ1	GND
AN2	BIAKO L	CM1	GND
AP2	BBS7 L	CT1	GND
AR2	BDMGI L (JUMPERED TO AS2)	CA2	+5 V
AS2	BDMGO L	CC2	+5 V
AU2	BDAL00 L	CM2	BIAKI L (JUMPERED TO CN2)
AV2	BDAL01 L	CN2	BIAKO L
BA1	BDCOK H	CR2	BDMGI L (JUMPERED TO CS2)
BJ1	GND	CS2	BDMGO L
BM1	GND	DJ1	GND
BT1	GND	DM1	GND
BA2	+5 V	DT1	GND
BE2	BDAL02 L	DA2	+5 V
BF2	BDAL03 L	DC2	GND

MR-1080

Microinstruction Bus Interconnect Cable Pin Assignments

Figure 6-13

CABLE PIN NUMBER (BOTH ENDS)	FUNCTION
7	MIB<15>
8	MIB<14>
9	MIB<13>
10	MIB<12>
11	MIB<16>
12	MIB<17>
13	MIB<18>
14	MIB<19>
15	MIB<20>
16	MIB<21>
19	GND
24	PH1 H (100 OHM SERIES RESISTANCE)
25	PH2 H (100 OHM SERIES RESISTANCE)
26	MIB<11>
27	MIB<10>
28	MIB<9>
29	MIB<8>
30	MIB<7>
31	MIB<6>
32	MIB<5>
33	MIB<4>
34	MIB<3>
35	MIB<2>
37	MIB<1>
38	MIB<0>

MR-1081

Standard TTL Control Bit Functions

Figure 6-15

BINARY	OCTAL	FUNCTION
0000	00	NO FUNCTION
0001	01	RESERVED
0010	02	AVAILABLE
0011	03	AVAILABLE
0100	04	AVAILABLE
0101	05	AVAILABLE
0110	06	AVAILABLE
0111	07	PAGE SWAP (WCS MODE 2 ONLY)
1000	10	RESERVED
1001	11	IFCLR+SRUN L
1010	12	TFCLR L
1011	13	RFSET L
1100	14	INITIALIZE SET
1101	15	FAST DIN
1110	16	PFCLR L
1111	17	EFCLRL

MR-1083

CHAPTER 7

LSI-11 SOFTWARE TOOLS

7.1 GENERAL

The WCS Software Tools consist of 3 utility programs:

1. A Microassembler (MICRO)
2. A WCS Load/Dump Program (WCSLOD)
3. A WCS Debug Program (MODT)

An understanding of MACRO-11 and ODT (as described in the RT-11 documentation) is assumed.

7.2 MICROASSEMBLER (MICRO)

7.2.1 Statement Format

The basic statement format is:

```
LABEL:  OPERATOR OPERAND(S) ;COMMENT
```

The LABEL is a means of symbolically referring to a location in a program and is optional. The first character of a LABEL must be alphabetic, a "$", or a ".". The remaining characters can be any of these or numerics. A LABEL may be any length; however, only the first six characters are significant and, therefore, must be unique among all the LABELs in the source program. All LABELs are terminated by a colon (:), which is not considered part of the LABEL. A symbol used as a LABEL must not be redefined in the source program.

The OPERATOR field contains either a microinstruction mnemonic or a microassembler directive.

The OPERAND field contains additional information to supplement the OPERATOR field. The format of the OPERAND field depends on the particular OPERATOR and in some cases may be omitted.

The COMMENT field must begin with a semicolon (;) and may contain any

information to help document the microprogram. The entire line may be a comment with no other fields.

7.2.2 Expressions

The OPERAND field may contain arithmetic or logical expressions. An expression consists of one or more of the following:

1. Numeric Constant
2. Symbol
3. Current Location Counter (.)
4. Arithmetic or Logical Operators

7.2.2.1 Numeric Constants - MICRO assumes that all numbers are interpreted as octal unless otherwise specified. A constant of a different radix may be specified by one of the following constructs:

^Dn Decimal
^Bn Binary

Where n is one or more numeric characters of the specified radix.

The ASCII equivalent of a character may be specified with the following construct:

'char

where "char" is the printable ASCII character for which the numeric equivalent is desired. For example, the constant 'A evaluates to 101(8).

7.2.2.2 Symbols - There are two types of symbols: Predefined Symbols and User-Defined Symbols.

Predefined symbols consist of the following:

1. Register names
2. Microinstruction Extension Field Names
3. Miscellaneous Symbols

Register names are used to specify an 8-bit internal register (for byte microinstruction) or a 16-bit internal register (for word microinstructions). The Predefined symbols are listed below:

SYMBOL MNEMONIC	VALUE	MEANING FOR A PDP11 MACHINE
RBA	2	bus address
RBAL	2	bus address lower byte
RBAH	3	bus address upper byte
RSRC	4	source operand
RSRCL	4	source operand lower byte
RSRCH	5	source operand upper byte
RDST	6	destination operand
RDSTL	6	destination operand lower byte
RDSTH	7	destination operand upper byte
RIR	10	instruction register
RIRL	10	instruction register lower byte
RIRH	11	instruction register upper byte
RPSW	12	program status word
RPSWL	12	program status lower byte
RPSWH	13	program status upper byte
SP	14	stack pointer
SPL	14	stack pointer lower byte
SPH	15	stack pointer upper byte
PC	16	program counter
PCL	16	program counter lower byte
PCH	17	program counter upper byte
G	0	indirect through G register
GL	0	lower byte indirect through G
GH	1	upper byte indirect through G

Microinstruction Extension Field Names are used to specify Microinstruction bits <23:16>. The predefined symbols are listed below:

MNEMONIC	OCTAL VALUE	MEANING
RSVC	2	Exit microcode after next instruction
LRR	1	Load Return Register
TROFF	200	Stop Trace

The Miscellaneous Symbols will be defined as part of the definition of the microinstruction(s) where they are used.

7.2.2.3 Current Location Counter - Expressions may be constructed to include the current lcoation counter (the address where the instruction will be ultimately be located). The location is indicated by a period (.).

7.2.2.4 Arithmetic or Logical Operators - Expressions are composed of one of the above items or any of the above items connected by one of the following operators:

+	arithmetic sum
-	arithmetic difference
*	arithmetic product
/	arithmetic quotient
&	logical AND
!	logical OR

The expression is evaluated left to right (all operators have equal priority). Angle brackets (<>) can be used to group parts of an expression into a term that is evaluated first as shown in the examples below:

```
1+2*3 equals 11 (octal)

1+<2*3> equals 7 (octal)
```

The negative value of a term may be represented by placing a minus (-) in front of the term.

7.2.3 Microinstructions

In the source formats explained in this section, an "x" is used to represent the extension bit field and a "t" is used to represent the translation symbol.

7.2.3.1 Jump Microinstruction - Format:

```
label:  JMP     address,x,t     ;comment
```

7.2.3.2 Conditional Jump Microinstructions - Format:

```
label:  opcode  address, x,t    ;comment
```

7.2.3.3 Literal Microinstructions - Format:

```
label:  opcode  literal,register,x,t    ;comment
```

7.2.3.4 Two Register Microinstructions - Format:

```
label:  opcode bregister,aregister,x,t ;comment
```

When it is desired to affect the LSI-11 condition codes (i.e., C, V, Z, N), an "F" is appended to the opcode mnemonic (for those opcodes that have the capability of affecting the condition codes).

7.2.3.5 Single Register Microinstructions - Format:

```
label:  opcode  aregister,x,t   ;comment
```

The following symbols are available to reference the status bits read by the CCF instruction:

MNEMONIC	VALUE	MEANING
C8	20	Carry into bit 8
C4	40	Carry into bit 4
ZB	100	Zero
NB	200	Negative

These symbols may be ORed (!) or ADDed (+) together as necessary.

7.2.3.6 Reset and Set Flags Microinstructions (RF and SF) - Format:

```
label:  opcode  flags,x,t       ;comment
```

The following predefined symbols are available for use in the flags field:

MNEMONIC	VALUE	MEANING
I4	1	internal interrupt flag 4
I5	2	internal interrupt flag 5
I6	4	internal interrupt flag 6

These symbols may be ORed (!) or ADDed (+) together as necessary.

7.2.3.7 Input Microinstructions - Format:

```
label:  opcode  accesscode,register,x,t ;comment
```

The following predefined symbols are available for use in the access code field:

MNEMONIC	ACCESS CODE VALUE	MEANING
UB	0	upper byte
UBC	1	upper byte conditionally
LB	2	lower byte
LBC	3	lower byte conditionally
RMW	4	read/modify/write
TG6(only IW)	1	load TR from DAL <15:0>, load G from DAL<6:4>
TG8(only IW)	2	load TR from DAL<15:0>, load G from DAL <8:6>

These symbols may be ORed (!) or ADDed (+) together as necessary.

7.2.3.8 No-Operation Microinstruction (NOP) - Format:

```
label:  NOP     x,t     ;comment
```

7.2.3.9 Load Condition Flags Microinstruction (LCF) - Format:

```
label:  LCF flagenables, register,x,t   ;comment
```

The following predefined symbols are available for use in the flag enable field:

mnemonic	value	meaning
C	1	carry
V	2	overflow
Z	4	zero
N	10	negative

These symbols may be ORed (!) or ADDed (+) together as necessary.

7.2.3.10 Return From Subroutine Microinstruction (RFS) - Format:

```
label:  RFS     x,t     ;comment
```

(Note that an RFS instruction assembles as a JMP 4000)

7.2.3.11 Reset TSR Microinstruction (RTSR) - Format:

```
label:  RTSR    x,t     ;comment
```

7.2.4 Microassembler Directives

7.2.4.1 NXT Directive -

The NXT (next) directive is used as an aid in placing or locating code at the beginning of a block of microinstructions.

```
NXT2
NXT4
NXT10
NXT20
NXT40
NXT100
NXT200
NXT400
```

These directives advance the LC to the next address evenly divisible by the specified power of two. If the current LC satisfies that condition, it is unchanged.

7.2.4.2 TITLE Directive - Format:

```
TITLE program heading title
```

The TITLE directive causes the assembler to list the program heading title on the top of every page following the TITLE directive. This is used only for documentation purposes and has no other effect on the assembly. Only the first 30 characters will be listed on the top of the pages. Examples of the title directive are shown in the sample programs at the end of this section.

7.2.4.3 SBTTL Directive - Format:

```
SBTTL subtitle text
```

The SBTTL directive causes the text to be listed at the top of every page under the title line. This can be used in multi-page listings to make it easier to locate parts of the program. The SBTTL directive has no other effect on the assembly. The maximum length for a sub title is 60 characters.

7.2.4.4 REG Directive - Format:

```
label:  REG     NAME,VALUE
```

The REG directive defines NAME to be a register and gives it the VALUE specified. Also 2 additional names are defined:

```
NAMEL
NAMEH
```

These additional register definitions represent the low and high registers of a pair. The high register is set to the VALUE+1. Because an additional character is concatenated to the name, it can not be more than 5 characters long. Also, the value must be an even number because it will be the lower register of the pair. Registers generated by the REG directive will be listed with an R following the value in the symbol table printed at the end of the assembly listing.

An example of the REG directive is:

```
        REG     RSLT,2
        .
        .
        .
        MW      RSLT,RDST           ; USE THE REGISTER PAIR NAME
        MB      RSLTH,RSRCH         ; OR THE HIGH BYTE
        MB      RSLTL,RIRL          ; OR THE LOW BYTE
```

Note that the symbols RSLTL and RSLT have the same values and can be used interchangably, however, the distinction is made to make the program more readable.

7.2.4.5 LOC Directive - Format:

label: LOC expression

The LOC directive sets the assembler's Location Counter to an absolute value. If the expression is omitted, the Location Counter is set to the value it had prior to the last LOC directive. This is useful for going out of a sequence of instructions then coming back without having to save the Location Counter. If a label is present it will be the value prior to assembling the LOC directive. The following example details this:

```
        LOC     3001        ; SET LOCATION CNTR TO ENTRY AREA
        JMP     DECODE      ; BRANCH TO MICRO INST DECODING

        LOC     3040        ; SET LOCATION CNTR TO NEW AREA
DECODE: CL      175,RIRL
        JZBF    ERROR       ; BRANCH IF ILLEGAL OPCODE

        LOC     3037        ; SET LOCATION COUNTER TO SINGLE
                            ; PLACE TO HANDLE ERROR
ERROR:  JMP     0           ; TRAP ILLEGAL INSTRUCTION

        LOC                 ; SET LOCATION COUNTER BACK TO
        LL      0,RDST      ; PREVIOUS SEQUENCE (3042)
        .
        .
```

7.2.4.6 End Directive - Format:

The END directive causes the assembler to stop reading source from the input file. Any lines following the END statement will be ignored by the assembler. If a file is created with no END statement, an error will be generated by the assembler.

7.2.4.7 Equated Symbols - Format:

```
SYMBOL = EXPRESSION
```

A symbol may be assigned a value by using the equal (=) operator as in the examples:

```
X=3             ; X IS GIVEN A VALUE OF 3
COUNT=X+16      ; COUNT IS GIVEN A VALUE OF 21(8)
TABLE=.+10      ; TABLE IS GIVEN A VALUE OF
                ; THE CURRENT LOCATION COUNTER +10.
```

The symbol value may be reassigned later in the assembly. The value of the symbol will be its last equated value. Equates do not cause any code to be generated and only the value of the symbol to the left of the equal sign is affected.

7.2.4.8 PAGE Directive - Format:

```
PAGE
```

The PAGE directive causes the assembler to skip to the top of the next page in the assembly listing. This only effects the listing and has no other effect on the assembly.

7.2.4.9 MODE Directive - Format:

```
MODE    expression
```

The MODE directive tells the assembler what WCS addressing mode is set on the target WCS module. Subsequent addresses must be in the range for the mode expression.

MODE EXPRESSION	ADDRESS RANGE
1	2000-3777
2	3000-3777 or 13000-13777
3	2000-3777
4	0000-1777

If no MODE directive is present, the default is mode 1 or mode 3. Any other values for the mode expression result in an error.

7.2.5 Using the MICRO Assembler

The assembler is invoked under RT-11 by the following command:

```
.R MICRO
```

The microcode assembler prompts with an asterisk. Command lines are entered in the standard RT-11 format.

```
*OBJFILE, LISTFILE=INPFILE(/SWITCHES)
```

The default input file extensions is .MIC, the output files are .OBJ and .LST.

Both the object file and the listing file are optional. Options selected with the switches are:

/W	loads programs directly into the WCS. This will happen independently of the object file generation.
/C	generates a cross reference of the program symbols.
/B	print a bitmap of all used WCS locations from this assembly.
/N	causes the listings to fit in 80 columns.

When the assembly is complete a message will be printed with the number of assembly errors.

7.2.5.1 Bitmap of Used Memory Locations - The assembler optionally prints a bitmap of memory showing which locations are used and which have not had code generated fo them. Each line of the bitmap represents 100(8) microwords. If a corresponding word had an instruction assembled into it, a 1 would be printed in the place representing the location. If no instruction was assembled into the location, a 0 would appear on the bitmap. If no locations in a 100(8) word segment were used, the line is left completely blank.

The bitmap is useful for locating free WCS memory locations or to modify a program with MODT.

7.2.6 Errors in the Source Program

Errors detected by the assembler are listed with a one character error code at the left of the source line and a description of the error in the statement following. If no listing is being produced, any error messages and the line which caused the error will be printed on the console terminal.

7.2.7 WCS Module Addressing Mode Support

The WCS module will interpret MIB addresses in one of four ways depending on the setting of a DIP switch on the board. Each mode defines what MIB addresses will be mapped to what WCS locations for fetching microinstuctions.

The micro assembler uses the MODE directive (see section 7.2.4.9) to indicate what mode the WCS module will be set to when the microprogram is loaded. If no MODE directive is present, the assembler assumes the program is for a mode 1 or mode 3 WCS module.

In MODE 2, the low and the high 512 word banks of WCS map to MIB addresses 3000-3777. To differentiate between the two banks of the RAM, the assembler uses addresses 13000-13777 to refer to the high bank. Microcode assembled for the low half of the RAM will be listed with addresses 3000-3777. If the program executes a JMP from one bank to the other, the assembler inserts a 34(8) in the extension bits field to cause the WCS module to switch to the alternate bank. To set the assembler's Location Counter to the high bank of RAM, the LOC directive is used. For example:

```
        LOC     3040
        JMP     HIGH            ;JMP TO HIGH BANK
                                ;EXTENSION BITS EQUAL
                                ;34(OCTAL)
        LOC     13477
HIGH:   MB      RDST,G          ;THIS INSTRUCTION IS IN
                                ;HIGH BANK OF RAM
```

7.3 LOADING AND SAVING WRITABLE CONTROL STORE (WCSLOD)

The loader accepts up to six object modules as input and can generate one loadable output file.

7.3.1 Loading Object Modules

At start up the loader will, by default, initialize all of WCS with:

```
JMP     0,200       ;  disable trace and trap to LSI-11 LOC 10
```

Upon termination the loader "enables" WCS thereby allowing the microprograms to be executed. The switches available are:

```
/D   disable WCS upon exit
/O   suppress notification of memory overlap
/N   chain to MODT upon exit
/R   do not initialize before loading
/T   suppress notification of translation array conflicts
```

Examples:

```
.R WCSLOD
*A,B,C
.
```

WCS has been initialized, the three files A.OBJ, B.OBJ, C.OBJ have been loaded, and WCS has been enabled for execution.

```
.R WCSLOD
*PPR3.OBJ/R
.
```

WCS is not initialized but PPR3.OBJ is loaded and WCS is enabled.

```
.R WCSLOD
*PPR3.OBJ/N
MODT V01.01
*
```

WCS is initialized, PPR3 is loaded, and the loader has chained to MODT.

```
.R WCSLOD
*<CR>
```

WCS is initialized.

7.3.2 Saving the Contents of WCS

After a session of debugging a microprogram with MICRO ODT (see Section 7.8.3) one might like to save the contents for later execution. The syntax for this is

```
.R WCSLOD
*ABC=/U
.
```

This creates the file ABC.OBJ containing the entire address space of WCS. This file can then be reloaded by WCSLOD.

7.4 MICRO ODT (MODT)

MODT is a symbolic microcode debugging tool. It can be linked (using the RT-11 LINK utility) with a macro program being debugged or it can be run by itself. The command syntax is identical to that of ODT-11 with enhancements to support the micromachine.

All ODT-11 commands are available in MODT. The additional features to support the microprogrammer are:

- Symbolic examination and modification of any location in the WCS module.
- Tracing of any 16 sequential microinstruction cycles for debugging.
- Stopping the execution of a microprogram at a given location to examine registers.
- Starting execution of microcode at any location in the micromachine.

- Automatic range checking on addresses typed to ensure the locations are consistent with the WCS module addressing mode in the host machine. Invalid addresses will cause an error to be printed. The valid addresses are:

WCS MODE	ADDRESS RANGE
1	2000-3777
2	3000-3777 or 13000-13777
3	2000-3777
4	0000-1777

These features will be described using a specific microprogram as an example. The microprogram adds the contents of PDP-11 registers R0 and R1 and stores the result in R2.

7.4.1 Symbolic Examination and Modifications of WCS Locations

The colon (:) command opens the WCS location whose address is specified before the colon. The operation of this command is similiar to that of the slash (/) of ODT-11, except the location opened is in the WCS and not PDP-11 main memory.

```
*3001:  JMP     0,200 (CR)
```

Just as in ODT-11, the line feed key opens the next sequential WCS location and the caret (^) opens the previous location.

```
*3040:  LL      0,RDST (LF)
3041:   LGL     RDST (LF)
3042:   MW      G,RDRC (CR)
*
```

The commands "commercial at" (@) and "underscore" (_) can also be used with microinstruction in the same way as in ODT-11. Since the binary for the microinstruction JMP is zero, the commercial at can be used to examine the target of a JMP instruction. The target of a conditional branch can also be examined with the underscore (_).

```
*3001:  JMP     3040 @
3040:   LL      0,RSRC (CR)
3155:   JZBT    3170
3170:   MW      RSRC,RDST (CR)
```

Once a location is opened with the colon, it can then be modified symbolically by typing the new contents of the location.

```
*3002:  JMP     0,200    JMP 3040 (CR)
```

Terminating a modification with a line feed or a caret will perform the modification and then open the next or previous sequential location. Using the line feed allows an entire program to be entered with MODT as shown below:

Example of a Microprogram to Compute R2=R0+R1
Entered Completely with MODT

```
*3001:  JMP     0,200    JMP 3040 (CR)
*3040:  JMP     0,200    LL 0,RSRC (LF)
3041:   JMP     0,200    LGL RSRC (LF)
3042:   JMP     0,200    MW G,RDST (LF)
3043:   JMP     0,200    LL 1,RSRC (LF)
3044:   JMP     0,200    LGL RSRC (LF)
3045:   JMP     0,200    AW G,RDST (LF)
3046:   JMP     0,200    LL 2,RSRC (LF)
3057:   JMP     0,200    LGL RSRC,RSVC (LF)
3050:   JMP     0,200    MW RDST, G,TROFF (CR)
```

Each microinstruction mnemonic is followed by one or more spaces, and operands are separated by commas. Notice the registers are typed symbolically using the same names as the standard MICRO default registers. Also the predefined symbols RSVC, LRR and TROFF are accepted in the extension bits field. TROFF causes the hardware to stop tracing microaddresses.

7.4.2 Executing a Microprogram - Format:

```
address;M
```

Normally a microprogram is invoked by a machine language program which executes an 0767XX instruction. A microprogram can also be executed by itself using the M command. Typing this command causes MODT to place a JMP microinstruction to the specified address in microlocation 3001. Then the PDP-11 instruction 076700 is executed to transfer control to the microprogram. To execute the Register Add example above, first set the PDP-11 registers to specific values.

```
*$0/     037246  4 (CR)
*$1/     177640  3 (CR)
*$2/     000243  0 (CR)
```

Then transfer control to the microprogram.

```
*3040;M
```

The registers can now be examined to see that the microprogram executed properly.

```
*$0/     000004 (CR)
*$1/     000003 (CR)
*$2/     000007 (CR)
```

7.4.3 Dump Points - Format:

```
address;D
```

To aid in debugging a microprogram, a facility is provided to stop execution of a microprogram and examine some of the internal registers. There are three restrictions with this facility:

1. Only the registers RSRC, RDST, RBA and RIR can be displayed. Other registers, RPSW, the return register, and the G register cannot be displayed.

2. Once a dump point is reached, there is no way to proceed.

3. The highest 24 locations in the second WCS bank of memory are used by MODT.

The restrictions exist because the hardware allows no way to access the internal registers of the micromachine.

The internal micromachine registers may be examined only following a Dump point. This is done by typing an ampersand (&) followed by the register name or number. Only the registers RSRC, RBA, RIR and RDST may be examined. For example:

```
*&RIR/   004367
```

The register may be examined as two bytes by typing a backslash following the register name. For example:

```
*&RIR\   010     367
         HIGH    LOW
         BYTE    BYTE
```

Dump points can be used to determine which of several paths a microprogram has taken and to give information about the state of a program. In the previous microprogram (section 7.4.1), a dump point could be inserted as follows:

```
*3043;D
```

The example can then be executed:

```
*$0/     037246  4 (CR)
*$1/     177640  3 (CR)
*$2/     000243  0 (CR)
*3040;M
DP @addr
```

The "addr" displayed is the address following the 0767XX which caused microcode to be entered. The microregisters could then be examined:

```
*&RSRC/  000000
*&RDST/  000004
```

7.4.4 Tracing Microprograms - Format:

```
T
```

Another debugging aid is the Trace facility in the WCS hardware. When a program is started, the WCS module will store the last 16 microaddresses from the microinsruction bus (MIB). The tracing continues until an instruction with the TROFF (200) bit is executed. By setting this bit on the last instruction of a complicated sequence, flow through several microinstructions can be seen. By executing the previous example, the operation of the T command can be seen.

```
*3040;M
*T
```

The trace output will be in reverse execution order. The first instruction printed is the last executed. Addresses with the line (WAIT) following indicate wait cycles (the extra cycles of multicycle instructions) or cycles where the instructions were fetched from the base machine microcode.

7.4.5 Transferring to WCSLOD - Format:

```
N
```

Control can be transferred to the WCS loader (WCSLOD) by using the N command. The N command is shorthand for the sequence:

```
*(CONTROL/C)
.R WCSLOD
```

7.4.6 Using MODT

MODT can either be linked by itself or included with MACRO programs. It is provided as a .OBJ file with the entry point MODT.

7.4.6.1 Using MODT as a SAVE File - If MODT is to be used by itself it must first be linked using the RT-11 command:

```
.LINK MODT
```

This will produce a SAV file which can be executed by typing:

```
.R MODT
MODT V01.01
*
```

7.4.6.2 Using MODT with a MACRO-11 Object Program - To debug an application involving both MACRO-11 programs and microprograms, MODT can be linked with a MACRO-11 object program in the same way as ODT. This allows the microprogram to work with the actual data structures set up by the MACRO-11 program.

MODT can be linked either before the program or after it by using the /TRANSFER switch to the linker.

```
.LINK PROG1,PROG2,MODT/TRANSFER (CR)
TRANSFER ADDRESS?  MODT (CR)

.LINK /EXEC:PROG1,MODT,PROG1,PROG2
```

In the second example, the program will automatically start in MODT. Also the /EXEC switch causes the .SAV file to be named PROG1.SAV instead of MODT.SAV.

CHAPTER 8

MICROPROGRAMMING TECHNIQUES

8.1 GENERAL

This chapter presents some techniques which are basic to LSI-11 microprogramming.

There are four entry points to user control store, microlocations 3000 through 3003 (octal). Techniques for utilizing these entry points are discussed below.

In addition, the External Device and Event Line interrupts can be sampled during execution of user microcode and if present, will transfer control to microlocation 3004 (octal).

The normal means of transferring control to user control store is to execute a user machine instruction in the range 076700 through 076777.

Each separate user machine instruction is implemented with a sequence of microinstructions. In many cases these microinstruction sequences will be largely independent routines and a decoding function is required to pass control to the appropriate sequence.

Optionally, the microinstruction can set or clear condition code bits in the Processor Status Word (PSW). The microprogrammer must decide which condition code flag state properly reflects the operation just completed.

The final step in the execution of a machine instruction is to rejoin the LSI-11 machine operating cycle. This action provides for servicing traps and interrupts and subsequently for fetching the next machine instruction.

8.2 USER MICROPROGRAMMING ENTRY POINTS

The user control store entry points are microaddresses 3000 through 3003 inclusive. Control is transferred to the microinstruction stored at these locations as explained in this section. In the normal application, not all entry points will be utilized. The unused locations should contain a JMP 0 microinstruction so that entry at these locations will result in a reserved instruction trap to LSI-11 vector location 10.

There are two ways that control is transferred to the entry points: (1) after an appropriate machine instruction is fetched and decoded a transfer to one of three microaddresses (3000,3001,3003) occurs; (2) By configuring the LSI-11 power-up jumpers, control can be transferred to 3002 at power-up.

8.2.1 Entry Microaddress 3000

Control is transferred to microaddress 3000 whenever a machine instruction in the range 000220 to 000227 (octal) is fetched and decoded. Note that these machine instructions are reserved by Digital. A JMP 0 microinstruction must always be assembled into microlocation 3000 to cause a trap to LSI-11 vector location 10.

8.2.2 Entry Microaddress 3001

Microaddress 3001 is the microcode entry point for user microprogramming. Control is transferred to microaddress 3001 whenever a machine instruction in the range 076000 to 076777 is fetched and decoded. Note that the only legal customer opcodes are in the range of 076700 to 076777. It is the responsibility of the microprogrammer to cause a JMP 0 microinstruction to be executed for any opcode in the range 076000 to 076677 since these are reserved by Digital.

8.2.3 Entry Microaddress 3002

Control is transferred to microaddress 3002 at power-up when Power-Up Mode 3 is selected. Power-Up Mode selection is made via a jumper option on the LSI-11 processor module (see The Microcomputer Handbook). The microinstruction located at 3002 should jump to a microroutine which executes special start-up operations, or possibly a system bootstrap.

Since the WCS memory is volatile, a microcoded power-up routine can not be executed out of User Control Store.

8.2.4 Entry Microaddress 3003

Control is transferred to microaddress 3003 whenever a machine instruction in the range 075040-075777 is fetched and decoded. Note that these machine instructions are reserved by Digital and a JMP 0 microinstruction must always be assembled into microlocation 3003 to cause a trap to LSI-11 vector location 10.

8.2.5 Entry Microaddress Summary

Because the microaddress entry points are positioned in sequential locations, the microinstructions located in the entry area 3000 to 3003 must all be unconditional jumps. Each JMP microinstruction then

transfers control to a microroutine which implements the appropriate operations. The source code for the entry area normally appears as follows:

```
        LOC     3000        ; SET MICROASSEMBLER COUNTER LOCATION
A3000:  JMP     0           ; TRAP RESERVED OPCODES 00022X
A3001:  JMP     DECODE      ; ENTER HERE FOR OPCODES 076XXX
A3002:  JMP     PWRUP       ; ENTER HERE FOR POWER-UP
A3003:  JMP     0           ; TRAP RESERVED OPCODES 075040-075777


DECODE:                     ; START OF USER OPCODE DECODE

PWRUP:                      ; START OF POWER UP ROUTINE
```

8.3 MACHINE INSTRUCTION DECODING TECHNIQUES

Control will be transferred to user entry point 3001 in response to machine instructions in the range 076000-076777. This instruction is contained in micromachine register RIR. The low order 8 bits of the instruction are in RIRH and the high order 8 bits are in RIRL. The microprogrammer's first task is to determine which machine instruction has been fetched. Note that opcodes 076700 to 076777 are the only legal customer opcodes and opcodes 076000 to 076677 must cause a JMP 0 microinstruction to be executed.

8.3.1 Successive Comparison Decoding

One technique for decoding user opcodes is by successive comparison. An implementation of this technique is to use a Compare Literal Microinstruction (CL) for each expected user opcode. The sequence of microinstructions which implements this technique is as follows:

```
DECODE: CL      175,RIRL    ; IS IT 076400 TO 076700?
        JZBF    ERROR       ; NO, GO TRAP
        CL      300,RIRH    ; IS IT 076700?
        JZBT    OP00        ; YES, GO EXECUTE
        CL      301,RIRH    ; IS IT 076701?
        JZBT    OP01        ; YES, GO EXECUTE
        CL      302,RIRH    ; IS IT 076702?
        JZBT    OP02        ; YES, GO EXECUTE
        .
        .
        .
        1 additional comparison for each opcode implemented
        .
        .
        .
ERROR:  JMP     0           ; IF NONE OF THEM, TRAP TO VECTOR 10.
```

8.3.2 Modified Jump Decoding

This microinstruction decoding technique allows user opcodes to be decoded more efficiently than the successive comparison technique, especially when a large number of opcodes are implemented.

This technique uses the lower six bits of the low byte of the machine instruction to dispatch to the appropriate microcode routine. Note that RPSWL contains all zeroes upon entry into user control store. The assembled address of the JMP instruction is modified to transfer control into a dispatch table of JMP instructions. This technique is illustrated in the following example:

```
DECODE: CL      175,RIRL        ; IS IT A LEGAL USER OPCODE?
        JZBT    DC1             ; MAYBE
DC0:    JMP     0               ; IF NOT, TRAP TO VECTOR 10
DC1:    AL      100,RIRH        ; IF LEGAL, C8=1 AND RIRH<7:6>=00
        JC8F    DC0             ; NO, GO TRAP
        MI      RPSWL,RIRH      ; MODIFY THE JMP BITS <5:0> WITH
        JMP     3200            ; MACHINE INSTRUCTION BITS <5:0>


        LOC     3200            ; THIS TABLE ADDRESS MUST HAVE
                                ; BITS <5:0>=0
DISPAT: JMP     OP00            ; JUMP FOR OPCODE 076700
        JMP     OP01            ; JUMP FOR OPCODE 076701
        JMP     OP02            ; JUMP FOR OPCODE 076702
        .
        .
```

In this example, the starting microaddress of the dispatch table is 3200 octal. Micromachine control is transferred here when a machine opcode of 076700 is decoded. Control is subsequently transferred to microaddress OP00 where the microprogram appropriate to the execution of the 076700 machine opcode begins.

8.4 PASSING OPERANDS TO USER MACHINE INSTRUCTIONS

LSI-11 data manipulation machine instructions allow very flexible operand addressing. In the design of new machine instructions, the microprogrammer must provide some means for delivering the operands to the micromachine for processing.

8.4.1 Predefined Operand Addressing

In less general applications, the operands of a user-defined instruction may be in specific LSI-11 General Purpose Registers or machine memory locations.

An example of this technique is to place the data word(s) in location(s) immediately following the instruction in main memory. To access a data word, a sequence similar to the following would be used:

```
        RIW2    PCH,PCL ; FETCH THE WORD AND UPDATE THE PC
        IW      ,RSRC   ; PUT THE DATA IN RSRC
```

8.4.2 Register Operand Addressing

Since user opcodes can be in the range 076700 through 076777, the low order 6 bits of the instruction can be utilized in any manner. In particular, the lower 3 bits could specify one of the 8 LSI-11 General Purpose Registers that will be used as an operand of the instruction.

The specific General Purpose Register can then be accessed in the following manner:

```
LGL     RIRH      ; LOAD G REGISTER
MW      G,RSRC    ; PUT CONTENTS OF REGISTER INTO RSRC
```

Note that when this technique is used, the number of unique instructions that can be specified is reduced.

Additional General Purpose Registers can be specified by assembling one or more 16-bit words following the instruction in main memory in the following format:

Bits <8:6> = 2nd register number

Bits <2:0> = 3rd register number

The register can then be accessed by the following:

```
RIW2    PCH,PCL   ; FETCH THE WORD AND UPDATED PC
IW      TG8,RIRL  ; LOAD G FROM BITS <8:6>
MW      G,RDST    ; PUT CONTENTS OF 2ND REGISTER IN RDST
LGL     RIRH      ; LOAD G REGISTER
MW      G,RBA     ; PUT CONTENTS OF 3RD REGISTER IN RBA
```

8.5 MICROPROGRAMMING THE USER MACHINE INSTRUCTION

8.5.1 Defining The Instruction

The first step in creating a new machine instruction is to define the functions the instruction is to accomplish. One approach is to program the desired function(s) first in LSI-11 machine-level assembly language. This approach usually will also reveal the suitability of microprogramming to accomplish a desired collection of functions.

8.5.2 Documenting the Instruction

The requirements for instruction documentation will vary with the nature of the functions provided. Such documentation includes information on input and output operands, resultant PSW condition code flags for all possible situations, and execution timing for all possible situations.

8.5.3 Temporary Flag Use

During the execution of user-designed machine instructions it is often necessary to monitor ALU results. These results are available in the status bit and condition code flag register and may be used to effect microprogram control via the Conditional Jump microinstructions. However, when the machine instruction operations have been completed, additional microinstructions may be needed to update the Processor Status Word condition code flags, (N,Z,V,C). Note that the PSW resides in a microprocessor register and that it is partially separated from the ALU flag register. The microprogrammer must decide what PSW flag states accurately represent the operation executed and provide for their implementation.

8.5.4 Executing Machine-Level I/O Operations

The microprogrammer has complete freedom in implementing any of the 5 possible machine-level I/O operations (DATI, DATO, DATOB, DATIO, DATIOB) as part of a new machine instruction. In designing the I/O portion of the instruction, the microprogrammer should make detailed reference to Chapter 5. All I/O operations, other than Input Status and Output Status, require a response from a system bus device and effectively transfer control outside the processor. Under normal conditions, a non-responding bus device will cause a bus error trap. The microprogrammer must recognize the possibility of non-responding bus devices.

8.5.4.1 Bus Error Trap Control - The microprocessor register RPSWL contains flags designated for bus error trap control. Prior to executing any of the five I/O operations (DATI, DATO, DATOB, DATIO, DATIOB), the contents of RPSWL must be zero. This will ensure that a bus timeout error will be handled in the normal way, as a trap to system memory location 4.

8.5.5 Scratch Register Usage

Many processes require scratch register space in which to store intermediate results. This section discusses the function of each microprocessor register and the conditions under which it may be used by the microprogrammer. Note that register names refect the conventions used in the PDP-11 emulation microcode and in no way dictate required usage.

8.5.5.1 Source Operand Register (RSRC) - The source operand register may be used for scratch storage during microprogram execution. It may be left in ye conclusion of the user microprogram.

8.5.5.2 Destination Operand Register (RDST) - The destination operand register may be used for scratch storage during microprogram execution. It may be left in any state at the conclusion of the user microprogram.

8.5.5.3 Instruction Register (RIR) - Every machine instruction, whether standard or user-designed, is loaded into the instruction register as part of the machine instruction fetch operation. The low byte of the machine instruction is loaded into RIRH. Similarly, the high byte of the machine instruction is loaded into RIRL. RIR can be used for scratch storage after the instruction is decoded. If the user opcode transmits no information to the micromachine the microprogram may then use the upper and lower byte of RIR for scratch storage.

8.5.5.4 Bus Address Register (RBA) - The bus address register may be used for scratch storage during microprogram execution. It may be left in any state at the conclusion of the user microprogram.

8.5.5.5 LSI-11 Processor Registers (R0-R5) - The LSI-11 processor General Purpose Registers R0 through R5 can be used for scratch storage during microprogram execution. Instruction results can be left in General Purpose Registers. Note that an access of a General Purpose Register requires the G register to contain the proper register number to indirectly access the general purpose register.

8.5.5.6 LSI-11 Stack Pointer (R6) and Program Counter (R7) - The Stack Pointer and Program Counter should never be used as scratch registers. However, a new instruction may pass parameters on the stack. Note that the Program Counter and the Stack Pointer can be accessed directly (without the G register).

8.5.5.7 Processor Status Word Register (RPSW) - The processor status word register serves two purposes: (1) RPSWH contains a copy of LSI-11 PSW bits <7:4>; (2) RPSWL contains a code to indicate which type of main memory bus timeout error has occurred.

The LSI-11 PSW is formed by the logical OR of RPSWH and four condition code flags. Therefore, bits <3:0> of RPSWH must always be 0.

Since a main memory bus timeout can occur not only during machine instruction execution, but also during console ODT routines, RPSWL contains a code to indicate which type of bus timeout error has occurred. Therefore, RPSWL will be a 0 after a machine instruction fetch and must be maintained as a 0 during any I/O operation.

Block Move Example

Figure 8-1-1

```
                        MICRO ASSEMBLER V01.01  00W

1                       ;THIS IS AN LSI-11 MICRO CODE SUBROUTINE
2                       ;TO MOVE A BLOCK OF MEMORY FROM ONE PLACE TO SOME PLACE
3                       ;ELSE.
4                       ;       THIS ROUTINE ILLUSTRATES...
5                       ;       1) HOW A LONG RUNNING MICROCODE SUBROUTINE ENSURES THAT
6                       ;          PENDING INTERRUPTS GET SERVICED IN A TIMELY FASHION.
7                       ;       THE INPUT PARAMETERS ARE
8                       ;       R0:SOURCE ADDRESS
9                       ;       R1:DESTINATION ADDRESS
10                      ;       R2:NR. OF WORDS TO BE MOVED
11      3000                    LOC     3000
12      3000 000 000000         JMP     0
13      3001 000 003014         JMP     MOV
14      3002 000 000000         JMP     0
15      3003 000 000000 ERROR:  JMP     0
16      3004                    LOC     3004            ;UNNECESSARY,JUST FOR CLARITY.
17                      ;AN INTERRUPT HAS OCCURRED.  MUST SUSPEND THIS OPERATION
18                      ;IN SUCH A WAY AS TO ALLOW THE OPERATION TO BE RESUMED RATHER
19                      ;THAN RESTARTED AFTER THE INTERRUPT HAS BEEN PROCESSED
20                      ;TO ACCOMPLISH THIS IT IS NECESSARY TO UPDATE THE INPUT PARAMETERS
21                      ;AND BACK UP THE BASE MACHINE PROGRAM COUNTER, THEN EXIT
22                      ;MICROCODE.  IN THIS WAY THE INTERRUPT WILL BE PROCESSED, AND
23                      ;THE NEXT INSTRUCTION FETCH WILL AGAIN EXECUTE THE 076000 INSTRUCTION
24                      ;ONLY NOW WITH UPDATED VALUES IN THE SOURCE, DESTINATION, AND SIZE
25                      ;PARAMETERS
26      3004 000 060011         LL      0,RIRH  ;UPDATE LSI-11 R0 TO POINT TO
27      3005 000 072411         LGL     RIRH    ; NEXT SOURCE WORD
28      3006 000 101100         MW      RSRC,G
29      3007 000 060031         LL      1,RIRH  ;UPDATE LSI-11 R1 TO POINT TO
30      3010 000 072411         LGL     RIRH    ; NEXT DESTINATION WORD
31      3011 000 101140         MW      RDST,G
32      3012 002 027756         AL      376,PCL,RSVC    ;DECREMENT BASE MACHINE PC BY TWO
33      3013 000 073417         CDB     PCH             ;AND EXIT MICROCODE
34                      ;PROCESS THE BLOCK MOVE INSTRUCTION
35      3014 000 033730 MOV:    CL      175,RIRL        ;LEGAL INSTRUCTION?
36      3015 000 010003         JZBF    ERROR
37      3016 000 022011         AL      100,RIRH        ;IF LEGAL C8=1
38      3017 000 011003         JC8F    ERROR
39      3020 000 060011         LL      0,RIPH  ;COPY CONTENTS OF LSI-11 R0
40      3021 000 072411         LGL     RIRH    ; INTO MICRO-REG RSRC
41      3022 000 101004         MW      G,RSRC  ;GET SOURCE BLOCK ADDR
42      3023 000 060031         LL      1,RIRH  ;COPY CONTENTS OF LSI-11 R1
43      3024 000 072411         LGL     RIRH    ; INTO MICRO-REG RDST
44      3025 000 101006         MW      G,RDST  ;GET DEST BLOCK ADDR
45                                              ;FROM THIS POINT ON
46      3026 000 060051         LL      2,RIPH  ;MICRO REG G WILL ALWAYS POINT TO
47      3027 000 072411         LGL     RIRH    ; LSI-11 REG R2 (WORD COUNT REG)
48      3030 000 143000         TW      G,G     ;IS THE WORD COUNT REG (R2) = 0?
49      3031 000 010443         JZBT    EXIT
50      3032 000 173124 LOOP:   RIW2    RSRCH,RSRCL     ;PUT SOURCE ADDR. ON DATA ACCESS LINES.
51                                              ;(ALSO BUMPING SOURCE ADDR POINTER).
52      3033 000 137400         DW1F    G,G             ;TAKE ADVANTAGE OF IDLE TIME TO UPDATE
53                                              ;CONTENTS OF LSI-11 R2
54      3034 000 161002         IW      ,RBA    ;COPY SOURCE OPERAND INTO SCRATCH
55      3035 000 173566         WIW2    RDSTH,RDSTL     ;THEN MOVE WORD BACK OUT TO DEST
56      3036 000 176462         OW      RBAH,RBAL       ; BLOCK ALSO BUMPING DEST ADDR
```

MR-1319

Block Move Example

Figure 8-1-2

```
                                MICRO ASSEMBLER V01.01  00W

57    3037 000 070500                   SI       I6       ;THIS FLAG ALTERS THE INTERRUPT DECISION CHAIN
58                                                        ;SUCH THAT AFTER THE EXECUTION
59                                                        ;OF A MICRO CODE RSVC, CONTROL
60                                                        ;WILL BE RETURNED TO THE MICROCODE
61                                                        ;IN ONE OF TWO WAYS. IF AN INTERRUPT
62                                                        ;IS PENDING CONTROL WILL GO
63.                                                       ;TO MICRO LOCATION 3004. IF NO
64                                                        ;INTERRUPTS PENDING CONTROL WILL
65                                                        ;GO TO THE INSTRUCTION FOLLOWING
66                                                        ;THE RI INSTRUCTION.
67    3040 002 177400                   NOP      RSVC     ;EXIT TO SERVICE ANY INTERRUPT
68    3041 000 070100                   RI       I6       ;RESET FLAG
69                                                        ;CONTROL RETURNS HERE IF NO
70                                                        ;INTERRUPT.
71    3042 000 014032                   JZF      LOOP     ;IF ANY MORE WORDS, MOVE THEM.
72    3043 002 177400             EXIT: NOP      RSVC     ; ELSE, RETURN TO BASE MACHINE
73    3044 000 177400                   NOP
74    3045                              .END

                                MICRO ASSEMBLER V01.01  00WSYMBOL TABLE

C       = 0001    C4      = 0040    C8      = 0020    ERROR     3003    EXIT      3043
G       = 0000R   GH      = 0001R   GL      = 0000R   I4      = 0001    I5      = 0002
I6      = 0004    LB      = 0001    LBC     = 0003    LOOP      3032    LRR     = 0001X
MOV       3014    N       = 0010    NB      = 0200    PC      = 0016R   PCH     = 0017R
PCL     = 0016R   RBA     = 0002R   RBAH    = 0003R   RBAL    = 0002R   RDST    = 0006R
RDSTH   = 0007R   RDSTL   = 0006R   RIR     = 0010R   RIRH    = 0011R   RIRL    = 0010R
RMW     = 0004    RPSW    = 0012R   RPSWH   = 0013R   RPSWL   = 0012R   RSRC    = 0004R
RSRCH   = 0005R   RSRCL   = 0004R   RSVC    = 0002X   SP      = 0014R   SPH     = 0015R
SPL     = 0014R   TG6     = 0001    TG8     = 0002    TGL     = 0034X   TROFF   = 0200X
UB      = 0000    UBC     = 0002    V       = 0002    Z       = 0004    ZB      = 0100

MR-1015
```

8.7 MICROPROGRAMMING USER-DEFINED TRAP VECTORS

New user-defined trap operations may be implemented by the microprogrammer. This is accomplished by setting up a vector address and transferring control to the base microcode routine which executes all other LSI-11 trap operations.

8.7.1 Creating the Vector Addresses

The vector address of the special user trap is inserted into the 16-bit source operand scratch register, RSRC. Usually this is done with two consecutive Load Literal microinstructions. RSRCH receives the high byte of the vector address and RSRCL the low byte.

8.7.2 Joining the Base Microcode

Once the 16-bit vector address has been loaded into RSRC, an unconditional jump microinstruction transfers control to the base microcode routine which executes the trap operation. The microinstruction sequence is as follows:

```
LL      VEC LOW BYTE, RSRCL     ; LOAD VECTOR LOW BYTE
LL      VEC HIGH BYTE, RSRCH    ; LOAD VECTOR HIGH BYTE
JMP     1402                    ; JUMP TO TRAP ROUTINE
```

8.8 MICROPROGRAMMING SYNCHRONIZED CONTROL SIGNALS

A microinstruction word contains four TTL Control Bits which are available from the LSI-11 CPU module at the backplane. Two additional Extended TTL Control Bits are available from the WCS module at the backplane. These TTL Control Bits can be used for high speed control of external logic.

8.8.1 Standard TTL Control Bits

The standard TTL control bit codes are assembled into MI<21:18>. Of the 16 possible codes, 8 are employed by the LSI-11 interface circuitry. One additional code, 07, is used to swap user control store pages for WCS address mode 2. Codes 02 through 06 and 10 may be used by the microprogrammer. These codes must be decoded by external hardware which is connected to the system backplane. Chapter 6 presents information on standard TTL control function codes as well as hardware connection details.

8.8.2 Extended TTL Control Bits

The WCS extended TTL control bits (MI<23:22>) are assembled into the same source field as the standard TTL control bits.

The assembly value corresponding to MI<22> is 100 and that corresponding to MI<23> is 200. These values may be ORed with the standard TTL control bit code, as shown below:

```
LL      0,RSRCL, 14!100!200    ; CONTROL CODE 03
                               ; PLUS MI<22> PLUS MI<23>
```

Note that MI<23> is shared with the microaddress trace RAM logic. This has no effect on MI<23> but it may complicate microprogram debugging.

8.9 CONTROLLING THE MICROINSTRUCTION FLOW

The two means of controlling microinstruction flow available to the microprogrammer are: (1) the RSVC bit and (2) the jump microinstructions. The RSVC bit is MI<17> and is a direct control input to the microprocessor Control chip translation array. The jump microinstructions include both conditional and unconditional jumps as well as a Return From Subroutine (RFS) microinstruction.

The microprocessor Control chip can also modify microinstruction flow as a function of Location Counter and translation register contents, but this facility is not at the microprogrammer's disposal.

Figure 8-2 contains a list of microlocations (normally used for the EIS/FIS microcode) that invoke such translations. The user should either (1) avoid assembling microcode in any of these locations, or (2) assemble only a JMP or RFS microinstruction in those locations (which will override the translation).

8.9.1 Leaving User Control Store

The function of the RSVC bit, MI<17>, is to return control to the beginning of the interrupt and trap interrogation sequence. The RSVC bit provides the only means for transferring control to this point. The RSVC bit must be set one microinstruction before the translation is to be invoked as shown in the example below:

```
        NOP     RSVC    ; EXIT USER CONTROL STORE AFTER THE
        NOP             ; NEXT MICROINSTRUCTION
```

Note that any microinstruction other than a jump (conditional or unconditional) or RFS can be used with the RSVC bit or as the subsequent microinstruction.

EIS/FIS Translation Locations

Figure 8-2

Micro-address	Translation	Micro-address	Translation
2033	EII	2552	RET
2072	EII	2553	RET
2123	PSW	2571	DMW
2172	PSW	2604	EII
2220	FII	2614	EII
2254	FII	2622	PSW
2274	FII	2630	PSW
2320	FII	2644	EII
2406	FII	2654	EII
2447	EII	2700	EII
2500	EII	2710	EII
2516	PSW	2714	PSW
2540	EII	2717	PSW
2550	RET	2740	EII
2551	RET	2750	EII
		2754	PSW

MR 1031

8.9.2 Jump to Subroutine and Return Microinstructions

The Jump to Subroutine (JSR) Microinstruction replaces the entire location counter contents with the 11-bits assembled into MI<10:0> and loads the Return Register with the updated Location Counter. The JSR microinstruction is a normal JMP microinstruction with the LRR bit (MI<16>) set to a 1. The Return From Subroutine (RFS) microinstruction replaces the entire Location Counter contents with the Return Register contents and will also override translations. The following example demonstrates these two microinstructions:

```
        .
        .
        .
        JSR     SUB         ; SUBROUTINE JUMP AND RETURN
        .
        .
        .
SUB:    NOP                 ; DUMMY
        NOP                 ; DUMMY
        RFS                 ; RETURN
```

8.9.3 Conditional Jump Microinstruction

The Conditional Jump microinstruction affects only the lower 8 bits of the Location Counter. The upper 3 bits remain the same from the updated Location Counter. Since only 8 bits may be modified, the conditional jump page is only 256 microaddresses in length. The microprogrammer is cautioned against placing conditional jumps in the last location of a (256 microaddress) page because the top four bits will normally be incremented during the second microcycle, causing control to transfer to the next page.

CHAPTER 9

INSTALLATION

9.1 GENERAL

This chapter contains procedures for unpacking, installation and initial checkout for the LSI-11 WCS options.

9.2 UNPACKING AND INSPECTION

The LSI-11 WCS is packaged in accordance with commercial packaging practices. Remove all packing material and check the equipment against the shipping list. Table 9-1 lists the items supplied per configuration. Report any damage shortages to the shipper immediately and notify the Digital representitive. Inspect all parts and carefully inspect the circuit boards for cracks, loose components and separations in the etched paths.

NOTE

If Digital Field Service Installation has been contracted, then the customer should not break any seals on the shipping containers.

9.3 INSTALLATION PROCEDURE

The following procedures should be followed to properly install the M8018 WCS module option in an LSI-11 system.

9.3.1 Switch Configurations

The range of microcode addresses to which the WCS will respond on the Microinstruction Bus (MIB) (when the CSR Enable Bit is a "1") is determined by an 8 wide DIP Switch (SW1) on the M8018 module.

Items Supplied Per Configuration

Table 9-1

ITEM	MODELS			
	KUV11-UH	KD11-WA	11/03-WC	11/03-WD
KD11-H CPU	X			
M8018 WCS MODULE	X	X	X	X
WCS CABLE Part NO 17-00124-00	X	X	X	X
KD11-R CPU (Includes MSV11-CD memory)		X	X	X
BDV11-AA BOOT MODULE			X	X
BA11-NC BOX (115 V)			X	
BA11-ND BOX (230V)				X

MR 1059

Set switches S1 through S7 (S8 is not used) on SW1 as shown in Table 9-2 to select one of the four modes of operation described below:

Mode I The microcode is loaded from the LSI-11 Bus into WCS RAM locations 0 to 1777. The WCS correspondingly responds to microaddresses 2000 to 3777 on the MIB.

Mode II The microcode is loaded from the LSI-11 Bus into WCS RAM locations 0 to 1777. The WCS initially responds to MIB microaddresses 3000 to 3777 from the first 512 words of RAM. If bits <21:18> of the microinstruction are coded to a 7 (octal) then the next microinsturction will be accessed from the second 512 words of RAM. A second 7 (octal) will toggle back to the first 512 words of RAM. This swapping between 512 word blocks for the same microaddress range is called Paging and allows 1024 words of microcode to be implemented when only 512 microaddresses are available.

Mode III This mode is the same as Mode I except that the two blocks of 512 words of RAM have been interchanged on the module. Addressing is identical to Mode I.

Mode IV The microcode is loaded from the LSI-11 Bus into WCS RAM locations 0 to 1777. The WCS correspondingly responds to MIB microaddresses 0 to 1777. This microaddress space is identical to MICROMs 0 and 1, which contain the base PDP-11 microcode for the LSI-11.

9.3.2 Cable Configuration

Configure the cable/plug assembly (Digital part number 17-00124-00) to the shape as shown in Figure 9-1.

9.3.3 WCS Installation

Install the M8018 WCS module into the LSI-11 system as follows: (CAUTION: Great care must be taken when inserting or removing the 40 pin DIP connector on the cable/plug assembly. A grounded work area as well as other normal anti-electrostatic discharge precautions are recommended).

1. Insert the plug, not marked "CPU", of the cabe/plug assembly into the 40 pin socket (J1) on the M8018 WCS module with the chamfer positioned at pin 1 so that the edge of the plug where the cable enters is toward the module handle.

2. Place the M7264-YC CPU module on the top of the M8018 WCS module between the module and the free plug marked "CPU". (See Figure 9-2).

3. Insert the plug marked "CPU" into the empty 40 pin DIP socket (E75) on the M7264-YC CPU module as shown in Figure 9-2. The chamfer should be positioned at pin 1 so that the edge of the plug where the cable enters is toward the handle.

WCS Address Mode Switch Settings

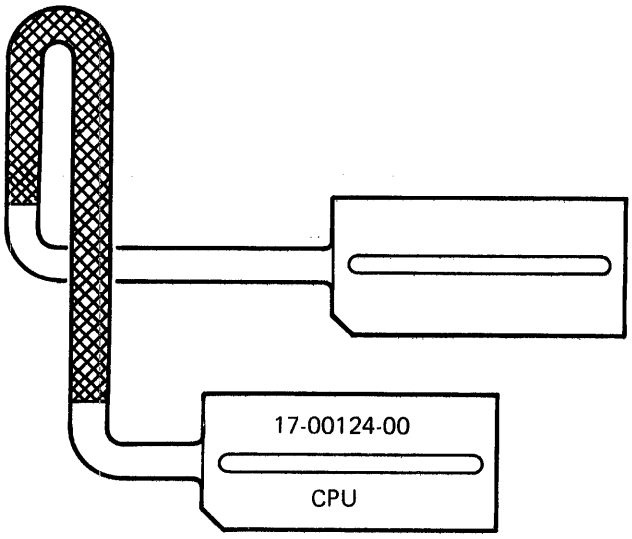
Table 9-2

MODE	SWITCH SW1							
	S1	S2	S3	S4	S5	S6	S7	S8
I	ON	OFF	OFF	ON	OFF	ON	OFF	–
II	OFF	ON	OFF	ON	OFF	OFF	ON	–
III	OFF	OFF	ON	ON	OFF	ON	OFF	–
IV	ON	OFF	OFF	OFF	ON	ON	OFF	–

MR 1060

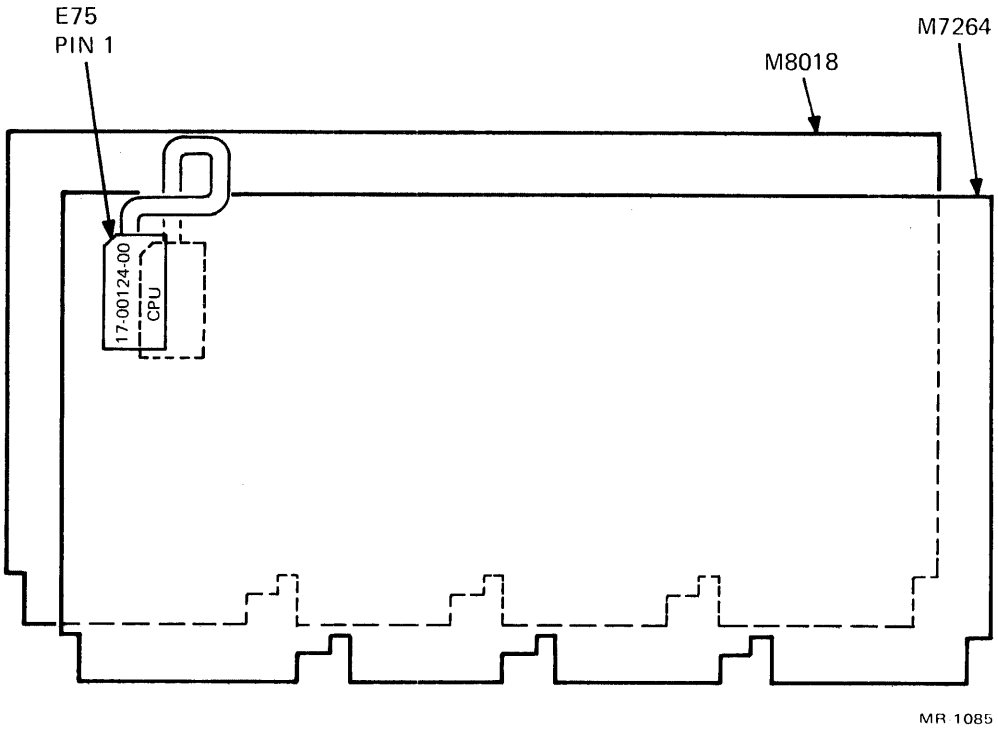
Cable/Plug Assembly Configuration

Figure 9-1



WCS To CPU Installation

Figure 9-2



4. Finally, insert the two modules simulataeously into slots 1 and 2 (M7264-YC in slot 1 and M8018 in slot 2) of the LSI-11 backplane.

9.4 PERFORMANCE CHECKOUT

The KUV11-AA LSI-11 WCS Module (M8018) can be checked for proper performance by running the KUV11-AA diagnostic (CVKUA-A). This is the only diagnostic for the KUV11-AA and it enables the user to check out and trouble shoot the module. The diagnostic is designed to run on an LSI-11 (M7264-YC) with a serial line interface, a console terminal, and 4K (minimum) of memory. It can be run under XXDP, ACT, and APT monitors and is not supervisor compatible. The software switch register (location 176) is used.

NOTE

If the diagnostic is run under XXDP, 6K of memory (minimum) will be needed.

For operating instructions and test details refer to the CVKUA-A diagnostic listing.

CHAPTER 10

MAINTENANCE

10.1 GENERAL

Maintenance for the KUV11-AA LSI-11 WCS Module (M8018) is discussed in this chapter. It is expected that all the material contained in previous chapters should be read and understood before performing any maintenance on the KUV11-AA.

10.2 PREVENTIVE MAINTENANCE

Preventive maintenance for the KUV11-AA consists of periodically checking the WCS cable for kinks, pinches or bends and to ensure that both 40 pin plugs are fully inserted in their sockets.

10.3 CORRECTIVE MAINTENANCE PHILOSOPHY

The KUV11-AA LSI-11 WCS module (M8018) is designed so that module replacement can restore the system to operating status in minimum time. Diagnosing the WCS will consist of first determining that the CPU is properly operating (it may be necessary to test the CPU with the Wcs removed from the system). Next the WCS module would be tested, and if it is at fault, it should be replaced with a spare.

10.4 CORRECTIVE MAINTENANCE

Corrective maintenance is performed on the WCS module by running the CVKUA-A KUV11-AA (LSI-11 WCS) diagnostic (listing part number AC-E102A-MC, paper tape part number AK-E104A-MC). In order to run all the tests most efficiently, a test cable (part number 17-00124-01) and a quad extender module may be used. Error messages will print out when a test sequence fails. These messages will aid in fault detection. The diagnostic performs the following tests:

TEST 1 Register Addressing-Tests that the three device register addresses respond to the LSI-11 Bus.

TEST 2 Bit Test Registers-Tries to set and clear bits in the three device registers. Checks the functionality of

the CSR enable bit (bit<12>).

TEST 3 RAM Test, via LSI-11 Bus-Tests the WCS RAM memory.

TEST 4 Address Mode 1 Test-Verifies WCS operation in Mode 1. Tests the MIB interface (both output and input with respect to the WCS) and the trace logic.

TEST 5 Address Mode 2 Test-Verifies WCS operation in Mode 1. Tests the MIB interface (both output and input with respect to the WCS), the paging logic and the trace logic.

TEST 6 Address Mode 3 Test-Verifies WCS operation in Mode 1. Tests the MIB interface (both output and input with respect to the WCS) and the trace logic.

TEST 7 Address Mode 4 Test-Verifies that the WCS occupies MIB address range 0 - 1777 (octal), which is the same microaddress range as the base LSI-11 microcode. When the enable bit is set (CSR bit<12>=1), the WCS will also respond to the microaddresses in the base microcode range, its output being ORed with the base microinstruction accessed.

Refer to the diagnostic listing for further description and test procedures.

An additional verification of WCS functionallity can be performed when the WCS Software Tools (QJV40-YY) and the EIS and FIS diagnostics (DVKAB-A and DVKAC-A respectively) are available. The test consists of loading the EIS/FIS microcode (included with the Software Tools) into the WCS module, enabling WCS and then executing the two diagnostics.

This test should be performed with the WCS module set to both address mode I and address mode III to give compete coverage to the WCS RAM.

APPENDIX A

INSTRUCTION SUMMARY

OP	NMEMONIC	OPERATION	CYCLES	NB	ZB	C4	C8	N	Z	V	C
0(0)	JMP	LC<--MIR<11:0>	2	-	-	-	-	-	-	-	-
0(8-F)	RFS	LC<--RETURN REGISTER	2	-	-	-	-	-	-	-	-
10	JZBF	If ZB=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
11	JZBT	If ZB=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
12	JC8F	If C8=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
13	JC8T	If C8=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
14	JIF	If ICS=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
15	JIT	If ICS=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
16	JNBF	If NB=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
17	JNBT	If NB=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
18	JZF	If Z=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
19	JZT	If Z=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1A	JCF	If C=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1B	JCT	If C=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1C	JVF	If V=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1D	JVT	If V=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1E	JNF	If N=0,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
1F	JNT	If N=1,LC<--MIR<7:0>	2	-	-	-	-	-	-	-	-
2x	AL	Ra<--Ra+LITERAL	1	*	*	*	*	-	-	-	-
3x	CL	Ra-LITERAL	1	*	*	-	-	-	-	-	-
4x	NL	Ra<--Ra&LITERAL	1	*	*	-	-	-	-	-	-

OP	NMEMONIC	OPERATION	CYCLES	NB	ZB	C4	C8	N	Z	V	C
5x	TL	Ra&LITERAL	1	*	*	-	-	-	-	-	-
6x	LL	Ra<--LITERAL	1	*	*	-	-	-	-	-	-
70	RI	RESET INTERRUPTS	1	-	-	-	-	-	-	-	-
71	SI	SET INTERRUPTS	1	-	-	-	-	-	-	-	-
72	CCF	Ra<--FLAGS	1	-	-	-	-	-	-	-	-
73	LCF	FLAGS<--Ra	1	-	-	-	-	-	-	-	-
74	RTSR	TSR<--0	1	-	-	-	-	-	-	-	-
75	LGL	G<--Ra<2:0>	1	-	-	-	-	-	-	-	-
76	CIB	Ra<--Ra+1	1	*	*	*	*	-	-	-	-
77	CDB	Ra<--Ra-1	1	*	*	*	*	-	-	-	-
80/81	MB/MBF	Ra<--Rb	1	*	*	-	-	*	*	0	-
82/83	MW/MWF	Ra<--Rb	2	*	*	-	-	*	*	0	-
84/85	CMB/CMBF	If C=1,Ra<--Rb	1	(*	*	-	-	*	*	0	-).C
86/87	CMW/CMWF	If C=1,Ra<--Rb	1	(*	*	-	-	*	*	0	-).C
88/89	SLBC/SLBCF	Ra<--2Rb+C	1	*	*	*	*	*	*	*	*
8A/8B	SLWC/SLWCF	Ra<--2Rb+C	2	*	*	*	*	*	*	*	*
8C/8D	SLB/SLBF	Ra<--2Rb	1	*	*	*	*	*	*	*	*
8E/8F	SLW/SLWF	Ra<--2Rb	2	*	*	*	*	*	*	*	*
90/91	ICB1/ICB1F	Ra<--Rb+1	1	*	*	*	*	*	*	*	*
92/93	ICW1/ICW1F	Ra<--Rb+1	2	*	*	*	*	*	*	*	*
94/95	ICB2/ICB2F	Ra<--Rb+2	1	*	*	*	*	*	*	*	*
96/97	ICW2/ICW2F	Ra<--Rb+2	2	*	*	*	*	*	*	*	*
98/99	TCB/TCBF	Ra<--Rb	1	*	*	*	*	*	*	*	*
9A/9B	TCW/TCWF	Ra<--Rb	2	*	*	*	*	*	*	*	*
9C/9D	OCB/OCBF	Ra<--Rb	1	*	*	0	0	*	*	0	1

OP	NMEMONIC	OPERATION	CYCLES	NB	ZB	C4	C8	N	Z	V	C	
9E/9F	OCW/OCWF	Ra<--Rb	2	*	*	0	0	*	*	0	1	
A0/A1	AB/ABF	Ra<--Ra+Rb	1	*	*	*	*	*	*	*	*	
A2/A3	AW/AWF	Ra<--Ra+Rb	2	*	*	*	*	*	*	*	*	
A4/A5	CAB/CABF	If C=1,Ra<--Ra+Rb	1	(*	*	*	*	*	*	*	*)	.C
A6/A7	CAW/CAWF	If C=1,Ra<--Ra+Rb	2	(*	*	*	*	*	*	*	*)	.C
A8/A9	ABC/ABCF	Ra<--Ra+Rb+C	1	*	*	*	*	*	*	*	*	
AA/AB	AWC/AWCF	Ra<--Ra+Rb+C	2	*	*	*	*	*	*	*	*	
AC	CAD	(Ra<--Ra+Rb)<3:0> (Ra<--Ra+Rb)<7:4>	1	*	*	*	*	*	*	*	*	
AE/AF	CAWI/CAWIF	If ICS=1,Ra<--Ra+Rb	2	(*	*	*	*	*	*	*	*)	.ICS
B0/B1	SB/SBF	Ra<--Ra-Rb	1	*	*	*	*	*	*	*	*	
B2/B3	SW/SWF	Ra<--Ra-Rb	2	*	*	*	*	*	*	*	*	
B4/B5	CB/CBF	Ra-Rb	1	*	*	*	*	*	*	*	*	
B6/B7	CW/CWF	Ra-Rb	2	*	*	*	*	*	*	*	*	
B8/B9	SBC/SBCF	Ra<--Ra-Rb-C	1	*	*	*	*	*	*	*	*	
BA/BB	SWC/SWCF	Ra<--Ra-Rb-C	2	*	*	*	*	*	*	*	*	
BC/BD	DB1/DB1F	Ra<--Rb-1	1	*	*	*	*	*	*	*	*	
BE/BF	DW1/DW1F	Ra<--Rb-1	2	*	*	*	*	*	*	*	*	
C0/C1	NB/NBF	Ra<--RaRb	1	*	*	-	-	*	*	0	-	
C2/C3	NW/NWF	Ra<--RaRb	2	*	*	-	-	*	*	0	-	
C4/C5	TB/TBF	Ra<--Rb	1	*	*	-	-	*	*	0	-	
C6/C7	TW/TWF	Ra<--Rb	2	*	*	-	-	*	*	0	-	
C8/C9	ORB/ORBF	Ra<--Ra!Rb	1	*	*	-	-	*	*	0	-	
CA/CB	ORW/ORWF	Ra<--Ra!Rb	2	*	*	-	-	*	*	0	-	
CC/CD	XB/XBF	Ra<--Ra!Rb	1	*	*	-	-	*	*	0	-	
CE/CF	XW/XWF	Ra<--Ra!Rb	2	*	*	-	-	*	*	0	-	

INSTRUCTION SUMMARY

OP	NMEMONIC	OPERATION	CYCLES	NB	ZB	C4	C8	N	Z	V	C
D0/D1	NCB/NCBF	Ra<--Ra+~Rb	1	*	*	-	-	*	*	0	-
D2/D3	NCW/NCWF	Ra<--Ra+~Rb	2	*	*	-	-	*	*	0	-
D8/D9	SRBC/SRBCF	Ra<--C:Rb/2	1	*	*	0	*	*	*	0	*
DA/DB	SRWC/SRWCF	Ra<--C:Rb/2	2	*	*	0	*	*	*	0	*
DC/DD	SRB/SRBF	Ra<--Rb/2	1	*	*	0	*	*	*	0	*
DE/DF	SRW/SRWF	Ra<--Rb/2	2	*	*	0	*	*	*	0	*
E0/E1	IB/IBF	Ra<--DAL	1	*	*	-	-	*	*	0	-
E2/E3	IW/IWF	Ra<--DAL	2	*	*	-	-	*	*	0	-
E4/E5	ISB/ISBF	Ra<--DAL	1	*	*	-	-	*	*	0	-
E6/E7	ISW/ISWF	Ra<--DAL	2	*	*	-	-	*	*	0	-
EC/	MI	MIB!Rb:Ra	1	-	-	-	-	-	-	-	-
EE	LTR	TR<--Rb:Ra G<--(Rb)8:(Ra)7-6	2	-	-	-	-	-	-	-	-
F0	RIB1	M<--Rb:Ra Ra<--Ra+1	1	*	*	*	*	-	-	-	-
F1	WIB1	M<--Rb:Ra Ra<--Ra+1	1	*	*	*	*	-	-	-	-
F2	RIW1	M<--Rb:Ra Ra<--Ra+1	2	*	*	*	*	-	-	-	-
F3	WIW1	M<--Rb:Ra Ra<--Ra+1	2	*	*	*	*	-	-	-	-
F4	RIB2	M<--Rb:Ra Ra<--Ra+2	1	*	*	*	*	-	-	-	-
F5	WIB2	M<--Rb:Ra Ra<--Ra+2	1	*	*	*	*	-	-	-	-
F6	RIW2	M<--Rb:Ra Ra<--Ra+2	2	*	*	*	*	-	-	-	-
F7	WIW2	M<--Rb:Ra Ra<--Ra+2	2	*	*	*	*	-	-	-	-
F8	R	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
F9	W	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
FA	RA	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
FB	WA	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
FC	OB	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
FD	OW	M<--Rb:Ra	1	-	-	-	-	-	-	-	-

OP	NMEMONIC	OPERATION	CYCLES								
FE	OS	M<--Rb:Ra	1	-	-	-	-	-	-	-	-
FF	NOP	--	1	-	-	-	-	-	-	-	-

NOTE: When two op codes refer to a specific mnemonic the even op code does not affect the condition codes whereas the odd op code does affect the condition codes.

NOTE: M refers to the LSI-11 Bus.

LSI-11 WCS
USER'S GUIDE
EK-KUV11-TM-001

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

What features are most useful? _____

What faults or errors have you found in the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, Ma 01532
Attention: Communications Services (NR2/M15)
Customer Services Section

Order No. EK-KUV11-TM-001

— — — — — — — — — — — — Fold Here — — — — — — — — — — — —

— — — — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
Maynard, Massachusetts 01754

digital

digital equipment corporation

Printed in U.S.A.