DIGITAL SCIENTIFIC META 4™ SERIES 16 COMPUTER SYSTEM

REFERENCE MANUAL

Publication Number 7032MO

(Revision B)

(Supersedes Publication Number 7006MO)

Digital Scientific

META 4™

May 1971

DIGITAL SCIENTIFIC CORPORATION
11455 Sorrento Valley Road
San Diego, California 92121

RECORD OF REVISIONS

Title: Digital Scientific META 4 Series 16 Computer System, Reference Manual

Publication Number: 7032MO (supersedes Publication Number 7006MO in its entirety)

DATE	ISSUE	
10/22/70	Original	
1/4/71	Revision A: List of Changed Pages:	
	Delete	Add
	page 1-5 (1-6)*	page 1-5 (1-6)
	pages 1-9 through 1-11 (1-12)	pages 1-9 through 1-11 (1-12)
	page 1-19 (blank)	page 1-19 (blank)
	pages 2-3 and 2-4	pages 2-3 and 2-4
	page (2-5) 2-6	page (2-5) 2-6
	page (2-9) 2-10	page (2-9) 2-10
	page 2-13 (2-14)	page 2-13 (2-14)
	page (2-15) 2-16	page (2-15) 2-16
	page 2-19 (2-20)	page 2-19 (2-20)
	page (2-21) 2-22	page (2-21) 2-22
	page 2-27 (2-28), page 2-29 (2-30)	page 2-27 (2-28), page 2-29 (2-30)
	page (2-43) 2-44	page (2-43) 2-44
	page A-3 (blank)	page A-3 (blank)
	page B-1 (B-2)	page B-1 (B-2)

* Pages indicated parenthetically are unchanged, but are back-up pages to the changed text. Note: Changes are indicated by bars in the margin opposite the affected text.

Address comments to:

DIGITAL SCIENTIFIC CORPORATION
11455 Sorrento Valley Road
San Diego, California 92121

Phone: (714) 543-6050
TWX: 910-322-1136

RECORD OF REVISIONS
(Continued)

Title: Digital Scientific META 4 Series 16 Computer System, Reference Manual

Publication Number: 7032MO

DATE	ISSUE
5/3/71	Revision B: List of Changed Pages: Delete / Add pages iii and iiia page vi / page vi page 1-3 / page 1-3 page 1-9 / page 1-9 page 1-14 and 1-15 / page 1-14 and 1-15 page 1-17 / page 1-17 page 2-4 / page 2-4 page 2-19 / page 2-19 page 2-44 / page 2-44 page A-2 / page A-2 page (Title page E) / page (Title page E) page F-2 / page F-2 page G-1 / page G-1 page H-2 / page H-2 NOTE: Changes are indicated by bars in the margin opposite the affected text.

Address comments to:

DIGITAL SCIENTIFIC CORPORATION
11455 Sorrento Valley Road
San Diego, California 92121

Phone: 714/453-6050
TWX: 910 322 1136

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

FOREWORD

This manual is periodically updated to reflect the current state-of-the-art of the Digital Scientific META 4 Computer System. As an addendum to this Revision B, the following statement is offered:

> System timing cycles stated in this Reference Manual are valid on META 4 Systems employing one full central processor unit (CPU) logic rack, one full core memory logic rack, and one full input/output (I/O) logic rack.
>
> A full CPU logic rack consists of a full complement of registers (32), and a maximum of 4 read-only memories (ROM's). A full core memory logic rack consists of a maximum of 65K words of core. A full I/O logic rack consists of as many as 11 wire-wrap boards or 22 printed-circuit boards.
>
> This is the configuration under which META 4 Computer System timing is valid. Deviations in configuration from that above can result in timing differences from those specified in this manual.

1. DIGITAL SCIENTIFIC META 4 SERIES 16 COMPUTER SYSTEM

INTRODUCTION[1]

META 4 System Concepts

•Organization of Digital Scientific's META 4 Computer provides "general-purpose applicability" not only in a main memory instruction set, but also in the underlying, microprogram instruction set and in its flexible hardware organization.

Contrary to the traditional computer organization, which bars the system designer from using registers and data paths other than as defined by a core memory instruction set, each META 4 System designer can determine the number and functional assignment of general-purpose registers for data handling, external input/output, core memory input/output, accumulators or indexing, and the core memory instruction set to use.

The object of META 4 microprogramming capability is to allow the system designer to exercise all of the possible hardware interconnections as directly as possible without intervening constraints. The system designer can be both a programmer and a logic designer and can manipulate not only algorithms, but architecture.

The META 4 Microprogrammed Processor can be described as a "computer within a computer, "where inner computer sequences (the microprogram) emulate the machine language instructions for the outer computer and also can execute special sequences not necessarily related to ordinary outer-machine instructions.

Conventional computer system organization limits the permissible interconnections among functional elements. Since machine language formats and logical organization of conventional computers are closely related, only one machine language instruction set can generally be executed efficiently and the full potential of possible interconnections among the functional elements cannot be realized.

---

[1] Digital Scientific Corporation is indebted to Dr. Robert Rosin for permission to include some of his concepts on emulation in this introduction.

The META 4 microprogrammed computer organization offers the system designer an opportunity to optimize internal hardware facilities to a particular set of requirements.

Organization of the META 4 system is independent of the machine language instruction set in main memory and permits flexibility in specifying interconnections for functional elements such as arithmetic and Boolean operational units, registers, memory, and input/output devices.

For example, core memory instruction execution in every computer involves a sequence of steps such as:

- Place program location counter in memory address register.
- Fetch instruction from main memory and place in memory data register.
- Increment program location counter to prepare for next instruction.
- Move current instruction to instruction register.
- Decode current instruction and addressing mode.
- Calculate operand address.
- Place operand address in memory address register.
- Fetch operand from main memory and place in memory register.
- Perform operation.

In a conventional computer, specially wired circuits control each sequence. The main memory instruction set is fixed and can be changed only by rewiring the computer.

In the META 4 Computer, a high-speed read only control memory (ROM) replaces the specially wired circuitry. Control memory instructions (the microprogram) specify interconnections of functional elements. The control memory contents can be altered easily by the system designer, thus permitting great flexibility in specifying interconnections and allowing almost any desired main memory instruction set.

A META 4 microprogram emulator for another computer differs from a conventional computer program interpreter in several features, as described below.

- Microprograms are stored in a control memory which is distinct from the main memory of the emulated machine and is read-only memory (ROM) for the purpose of optimizing speed.

- Facilities to improve implementation and operation of emulators can be implemented at speeds comparable to those available from hardware. Examples are floating-point arithmetic, character code conversion, and control panel operation.

Digital Scientific Corporation's unique Read-Only Memory (ROM) controlled instruction cycle times average 90 nanoseconds, enabling 10 or more microinstructions to be executed during each 900-nanosecond core memory cycle.

META 4 ROM microprograms (firmware) allow both emulation of ordinary core memory instructions and execution of special algorithms to be executed much more rapidly than in other "general-purpose" devices. These factors - the computer organization and the high-speed ROM firmware - mean that applications which would normally require special hardware are standard capabilities for a META 4 Computer. For example:

- Instruction Set Emulator for Other Computers (with improved performance): An IBM 1130/1800 can be emulated completely with higher performance and can also be improved upon substantially by adding floating-point instructions and register-to-register instructions.

- Channel Interface or Peripheral Equipment Controller for Other Computers: A disc controller with code and format conversion capabilities can be microprogrammed to operate at high speeds to provide economical standard interfacing to a variety of other computers; an IBM 2314 disc controller can be emulated.

- Communications Line Controller, Buffer, Editor, and Preprocessor: Serial-to-parallel conversion and data editing for multiple nonsynchronous or synchronous lines can be done at high speed to relieve a data processing system of a substantial overhead load; an IBM 2703

communications controller can be emulated with features not available in the original.

- Digital Algorithm Processor: Convolution, fast Fourier transformation, correlation, high-level language compilation, or queue optimization algorithms can be executed at high speed, among multiple registers, using core memory for data only, not for program execution.

PROCESSOR ORGANIZATION

- The processing unit (see Figure 1-1, 1-2, and 1-3) consists of data registers, data processing logic, sequence control and the read-only memory (ROM), input/output facilities, core memory, and integrated scratch-pad memory.

Data Registers

- Data Registers are 16-bit, integrated-circuit registers. Up to 31 directly addressable registers may be installed. During certain operations, data from the Read-Only Memory (ROM) may be used in place of register data.

In Figure 1-1, registers for addresses 4 through 31 are optional. The requirement for and the choice of a particular type of register depend upon the user's system requirements for accumulator and scratch-pad registers, core memory registers, and input/output registers.

Data Processing Logic

- Data Processing Logic consists primarily of an arithmetic/Boolean unit, which processes data received via the A-bus and the B-bus; followed by a Skew unit, which transmits data to a destination register via the D-bus. The arithmetic unit is a 16-bit, high-speed parallel adder. Carry-in controls, together with overflow and carry-out condition register bits, allow multiple precision operation. The Boolean functions comprise the logical connectives AND, OR, or Exclusive OR. The skew unit manipulates the result of either an arithmetic or a Boolean operation.

Sequence Control

- Sequence Control for the processor is a program stored in high-speed, Read-Only Memory (ROM) and coded in a manner similar to Assembly language instructions for a conventional (hardware-sequenced) computer. Addresses in ROM instructions and in Register 2 (the Link register) are used by the branch-control unit to shift control between various sequences as the result of testing operations.

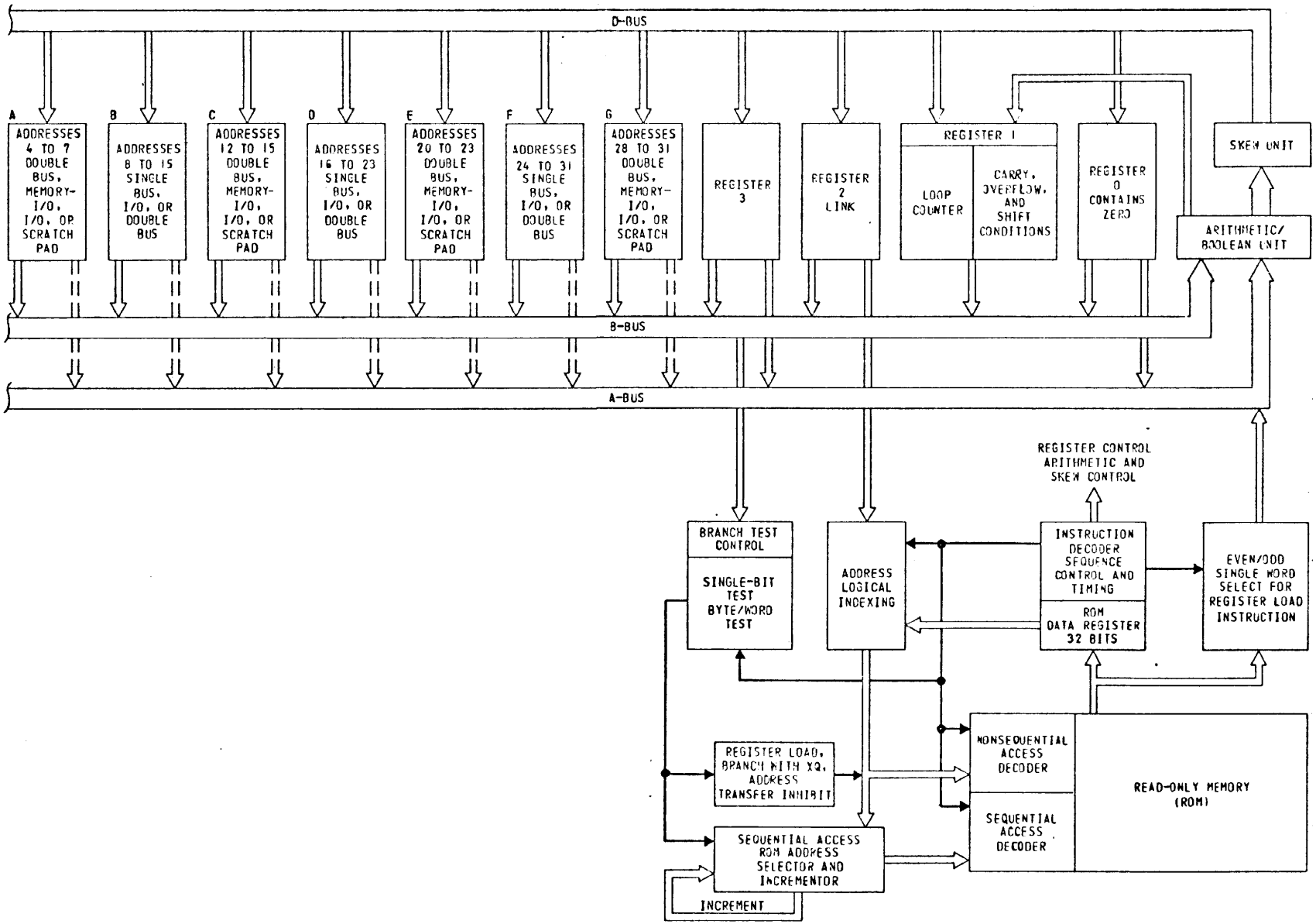


FIGURE 1-1. META 4 COMPUTER SYSTEM, BLOCK DIAGRAM OF CENTRAL PROCESSOR UNIT

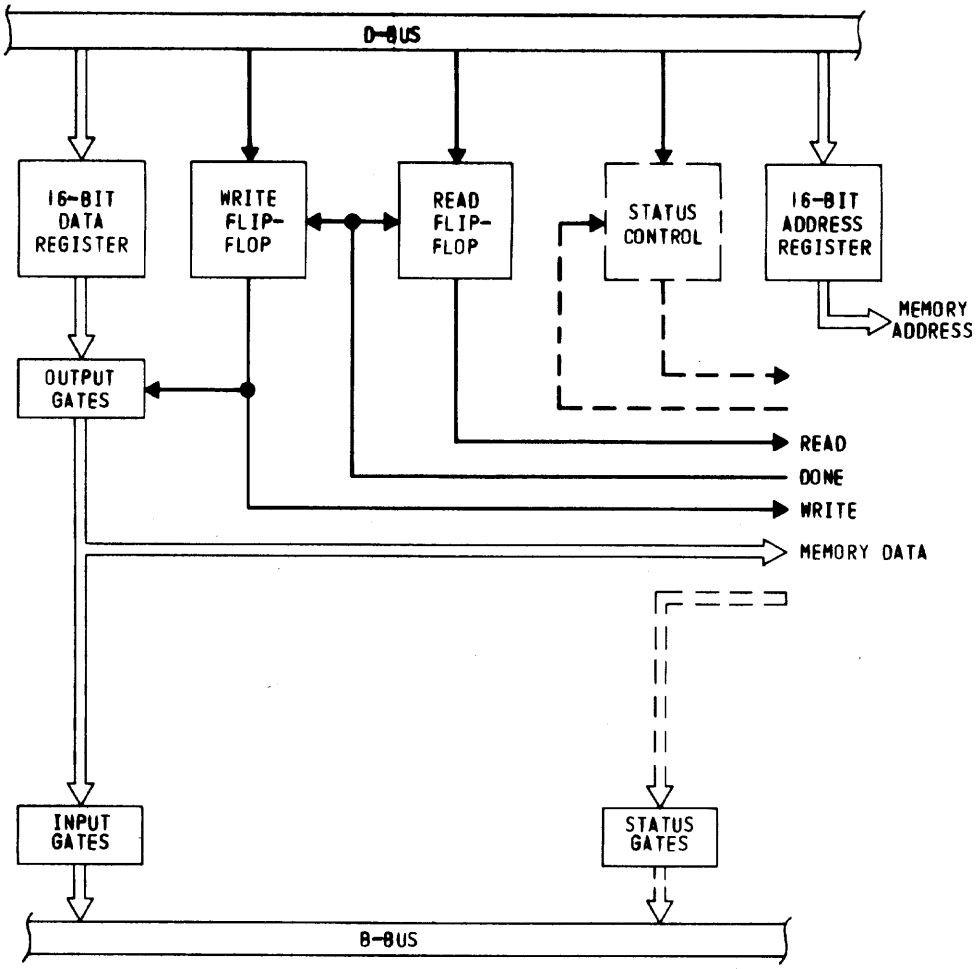


FIGURE 1-2. MEMORY INTERFACE REGISTER

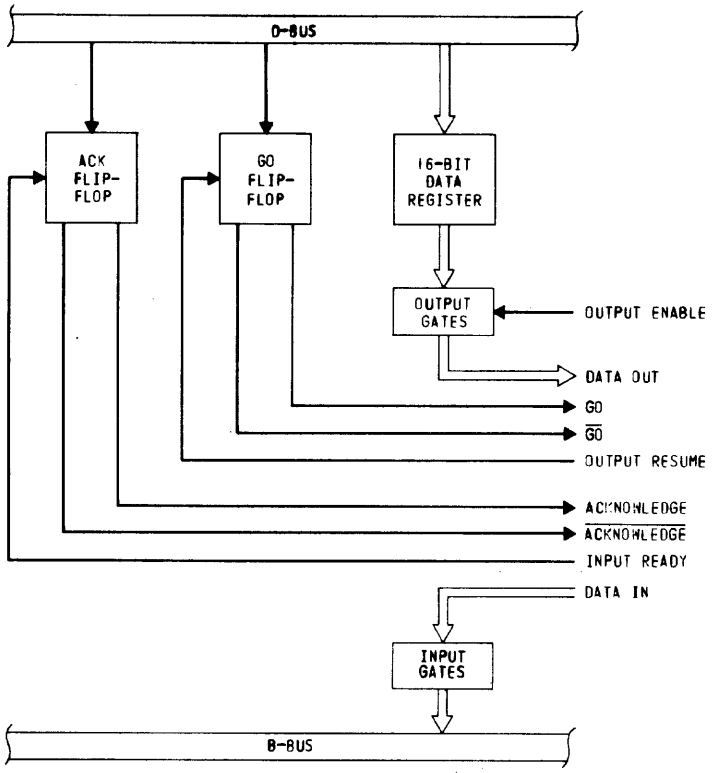


FIGURE 1-3. I/O INTERFACE REGISTER

Any single bit of any addressable register may be tested for zero or nonzero, 8-bit or 16-bit fields may be tested for zero or nonzero, and a self-decrementing register (Register 1) may be tested for zero concurrently with operations of functional units.

Input/Output

•Input/Output Facilities are implemented at three levels:

1. Direct cable connections to special types of directly addressable registers. The sequence control program may communicate with the system peripheral equipment through these registers.

2. Chassis accepting standard controller for various peripheral equipment on a plug-in basis. No field wiring changes are required to add or delete peripheral equipment. Peripheral equipment controllers operate on a party-line I/O bus or directly to memory, as applicable.

3. Direct access to core independently of the adapter chassis.

Core Memory

•Core Memory is operated by the control program through special registers and controls. Four standard memory ports allow multiple processors or special equipment to share multiple banks of memory. Each bank of core memory is an independently operable unit. The processor can use additional memory registers or interleaving to overlap accesses to several banks.

Scratch-Pad Memory

•Integrated Circuit Scratch-Pad Memory is operated by the control program internal control and data through special registers incorporated into the scratch-pad controls

PROCESSOR HARDWARE DESCRIPTIONS

•The complete processor, including control memory, mounts in a 19-inch-wide rack housing and requires a 14-inch height for the logic and control memory chassis, a 14-inch height for memory banks, and a 14-inch height for the input/output adaptor chassis. Power supplies are normally mounted on the rear rails of a cabinet behind the processor chassis. (See Figure 1-2.) Air movement is provided by for assemblies which require additional space on the rack.

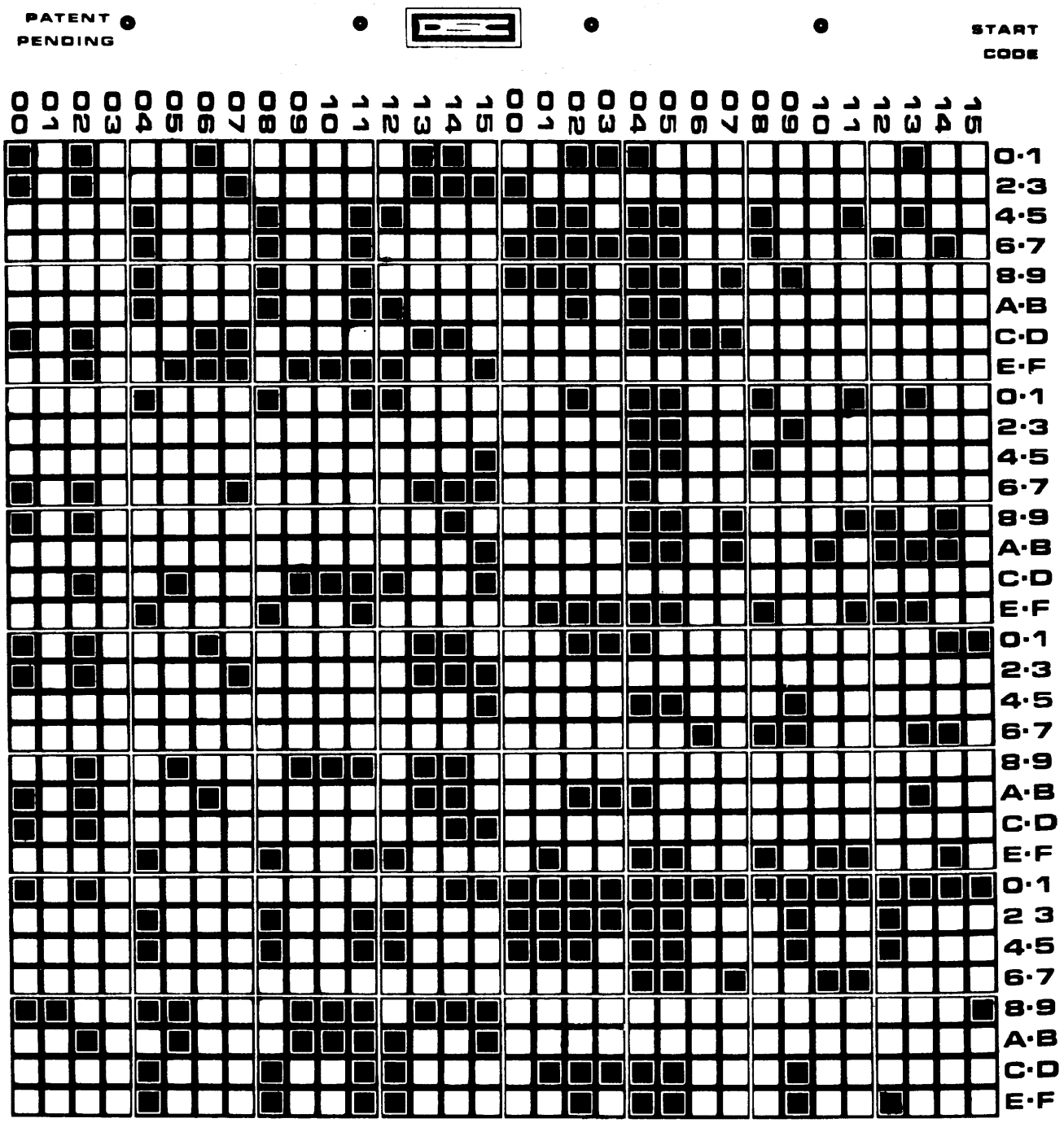


FIGURE 1-4. DIGITAL SCIENTIFIC META 4 SERIES 16 PROCESSOR, HARDWARE ORGANIZATION

Processor logic uses high-speed, emitter-coupled, integrated circuits for reliable operation at high speed. Major operation cycle time averages 90 nanoseconds including Read-Only Memory (ROM) and data source register accesses, arithmetic and/or logical shift processing, and storage in a destination register.

Input/output logic uses a mixture of DTL and TTL integrated circuits.

Registers and Scratch-Pad Memory

•Optional assemblies for directly addressed registers are available in several versions. Optional assemblies for integrated circuit scratch-pad memory are also available. Register types differ in internal bus connections, external I/O connections, and associated control functions. Register assemblies differ in the number and types of registers.

TABLE 2-1. META 4 PROCESSOR REGISTER VERSIONS

BOARD TYPE	BOARD DESCRIPTION	DATA BUS CONNECTIONS
Double-Bus Accumulators	Four 16-Bit Data Registers	A, B, and D
Single-Bus Accumulators	Eight 16-Bit Data Registers	B and D
I/O	Four 16-Bit Input Gates and Four 16-Bit Output Latches	B and D
Memory I/O	One Memory Address and Control Register, One 16-Bit Memory Input Gate and One 16-Bit Memory Output Latch; Two 16-Bit Input Gates and Two 16 Bit Output Latches	B and D
Scratch-Pad Memory	Sixty-four 16-Bit Data Registers	B and D (both indirectly)

Registers may be data sources on the A-bus and B-bus and data destinations on the D-bus. Registers without physical connection to the A-bus will function as if the contents were zero for the A-bus only.

Register addresses 0, 1, 2, and 3 are assigned to the basic machine structure (see Figure 1-1). All other addresses (4 through 31) are available for general use and are assigned in groups of four registers to seven connectors. Addresses 4, 5, 6, and 7 are normally assigned to core memory and general I/O functions. I/O, Memory-I/O, and Single-bus registers can be installed only in the positions indicated.

The control logic of the META 4 has jumpers installed to control timing for any connector with a Memory-I/O register installed.

Dedicated Registers

•The dedicated META 4 register address functions and connections are indicated below:

•Register 0 — Zero register: contains zero for operand use and serves as a dummy destination.

•Register 1 — Condition/Counter register: bits 8 through 15 contain a self-decrementing counter which may be initialized from the D-bus and decremented and tested by instruction control bits. Bits 0, 1, and 2 represent carry-out, overflow, and shifter conditions and are not controllable from the D-bus. Bits 3 through 7 are fixed at zero. All register bits can be gated to the B-bus for operand use or program testing with the limitation that if Register 1 is specified as both the B-bus source and the D-bus destination of a single instruction the counter contents are indeterminate.

•Register 2 — Link register: serves as an address source for the ROM address selector during specific instructions. The Link register may be set from the D-bus and gated to the B-bus as required and may serve as a single bus accumulator if not required for ROM addressing.

•Register 3 — General-purpose, double-bus accumulator: has no special properties.

Data Processing Logic

•The high-speed, 16-bit parallel adder operates in two's complement mode with carry input under program control. Carry-out and overflow automatically force the appropriate condition register bits and may be tested in the Condition/Counter register. Carry input during an instruction may be either inhibited, selected to be the previous carry output, or forced unconditionally. The ability to select a previous carry-out as a carry input simplifies multiple precision operation. The ability to force a carry input facilitates two's

complement subtraction operations using logical complementing of operands rather than arithmetic complementing. If one's complement arithmetic operation is required, the processor program may use two-step additions in which the second step provides the end-around-carry characteristics of one's complement operations.

Two special addition operations expedite multiply and divide operations:

- Multiply step is addition which is completed only if the shift condition was previously true.
- Divide step is a trial addition where a negative sum inhibits changing the destination register.

Boolean Function Unit

- The Boolean unit provides the logical connectives AND, OR, or Exclusive OR of the A-bus and B-bus sources. Since Register 0 (containing zero) may be used as one of the operands, the Boolean unit may be used to zero registers using the AND function and to copy data using the OR function. An Exclusive OR using a data field from the ROM with all 16 bits true is used to complement data.

Skew Function Unit

- The Skew unit provides bit manipulations on the output of the Arithmetic/Boolean unit. Both carry-out to the shifter link and carry-in from the shifter link for shift operations may be selected independently. The shifter link status may be tested in the Condition/Counter register and represents the data spill from the most recent skew operation having shift out enabled.

Skew operations comprise:

- One-place left or right shift
- Eight-place left or right shift/rotate
- Sign extend (copy bit 8 into bits 0 through 7)
- Scale (one-place end-off right shift with arithmetic carry entering at left)
- No shift.

Sequence Controls and Read-Only Memory (ROM)

•The ROM comprises a word drive and bit sense structure which is loaded with firmware contents by sliding in storage boards that have removable adhesive-bonded metallic "bit-patch" patterns, representing bit positions in sequential instructions. A bit patch is binary "1" and the absence of a bit-patch indicates binary "0." See Figure 1-5.

ROM Organization

•Contents of the ROM can be readily modified or replaced in the field by either Digital Scientific Corporation or user personnel at the bit or board level. Addresses of instruction words must be even. Logically indexed references to data words may use either even or odd addresses. Up to 4096 single words may be installed in multiples of 1024 words. Each reference to the ROM calls up a double word so that access time is identical for single words and double words.

ROM instructions are executed in sequence unless a Branch causes transfer to another sequence. Branches occur in one of three ways: if the J modifier is specified during an RR format instruction, the next instruction is unconditionally taken from the address in the Link register (Register 2); if J and D modifiers are specified and the counter section of the Condition/Counter register (Register 1) does not decrement to zero during an RR format instruction when tested, the next instruction is taken from the address in the Link register; if a Branch instruction to test various data or machine conditions is successful, the next instruction is taken from the data field of the instruction and logical indexing by the Link register is selectable.

A 4-bit field in the branch instruction "points" at any single bit of any addressable register. Branching may be selected for the true or false state of the specified bit, allowing tests for data sign, arithmetic carry/overflow, shift carry, or any other single bit condition. Branching on zero or nonzero half words or single words is selected by a modified branch instruction.

The system is initialized by an externally applied signal which clears the I/O register controls and the ROM register. Execution of the instruction at ROM address 000_{16} (normally a Branch) can lead to a firmware routine that initializes other parts of the system such as internal working registers.

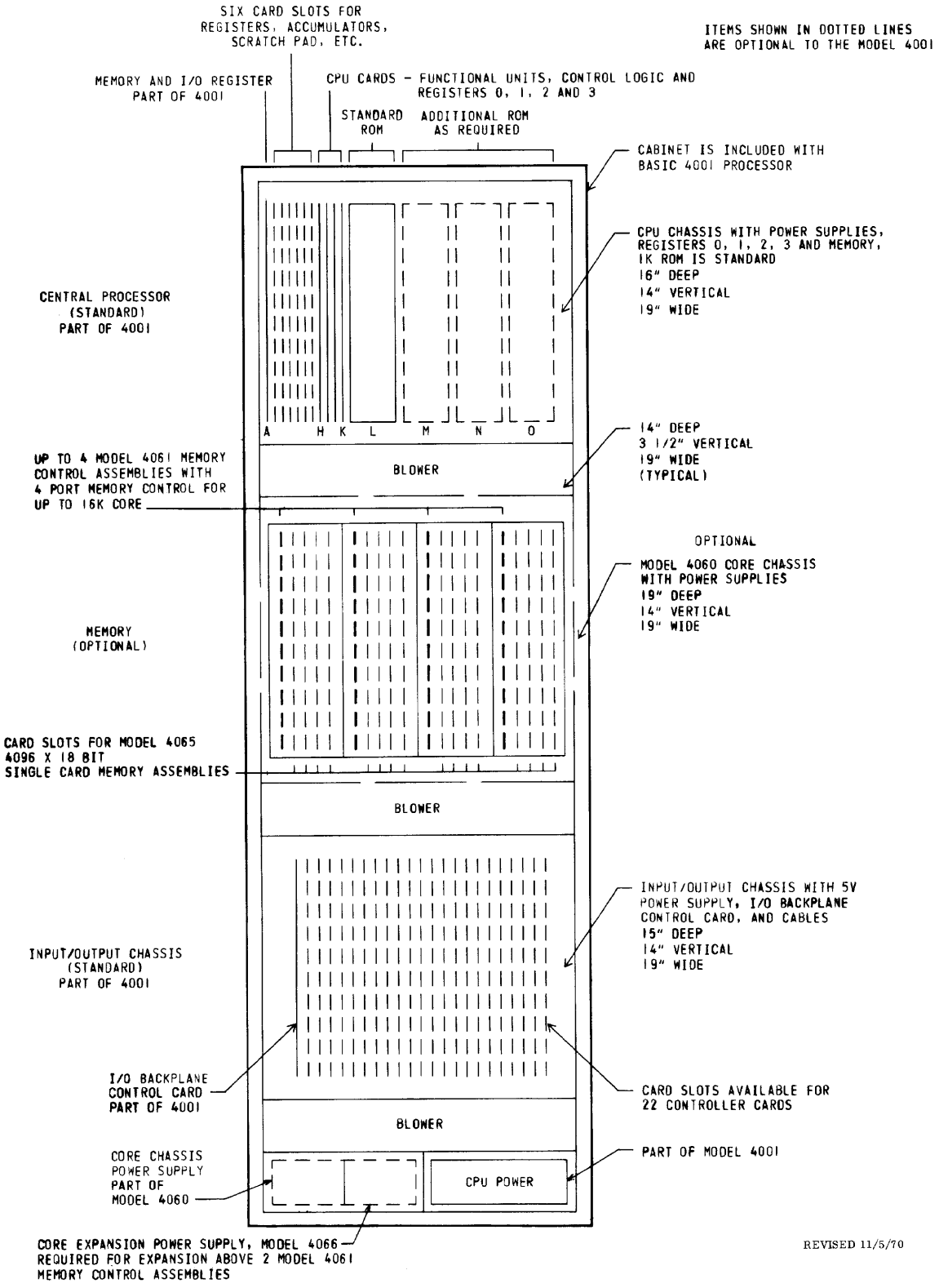


FIGURE 1-5. DIGITAL SCIENTIFIC META 4 COMPUTER SYSTEM ROM BOARD, TYPICAL PATTERN

META 4 System I/O Registers

•The Digital Scientific META 4 Central Processor Unit (CPU) transfers data between internal buses and external devices using I/O or Memory-I/O interface registers. The design of the CPU restricts use of Memory-I/O boards to those sockets for register groups 04 through 07, 0C through 0F, 14 through 17, and 1C through 1F. The corresponding CPU card sockets are labeled A, C, E, and G.

Each I/O board provides four independent front connectors for external I/O cables. Each Memory-I/O board provides two independent front connectors for external I/O cables, one connector for a memory cable, and one connector for memory status, if the Memory Address Permuter (MAP) option is implemented. The I/O board connectors (ports) are arranged from top to bottom in order of increasing register address. The Memory-I/O board connectors are arranged from top to bottom in order of increasing register address and are: memory status, memory address and data, and two I/O registers.

Stopping the META 4 Clock

•Understanding the logical and timing requirements for I/O registers requires a description of the manner in which the META 4 Processor's clock system operates:

- •The META 4 clock rate is not constant, but each command operates in one clock cycle, except for Register Load instructions which require two clock cycles. After each cycle, the machine may be stopped with the next command already present in the ROM data register and partially executed in the sense that the internal address and data bus paths are enabled. In other words, the D-bus data is available but is not yet transferred; and the ROM address for obtaining the next command is selected, but the ROM data is not yet read.

- •The clock restart cycle causes the D-bus data to be transferred to the destination register and the ROM to be pulsed in order to load the next command into the Command register.

The META 4 clock may be triggered at intervals of less than 85 nanoseconds (depending on configuration and bus loading) if the ROM addresses occur in increasing sequence unless:

- •A command implies that the next ROM address might be out of sequence (e.g., Branch, Register Load, or Jump).

- PZ modifier is true.

- A memory register is addressed on the D-bus (CPU control jumpers define memory register positions).

In such cases, the start of the next clock cycle is delayed for 30 nanoseconds (120-nanosecond total cycle) either to allow for ROM address selection settling or to allow time for the I/O port pause condition to be recognized.

Once an I/O or Memory-I/O register pause condition is recognized, the start of the next clock cycle can be delayed indefinitely beyond the 30-nanosecond minimum delay. That is, a META 4 machine cycle can be externally controlled to be any time longer than the minimums specified here.

I/O Transfer

- Each I/O register port provides 16 bits of data output from the CPU and 16 bits input to the CPU. The data outputs are buffered by a flip-flop register, which is addressed and loaded from the D-bus. The data inputs are gated directly (no flip-flops) onto the B-bus.

Data outputs change state only when the clock cycle pertaining to the next command, which may have been delayed by an I/O port pause, is initiated and therefore finishes execution of the previous command. Input data pertaining to a command which has been delayed by an I/O pause is, however, gated to the B-bus. Therefore, the destination data during the paused condition continuously follows input data. When the clock system is restarted, the data is transferred to the designated D-bus register.

The separation of data input and output paths implies that data loaded into an output register cannot be read by a B-bus input command unless an external connection is made to the register port.

I/O Interlocking

- The META 4's clock system pause logic uses I/O control flip-flops. Each input and each output register path has one such control flip-flop. The control flip-flop for an input register is known as the Acknowledge (ACK) flip-flop. The control flip-flop for an output register is known as the GO flip-flop.

A control flip-flop is set only when the next clock cycle (which finishes execution of a command) is initiated and the I/O bit of the command is true, that is, a control flip-flop can be set only at the same time that data is loaded into destination register.

A control flip-flop can be reset by:

- The CLEAR switch on the microprogrammer's panel
- An automatic clear on initial power up
- By an external signal on the I/O port.

These external signals are known as the Input-Ready signal for an ACK flip-flop and the Output-Resume signal for a GO flip-flop.

The clock system will halt as long as a control flip-flop is set or the reset pulse has not terminated if:

- The pause bit of the command is true, and
- The register B-bus or D-bus address in the command corresponds with the associated register. The B-address applies to an ACK flip-flop and the D-address applies to the GO flip-flop.

When both PZ and IO modifiers are specified in a command:

- PZ controls whether or not the command will pause because of the current state of a flip-flop.
- IO controls the subsequent state of the control flip-flop.

That is, PZ applies to conditions prior to initiation of a META 4 clock cycle, and IO applies to conditions after a META 4 clock cycle.

A signal, called Output-Enable, enables gates between the output data flip-flops and the output data lines. This feature allows wired-OR connections from more than one output register to a command cable.

Detailed Timing Considerations for I/O Interface Register

•Restoring a Ready condition initiates a clock cycle if the clock pause resulted from that control flip-flop PZ test. The cycle completes (including control flip-flop setting) and the next instruction starts 75 nanoseconds later.

Input data should settle before Input-Ready becomes active and the data should be maintained until Acknowledge becomes active. Output may occur as soon as 75 nanoseconds after Output-Ready is active. However, the next data output could be considerably later either because the firmware may be executed in single-step mode or because the program has not yet arrived at a ROM command which changes output data. Similar remarks apply to the time duration between Input-Ready and the Acknowledge signal.

Refer to Appendix H for signal pin assignments and interface information.

Core Memory Read/Write Transmission and Control

•Core memory read/write transmission uses register reference instructions in a manner similar to ordinary input/output register reference instructions. The two control instruction bits are interpreted as Read and Write rather than Pause and I/O control. The program sequence Pause function is implied as active when addressing any memory register.

The core memory is a coincident current system with a 900-nanosecond full cycle (read/write). Each completely independent bank has four independent access ports. Port priority may be assigned at the discretion of the user and may differ between banks. Memory is protected against power failure.

One 16-bit output register is assigned as the core memory address register and a second 16-bit output register is assigned as the core memory data output register. One of the corresponding 16-bit input register addresses is assigned to the core memory data input gate and the second corresponding 16-bit input register address is assigned to the input path for memory parity, protect status condition, and memory control signals if the MAP option is implemented. Input/output pairs are not externally connected except for memory data. Only one standard cable is required to connect the core memory with the four register paths. The memory data lines are bidirectional and are shared for input and output.

Standard memory feature:

- One 16-bit odd parity bit.
- One 16-bit odd parity and one protect bit with automatic abort of Write instructions when the memory cell is protected and the Write control does not indicate a protected write status. Error conditions must be transmitted to the I/O system and from there to the processor.

PERIPHERAL EQUIPMENT

•Peripheral equipment is operated either by dedicated registers or by a multiplexed signal bus using one pair of standard I/O registers. One output register is used for addressing and a second output register is used for control and data output. One of the corresponding input register addresses is used for data input and the second corresponding

input register address is used for miscellaneous status and data bit inputs. Two standard cables are required to connect the two register paths with the chassis of the peripheral devices.

Standard peripheral devices are listed below:

- Keyboard/Printer
- IBM 1130 Control Panel
- IBM 1130 SAC channel
- IBM 1800 Computer Data Channel
- IBM 1800 Control Panel
- 300-character-per-second Paper Tape Reader
- 50-character-per-second Paper Tape Punch
- 300-card-per-minute Hollerith Card Reader
- 200-card-per-minute Hollerith Card Punch and Reader combination
- Moveable Head Disc with Removeable Single Disc Pack
- 300-line-per-minute Line Printer
- 600-line-per-minute Line Printer
- Magnetic Tape Transports (1 x 2 controller, 7- or 9-track)
- Digital Input/Output Interfaces
- 300-step-per-second Incremental Plotter
- Teletype Line Adapters
- High-Speed Communications Line Adapters
- Real-Time Clock
- Stall Alarm/Timer

2. READ-ONLY MEMORY(ROM) INSTRUCTIONS AND INSTRUCTION MODIFIERS

GENERAL DESCRIPTION

Instructions

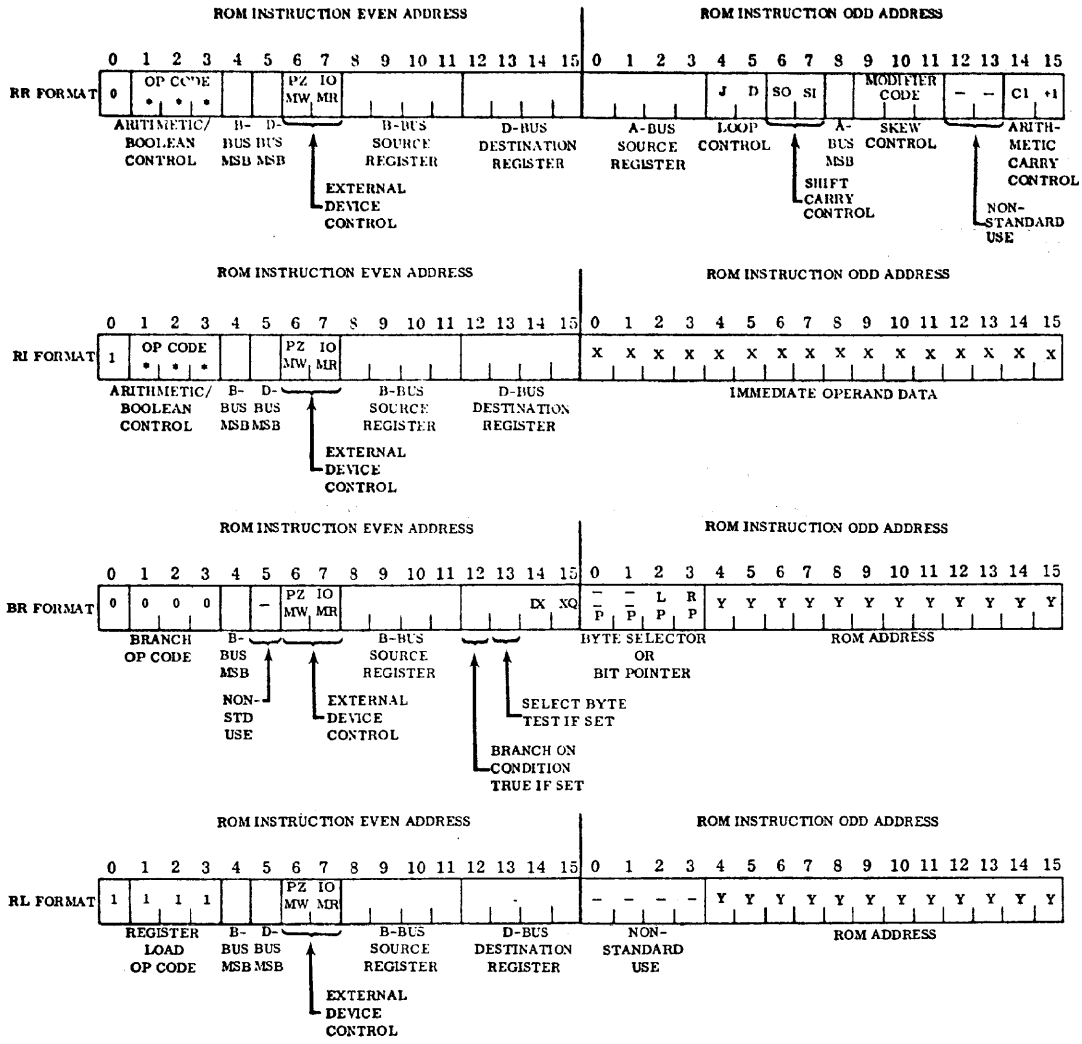
•ROM instructions select specific operations of the Arithmetic/Boolean, Branch, and Register Load functions of the META 4 computer. Instructions are grouped into four categories, as shown in Figure 2-1.

- •BR is the Branch format.
- •RR is the Register-Register format.
- •RI is the Register-Immediate format.
- •RL is the Register Load format.

ROM instructions are described in detail on the following pages.

Modifiers

•ROM instruction modifiers select operations of the computer in addition to those selected by the basic instructions. Multiple modifiers may be specified and will operate within the basic instruction cycle times. The modifiers are described in detail beginning on page 2-17.



ARITHMETIC/BOOLEAN CONTROL OP CODES

MNEMONIC	*** BINARY VALUE
BRZ BNZ	000
AND	001
OR	010
XOR	011
ADD	100
MULT	101
DIV	110
LOAD	111

SKEW CONTROL MODIFIER CODES

MNEMONIC	BINARY VALUE
(NONE)	000
R1	001
L1	010
SK	011
SE	100
R8	101
L8	110
EX	111

MISCELLANEOUS MODIFIERS

SYMBOL	SYMBOL
CI	L
D	PZ
IO	R
IX	SI
J	SO
MR	XQ
MW	+1
W	

ARITHMETIC/BOOLEAN CONTROL MICROASSEMBLER PSEUDO-OP CODES

COPY	BO
JMP	BNO
NOP	BS
BRP	BNS
BRN	LDI
BC	SUBI
BNC	

ASSEMBLER CONTROL PSEUDO-OP CODES

EQU
EQUR
ORG
END
LIST
PNCH
EJCT

DATA STATEMENT PSEUDO-OP CODE

HEX

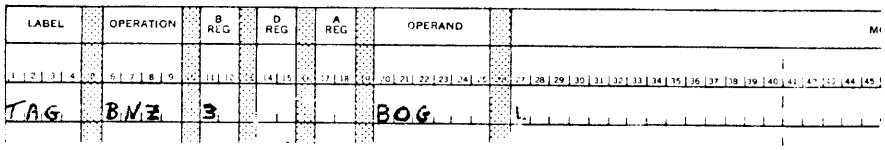
FIGURE 2-1. META 4 SERIES 16 COMPUTER CONTROL INSTRUCTIONS

BNZ

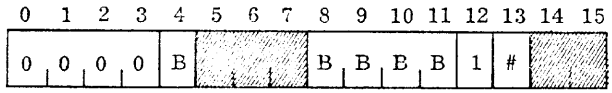
Format BR BRANCH IF NONZERO CONDITION

Valid Modifiers: R L R,L W IX XQ PZ IO

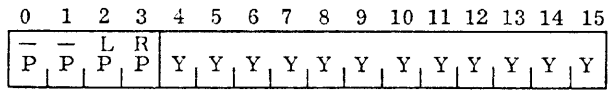
The condition specified by modifiers or a register bit position is tested. If the test result is nonzero, the next instruction is taken from the even ROM location specified in the operand field. If the test result is zero, the next sequential instruction is executed. Registers and machine conditions are not changed. The operand field must contain either a label or an absolute address. The least significant bit of an address is ignored and interpreted as zero. Logical indexing applies if IX modifier is specified.



In this example, the left byte (8 bits) of Register 3 are tested. If the byte is nonzero, a Branch to the address in the operand field occurs. If the byte is zero, the next sequential instruction is executed.



ROM INSTRUCTION, EVEN ADDRESS



ROM INSTRUCTION, ODD ADDRESS

B = B-Register

P = Bit Pointer

R = Test Right Byte

L = Test Left Byte

= Byte or Bit Test Operation Select

1 = Byte

0 = Bit

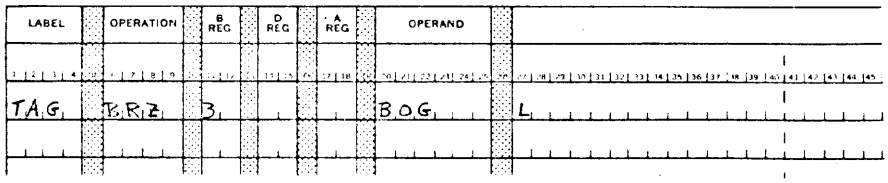
Y = ROM Address

BRZ

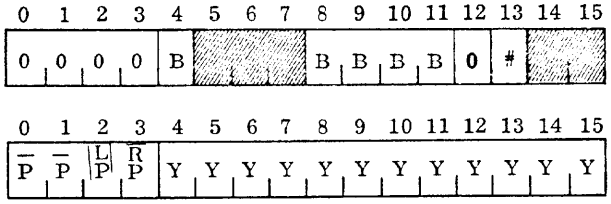
Format BR BRANCH IF ZERO CONDITION

Valid Modifiers: R L R,L W IX XQ PZ IO

The condition specified by modifiers or by a bit position is tested. If the test result is zero, the next instruction is taken from the ROM location specified in the operand field. If the result is nonzero, the next sequential instruction is executed. Registers and machine conditions are not changed. The operand field must contain either a label or an absolute address. The least significant bit of an address is ignored and interpreted as zero. Logical indexing applies if IX modifier is specified.



In these examples, the left byte (8 bits) of Register 3 are tested. If the byte is zero, a Branch occurs to the address in the operand field. If the byte is nonzero, the next sequential instruction is executed.



B = B-Register

P = Bit Pointer

R = Test Right Byte

L = Test Left Byte

= Byte or Bit Test Operation Select

1 = Byte

0 = Bit

Y = ROM Address

The use of mnemonics BNZ and BRZ is shown in the following table:

	Bits 0 - 7 zero Bits 8 - 15 zero	Bits 0 - 7 zero Bits 8 - 15 nonzero	Bits 0 - 7 nonzero Bits 8 - 15 zero	Bits 0 - 7 nonzero Bits 8 - 15 nonzero
BRZ W	•			
BRZ R	•		•	
BRZ L	•	•		
BRZ R, L	•	•	•	
BNZ R, L				•
BNZ R		•		•
BNZ L			•	•
BNZ W		•	•	•

• indicates conditions for successful branch

NOTE: Branch testing of an I/O register input must not be attempted unless the I/O system is stabilized at the time. Stabilization is assured by input/output system data via timing interlocks. Stabilization is not assured for non-synchronized inputs such as those used for interrupts. The effect of testing a nonstabilized input may be a ROM program branch to an address which is neither the next sequential address nor the expected branch address.

AND

Format RR LOGICAL AND

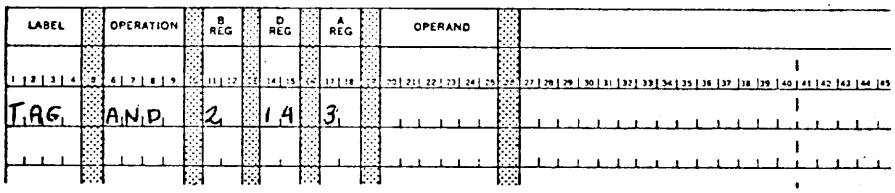
Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW

The contents of the A-register and the B-register are AND'ed bit by bit. The result is stored in the D-register. The four possible AND'ing results are:

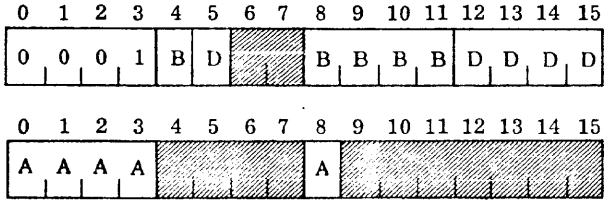
BIT VALUES

A-REGISTER	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER RESULT	1	0	0	0

The contents of the A- and B-registers are left unchanged by this operation.



In this example, the contents of Registers 2 and 3 are AND'ed and the result appears in Register 14.



ROM INSTRUCTION, EVEN ADDRESS

ROM INSTRUCTION, ODD ADDRESS

A = A-Register

B = B-Register

D = D-Register

OR

Format RR LOGICAL INCLUSIVE OR

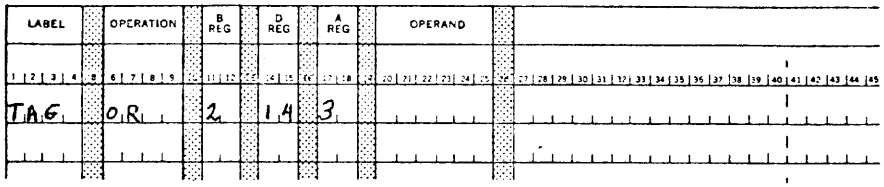
Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW

The contents of the A-register and the B-register are Inclusive OR'ed bit by bit. The result is stored in the D-register. The four possible OR'ing results are:

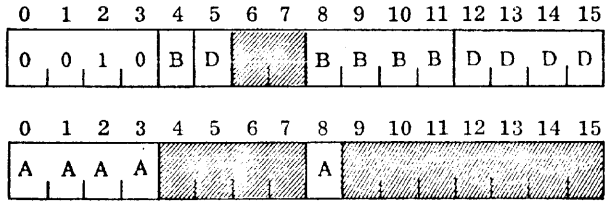
BIT VALUES

A-REGISTER	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER RESULT	1	1	1	0

The contents of the A- and B-registers are left unchanged by this operation.



In this example, the contents of Registers 2 and 3 are Inclusive OR'ed and the result appears in Register 14.



A = A-Register

B = B-Register

D = D-Register

XOR

Format RR LOGICAL EXCLUSIVE OR

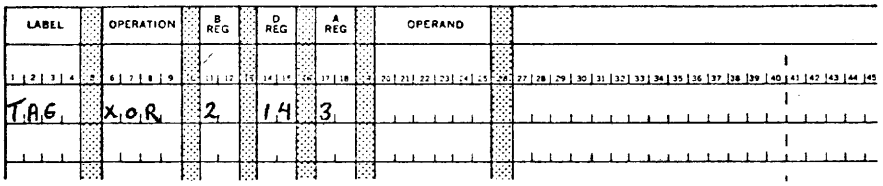
Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW

The contents of the A-register and the B-register are Exclusive OR'ed bit by bit. The result is stored in the D-register. The four possible XOR'ing results are:

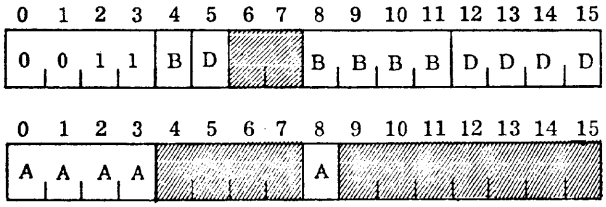
BIT VALUES

A-REGISTER	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER RESULT	0	1	1	0

The contents of the A- and B-registers are left unchanged by this operation.



In this example, the contents of Registers 2 and 3 are Exclusive OR'ed and the results appear in Registers 14.



ROM INSTRUCTION, EVEN ADDRESS

ROM INSTRUCTION, ODD ADDRESS

A = A-Register

B = B-Register

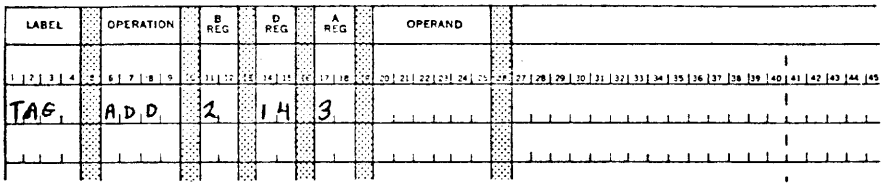
D = D-Register

ADD

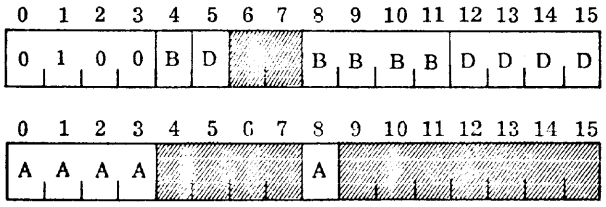
Format RR ADD

Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW CI +1

The contents of the A-register and the B-register are added. The result is stored in the D-register. Addition is carried out in two's complement format. Carry input to the least significant bit is controlled by CI, which enables the previous carry condition as input; and by +1, which forces carry input. The carry condition is set to correspond to the carry from bit 0 and the overflow is set to correspond to the Exclusive OR of the carries from bits 1 and 0. The A- and B-registers are left unchanged by this operation, but the carry and overflow bits are changed.



In this example, the contents of Registers 2 and 3 are added and the sum appears in Register 14.



A = A-Register

B = B-Register

D = D-Register

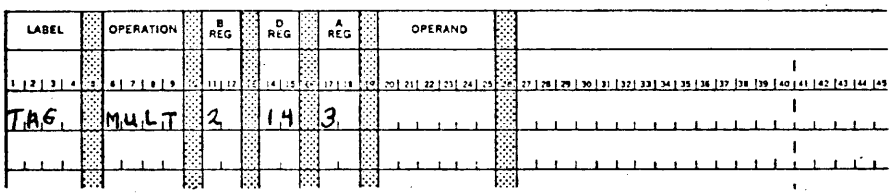
MULT

Format RR MULTIPLY STEP

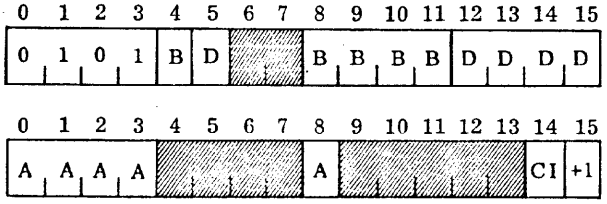
Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW CI +1

If the shift condition is 1 prior to the MULT instruction, the contents of the A- and B-registers are added. The result is stored in the D-register. Addition is carried out in two's complement format. Carry input to the least significant bit is controlled by CI, which enables the previous carry condition as input; and by +1, which forces carry input. The carry condition is set to correspond to the carry from bit 0 and the overflow is set to correspond to the Exclusive OR of the carries from bits 1 and 0. The A- and B-registers are left unchanged by this operation, but the carry and overflow bits are changed.

If the shift condition is zero prior to the MULT instruction execution, the B-register data is inhibited so that the A-register data passes through the input adder unchanged.



In this example, the contents of Register 2 are added to the contents of Register 3 and the sum is stored in Register 14. Multiplication routines are constructed with this instruction and its modifiers.



ROM INSTRUCTION, EVEN ADDRESS

ROM INSTRUCTION, ODD ADDRESS

A = A-Register

B = B-Register

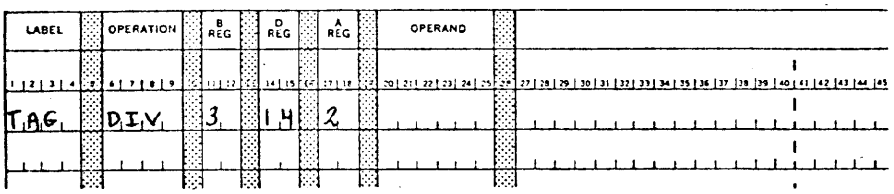
D = D-Register

DIV

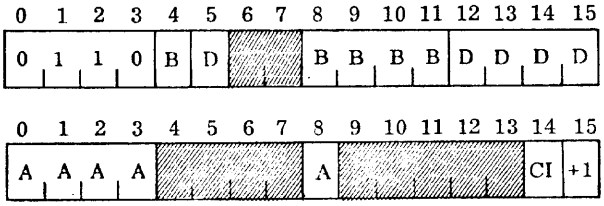
Format RR DIVIDE STEP

Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW CI +1

The contents of the A- and B-registers are added. If the sum is positive, the result is stored in the D-register. If the sum is a negative number, the D-register is not changed. Addition is carried out in two's complement format. Carry input to the least significant bit is controlled by CI, which enables the previous carry condition as input; and +1, which forces carry input. The carry condition is set to correspond to the Exclusive OR of the carries of bits 1 and 0. The carry and overflow conditions, and the shift condition (if SO is specified) are changed by the DIV instruction whether the sum is positive or negative.



In this example, the contents of Register 3 are added to the contents of Register 2. The sum is stored in Register 14.



ROM INSTRUCTION, EVEN ADDRESS

ROM INSTRUCTION, ODD ADDRESS

A = A-Register

B = B-Register

C = C-Register

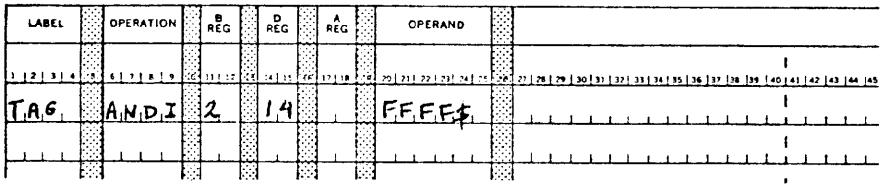
Format RI LOGICAL AND IMMEDIATE

Valid Modifiers: PZ IO MR MW

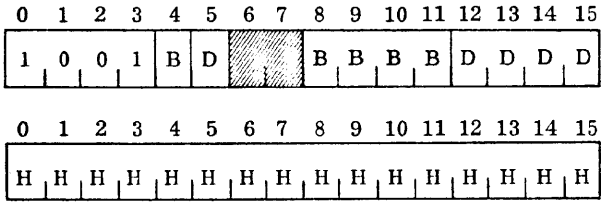
The contents of the B-register and the operand field are AND'ed bit by bit. The result is stored in the D-register. The operand must be either a left-justified hexadecimal constant or a label.

IMMEDIATE OPERAND	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER RESULT	1	0	0	0

The contents of the B-register are left unchanged by this instruction.



In this example, the contents of Register 2 are AND'ed with the hexadecimal value FFFF and the result appears in Register 14.



B = B-Register

D = D-Register

H = HEX Constant

ORI

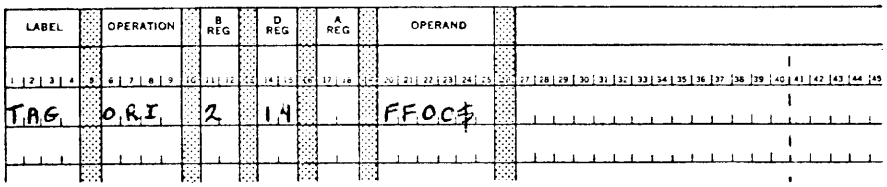
Format RI LOGICAL INCLUSIVE OR IMMEDIATE

Valid Modifiers: PZ IO MR MW

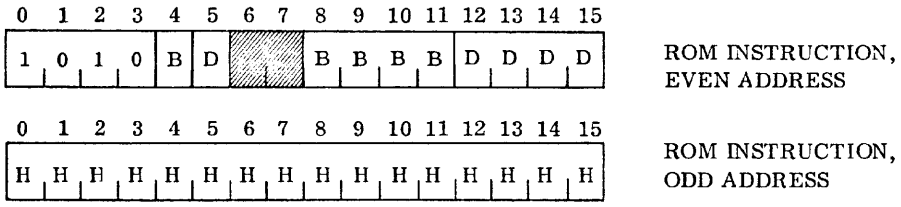
The contents of the B-register and the operand field are Inclusive OR'ed bit by bit. The result is stored in the D-register. The contents of the operand field must be either a left-justified hexadecimal constant or a label.

IMMEDIATE OPERAND	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER RESULT	1	1	1	0

The contents of the B-register are left unchanged by this instruction.



In this example, the contents of Register 2 are Inclusive OR'ed with the hexadecimal value FF00 and the result appears in Register 14.



B = B-Register

D = D-Register

H = HEX Constant

XORI

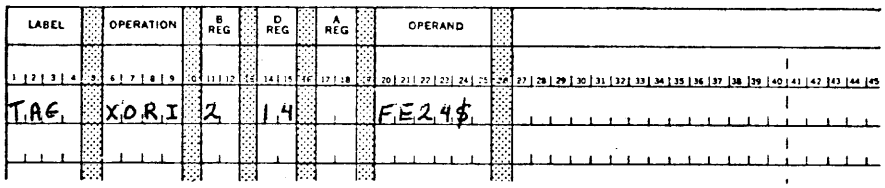
Format RI LOGICAL EXCLUSIVE OR IMMEDIATE

Valid Modifiers: PZ IO MR MW

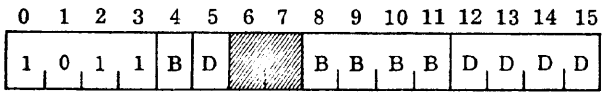
The contents of the B-register and the operand field are Exclusive OR'ed bit by bit. The result is stored in the D-register. The operand field must be either a left-justified hexadecimal constant or a label.

IMMEDIATE OPERAND	1	1	0	0
B-REGISTER	1	0	1	0
D-REGISTER	0	1	1	0

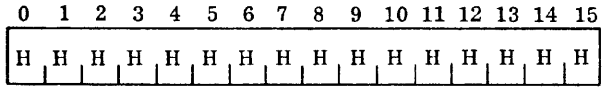
The contents of the B-register are left unchanged by this instruction.



In this example, the contents of Register 2 are Exclusive OR'ed with the hexadecimal value FE24 and the result appears in Register 14.



ROM INSTRUCTION, EVEN ADDRESS



ROM INSTRUCTION, ODD ADDRESS

B = B-Register

D = D-Register

H = HEX Constant

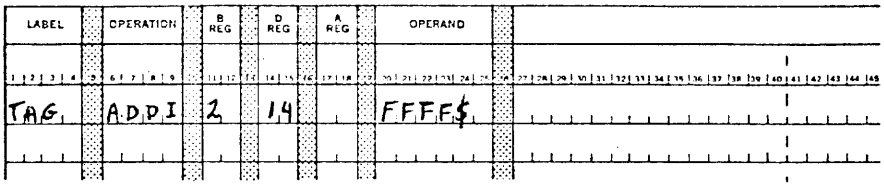
ADDI

Format RI LOGICAL ADD IMMEDIATE

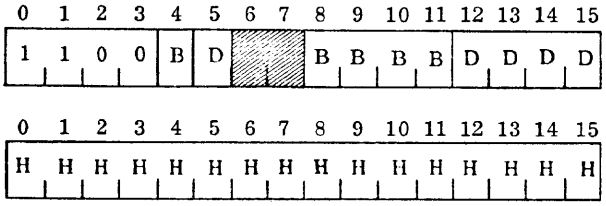
Valid Modifiers: PZ IO MR MW

The contents of the B-register and the operand field are added. The sum is stored in the D-register. The operand must contain a left-justified hexadecimal constant, or a label. Addition is carried out in two's complement format. The carry condition is set to correspond to the carry from bit 0 and the overflow is set to correspond to the Exclusive OR of the carries from bits 1 and 0.

The B-register is left unchanged by this instruction.



In this example, the contents of Register 2 are added to the hexadecimal value FFFF and the sum appears in Register 14.



B = B-Register

D = D-Register

H = HEX Constant

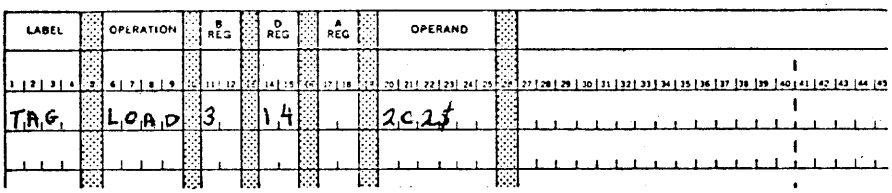
LOAD

Format RL LOAD REGISTER

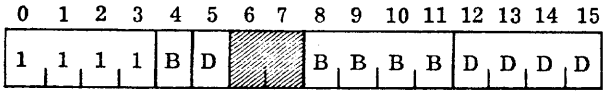
Valid Modifiers: PZ IO MR MW

Load is a two cycle instruction. During Cycle 1, the effective ROM address is obtained by Inclusive OR'ing the instruction's ROM address field with the contents of the Link register. During Cycle 2, the contents of the effective ROM address and the contents of the specified B-register are then Exclusive OR'ed bit by bit. The result is stored in the D-register.

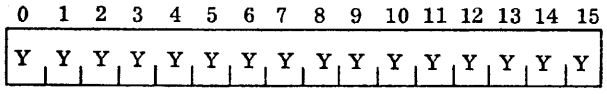
The contents of the B-register are unchanged by this instruction.



In this example, hexadecimal constant 2C2 and the contents of the Link register are Inclusive OR'ed to form an address. The contents of the address are Exclusive OR'ed with Register 3 and the final result is stored in Register 14.



ROM INSTRUCTION, EVEN ADDRESS



ROM INSTRUCTION, ODD ADDRESS

B = B-Register

D = D-Register

Y = ROM Address

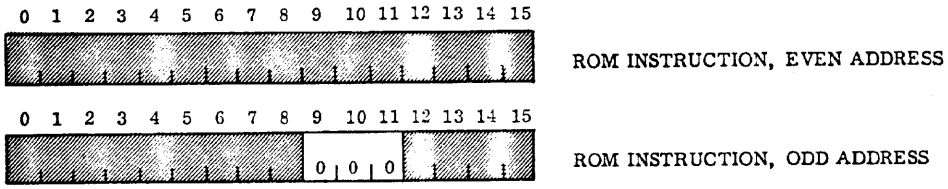
ROM Instruction Modifiers

• The ROM instruction modifiers are described in detail on the following pages. They are grouped as outlined below:

- Skew Control Modifiers
- Arithmetic Control Modifiers
- Instruction Loop Repeat Control Modifiers
- Branch Control Modifiers
- Input/Output and Memory Control Modifiers

(No Skew Control Modifier)

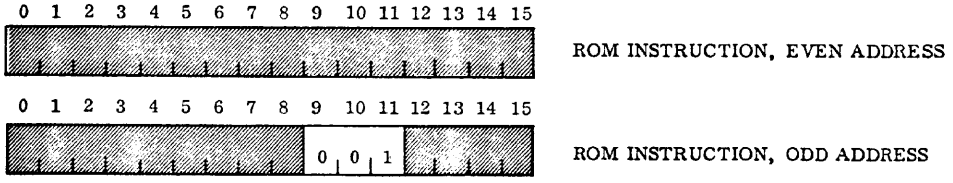
TRANSMIT DATA WITHOUT MODIFICATION



Output from the Arithmetic unit is transmitted without modification.

R1

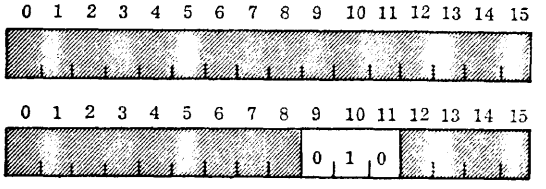
SHIFT RIGHT ONE PLACE



Output from the Arithmetic unit is displaced right one place. Spill from bit 15 may be saved in the Shift Condition register bit by the SO modifier. Entry to bit 0 from the previous shift condition is controlled by the SI modifier.

L1

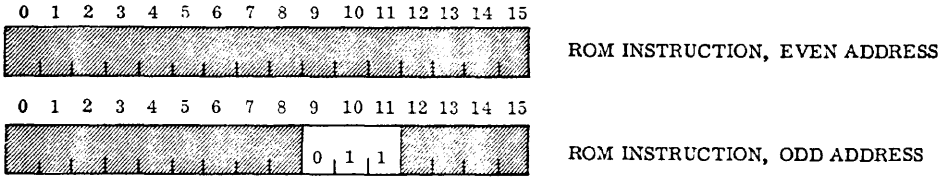
SHIFT LEFT ONE PLACE



Output from the Arithmetic unit is displaced left one place. Spill from bit 0 may be saved in the Shift Condition register bit by the SO modifier. Entry into bit 15 from the previous shift condition is controlled by the SI modifier.

SK

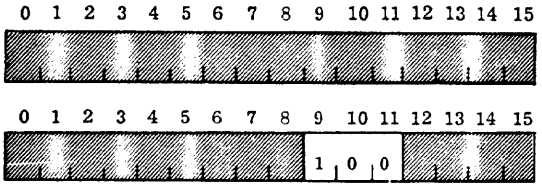
SCALE



Output from the Arithmetic unit is displaced one place to the right. Spill from bit 15 may be saved by the SO modifier. Entry to bit 0 is made from the current arithmetic extended sign during an ADD operation or from the Carry Condition register bit during operations other than ADD. If the SI modifier is specified concurrently with SK, entry to bit 0 is the OR between the Shift Condition register bit and the proper carry condition.

SE

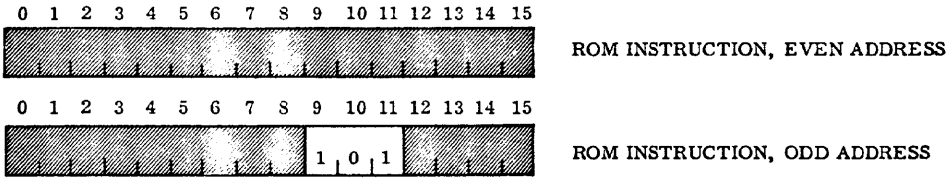
SIGN EXTEND



Output from the Arithmetic unit is transmitted with bits 0 through 7 replaced by copies of bit position 8.

R8

SHIFT RIGHT EIGHT PLACES

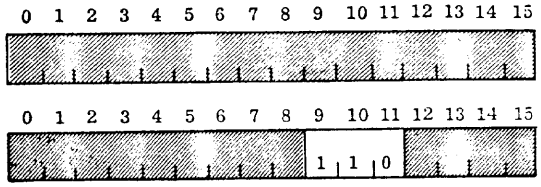


Output from the Arithmetic unit is displaced right eight places. Spill from the right is lost; zeros enter at the left.

L8

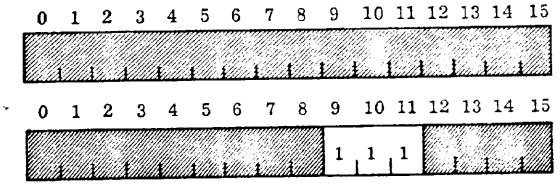
SHIFT LEFT EIGHT PLACES

Output from the Arithmetic unit is displaced left eight places. Spill from the left is lost; zeros enter at the right.



EX

EXCHANGE BYTES



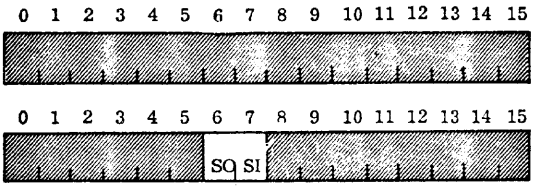
Output from the Arithmetic unit is rotated eight places so that bits 0 through 7 and bits 8 through 15 are interchanged.

SO

SHIFTER OUTPUT SPILL TO SHIFT CONDITION BIT

SI

SHIFTER INPUT ENTRY FROM SHIFT CONDITION BIT



Shifter spill is always from either bit 0 or bit 15 of the operand being shifted. SO is effective for any shifter control modifier code. Spill is the original contents of bit 0 for no shift, L1, SE, and L8; and the original contents of bit 15 for R1, SK, R8, and EX. Refer to the DIV instruction description for use of the SO modifier with operations other than a shift.

Shifter input is taken from the Shift Condition register bit when the SI modifer is specified. The SI modifier is enabled only for R1, L1, and SK modifiers and controls either bit 0 or bit 15 entry, as appropriate. If SK and SI modifiers are specified concurrently, the entry to bit 0 is the OR between the shift condition and the arithmetic carry.

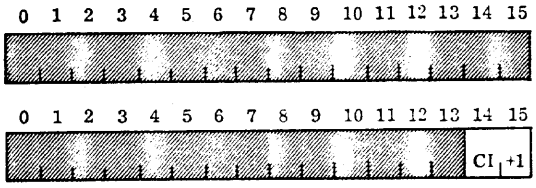
Circular shifts (end around) may be implemented by first executing a single shift operation (right or left, as appropriate) with register zero as the destination and SO specified. The shift condition bit will then be properly set so that subsequent shift operations with both SO and SI specified will be a circular shift.

CI

ENABLE ARITHMETIC CARRY INPUT

+1

FORCE ARITHMETIC CARRY INPUT



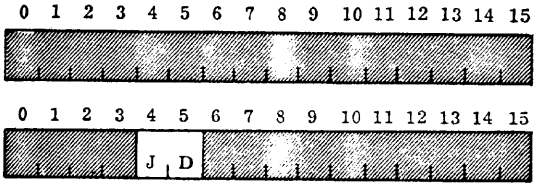
Carry input to the adder is controlled by CI and +1 modifiers. If neither is specified, the add is without carry input. If CI is specified, the previous carry condition is used as carry input to the least significant stage of the adder. If +1 is specified, a carry input is forced unconditionally, regardless of whether CI is also specified.

D

DECREMENT COUNTER

J

JUMP ON COUNTER NONZERO



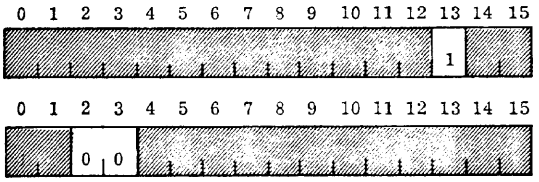
The low-order 8 bits of the Loop Counter are decremented and tested using the J and D modifiers. If J is specified without D, a branch to the address specified by the contents of the Link register occurs.

If D is specified without J, the counter is decremented.

If J is specified concurrently with D, a branch to the address specified by the contents of the Link register occurs unless the counter decrements to zero. The test is made after conclusion of the instruction.

W

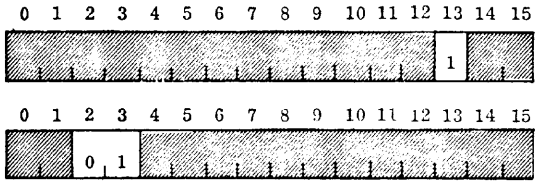
TEST WORD (RIGHT AND LEFT BYTES)



The contents of the register referenced by the B-bus address is tested for zero or nonzero condition.

R

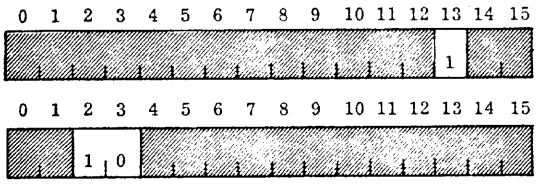
RIGHT BYTE TEST



The right byte (8 bits) of the contents of the register referenced by the B-bus address is tested for zero or nonzero condition.

L

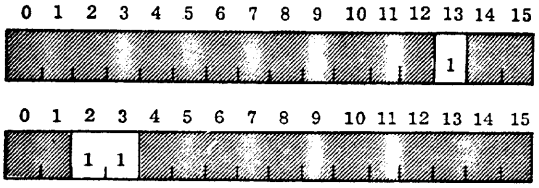
LEFT BYTE TEST



The left byte (8 bits) of the contents of the register referenced by the B-bus address is tested for zero or nonzero condition.

R, L

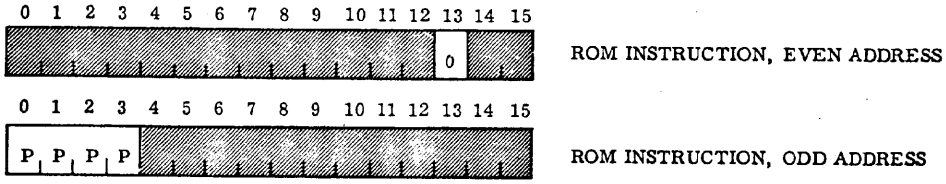
RIGHT OR LEFT BYTE TEST



The right byte (8 bits) and left byte (8 bits) of the contents of the register referenced by the B-bus address are checked independently for zero or nonzero, with the Inclusive OR of the results tested for the zero or nonzero condition.

(No Byte Test Modifiers)

TEST SPECIFIED BIT



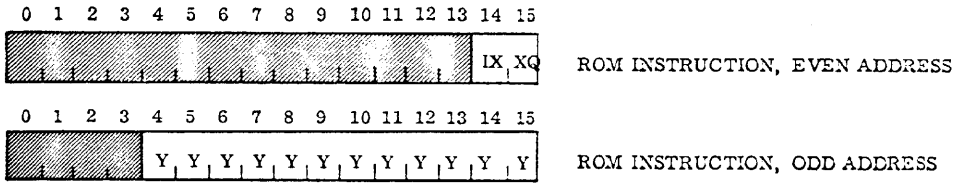
One bit of the contents of the register referenced by the B-bus address and the P-field is tested for zero or nonzero. This modifier enables a test and branch capability on any bit of any register and encompasses tests for even/odd, positive/negative, arithmetic carry, arithmetic overflow, and shift carry. The 4-bit P-field (pointer) is decoded to define one of 16 bit positions within the tested word. Pseudo-operation mnemonics are defined for positive/negative, carry, overflow, and shift condition tests.

IX

LOGICAL INDEX

XQ

EXECUTE ONE INSTRUCTION AFTER BRANCH



Logical indexing, if selected, OR's the contents of the Link register with the Y-field to form the effective address.

Execute mode inhibits changing the ROM address register if the Branch instruction test is successful. The effective address is used directly for the execution of one instruction and the control sequence then reverts back to that instruction which would have been executed had the Branch not been successful, unless that one instruction is itself a Branch instruction. Multilevel Branch and Execute instructions may be used with ultimate reversion of control back to that instruction which would have been done with only one level of Branch and Execute. If a Branch without Execute is in the multilevel Branch sequence, then reversion of control will not occur if the Branch without Execute is successful.

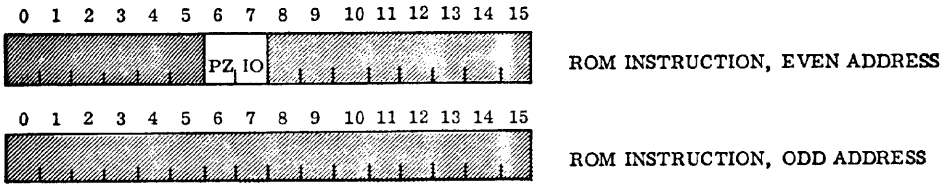
Execute mode may be considered as a capability for calling a one-instruction subroutine. The address in ROM is taken directly from the Link register (and Y-field of the instruction, if IX is specified), but the ROM address register is inhibited from copying the out-of-sequence ROM address reference.

IO

OUTPUT I/O REGISTER CONTROL SIGNALS

PZ

PROGRAM PAUSE FOR CONTROL SIGNAL INPUT



PZ and IO functions are enabled when any I/O register is specified by the B-bus or D-bus address fields of the instruction word. When the PZ modifier is specified, the program pauses until a control flip-flop is cleared by an external signal and the clearing pulse has terminated. If more than one I/O register is specified by the bus register address fields of the instruction word, the pause condition occurs while any one of the associated control flip-flops is set and the control signal is output on all of the control lines. When the IO modifier is specified, the I/O register control line signals are output and control flip-flops are set. The control flip-flops are cleared by the external equipment or by the computer Master Clear.

Pause occurs prior to any instruction execution; I/O control signals are output at the conclusion of the instruction execution. Concurrent PZ and IO selection operates to delay the instruction until the control flip-flop is reset (pause condition is released); the instruction then executes, and the IO control flip-flop is set again.

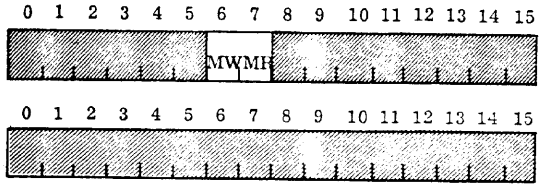
Note the possible conflicts: MW and MR modifiers use the same control word bit positions as PZ and IO modifiers. Conflicts may occur if memory and I/O registers are specified concurrently.

MW

INITIATE MEMORY WRITE

MR

INITIATE MEMORY READ



When either the address or data register for core memory is specified by the D-bus register field, MW or MR modifiers individually may set control flip-flops on the Memory-I/O register board.

While a control flip-flop is set, the microprogram pauses if 1) either the address or data register for core memory is subsequently referenced by a D-bus register field or 2) the Core Memory Data register is specified by a B-bus register field and the PZ modifier is specified. The microprogram resumes when the Done signal from core memory resets the control flip-flops. NOTE (1)

The PZ modifier must be used with memory register addresses in the B-bus field unless:

1. The data has been previously read using a PZ bit and cannot have changed since, NOTE (2) or

2. A memory register is also specified in the D-bus field.

NOTE (1) The program resumes at the leading edge of the Done signal for a B-bus pause, and resumes at the trailing edge of Done for a D-bus pause. This minimizes core memory data access time while still allowing proper ready-resume interlocking.

NOTE (2) Memory Read data is valid for approximately eight ROM instruction cycles, following an MR operation, unless a buffered Memory Data register is used.

Whenever MR and MW modifiers are specified concurrently, the core memory is read, the cycle is suspended prior to restoring the data, and the control flip-flops are reset. That core memory bank only waits (indefinitely) in the suspended state. The memory cycle may be completed when the core memory Data register is specified in the D-bus address field and 1) new data is written (using both MR and MW modifiers); or 2) bit 15 of the data is written into the protect bit position (using MW only); or 3) the original data is restored (using MR only).

If the core memory address register is specified by a D-bus address field while the core memory operation is suspended, and a new core address is attempted, the core memory will not accept the new address, but will accept any accompanying MW and MR modifiers to enable completion of the cycle.

Note the possible conflicts if memory registers and I/O registers are addressed concurrently. PZ and IO modifiers use the same bit positions as MW and MR.

Microassembler Pseudo-Ops

Microassembler pseudo-operations are provided in four distinct categories:

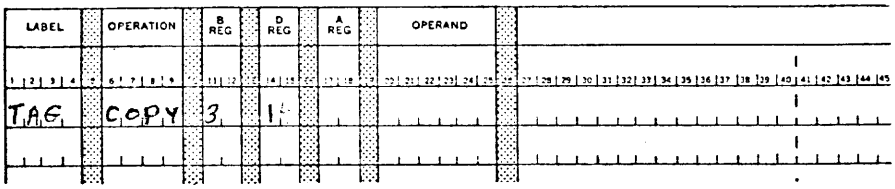
- Special mnemonics duplicating functions, which may also be implemented using standard mnemonics, and which are assembled normally into the program.
- Data statements, which provide constants for use by the Microassembler, and which are assembled into the program as constants.
- Data statements, which equate labels and constants that are not assembled into the program as constants.
- Microassembler mnemonics which enable control of the assembly process, and which are not assembled into the program as instructions.

COPY

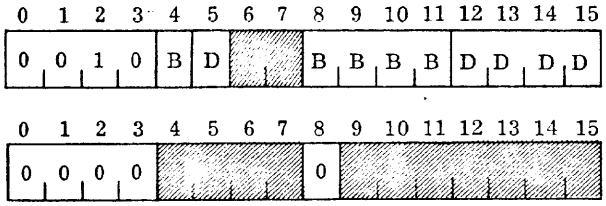
Format RR COPY

Valid Modifiers: R1 L1 SK SE R8 L8 EX SI
SO J D PZ IO MR MW

Data from the B-register is stored in the D-register. The B-register is left unchanged by this instruction.



In this example, the data in Register 3 is copied into Register 14.



B = B-Register

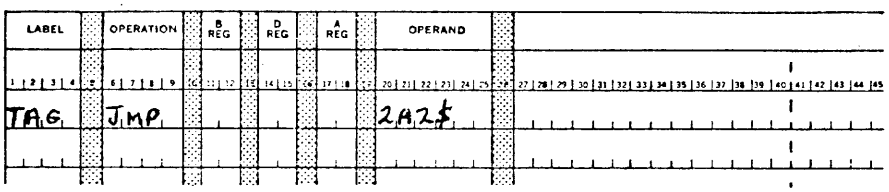
D = D-Register

JMP

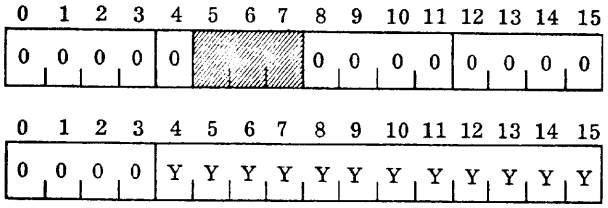
Format BR UNCONDITIONAL JUMP

Valid Modifiers: IX XQ PZ IO

This is an unconditional Branch to the address specified in the operand field. The operand field must contain a label or an absolute address and logical indexing applies if IX modifier is specified. Registers and machine conditions are not changed. The least significant address bit is ignored and interpreted as zero.



In this example, an unconditional Branch to Address 2A2 occurs.



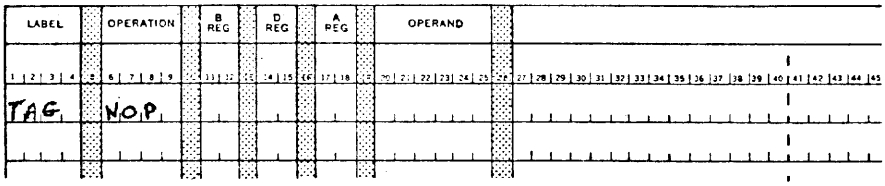
Y = ROM Address

NOP

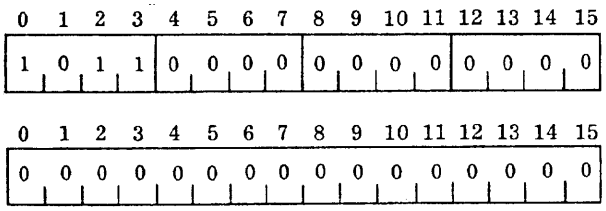
Format RR NO OPERATION

No Valid Modifiers

The codes set up by a NOP signify an immediate Exclusive OR of Register 0 with itself and the results left in Register 0. Nothing is changed.



In this example, the CPU does nothing for one machine cycle.

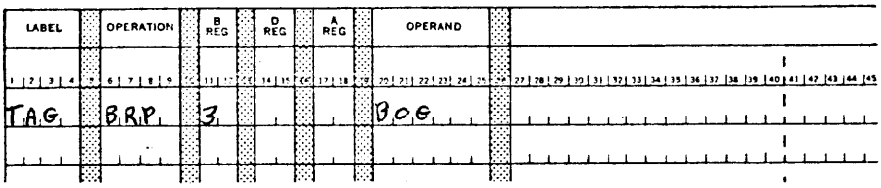


BRP

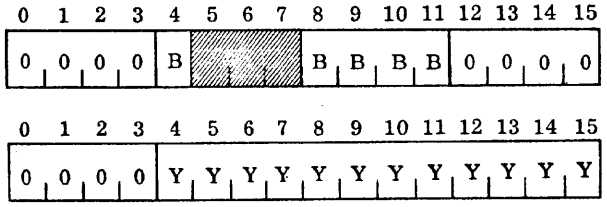
Format BR BRANCH IF REGISTER POSITIVE

Valid Modifiers: IX XQ PZ IO

The specified B-register is tested for positive or negative condition. If the test result is either positive or zero, the next instruction is taken from the ROM location specified in the operand field. If the test result is negative, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by a Branch instruction. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, Register 3 is tested. If it is either positive or zero, a Branch to BOG occurs. If it is negative, the next sequential instruction is executed.



B = B-Register

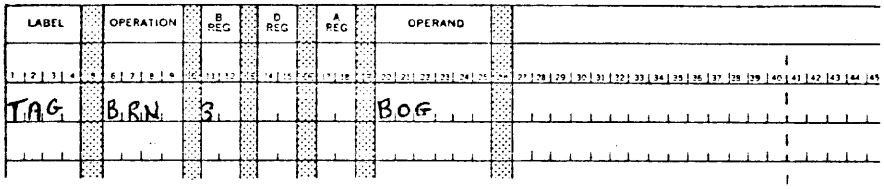
Y = ROM Address

BRN

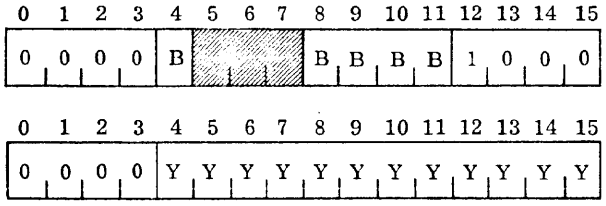
Format BR BRANCH IF REGISTER NEGATIVE

Valid Modifiers: IX XQ PZ IO

The specified B-register is tested for either positive or negative condition. If the test result is negative, the next instruction is taken from the ROM location specified in the operand field. If the test result is positive or zero, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by a Branch instruction. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, Register 3 is tested. If it is negative, a Branch to BOG occurs. If it is either positive or zero, the next sequential instruction is executed.



B = B-Register

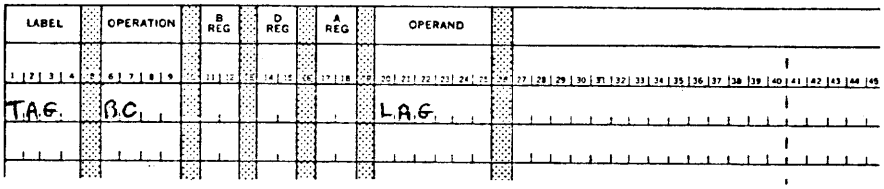
Y = ROM Address

BC

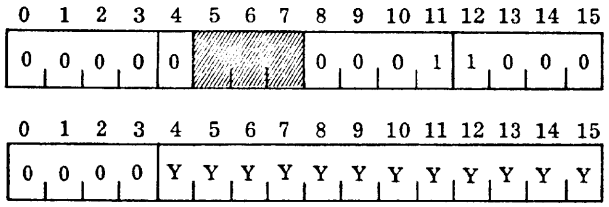
Format BR BRANCH IF CARRY ON

Valid Modifiers: IX XQ PZ IO

The carry indicator is tested. If the test result is true, the next instruction is taken from the ROM location specified in the operand field. If the test result is false, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by Branch instructions. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the carry indicator is tested. If it is true, a Branch to LAG occurs. If it is false, the next sequential instruction is executed.



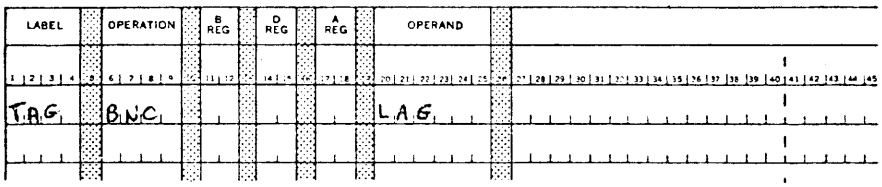
Y = ROM Address

BNC

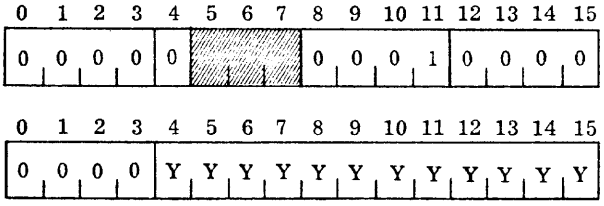
Format BR BRANCH IF CARRY OFF

Valid Modifiers: IX XQ PZ IO

The carry indicator is tested. If the test result is false, the next instruction is taken from the ROM location specified in the operand field. If the test result is true, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by Branch instructions. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the carry indicator is tested. If it is false, a Branch to LAG occurs. If it is true, the next sequential instruction is executed.



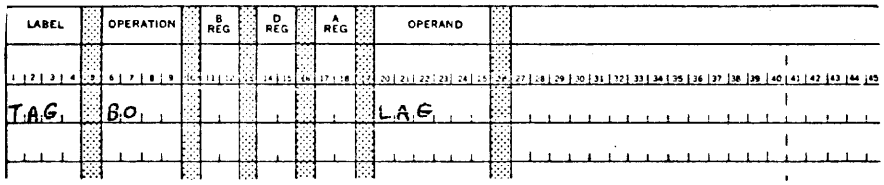
Y = ROM Address

BO

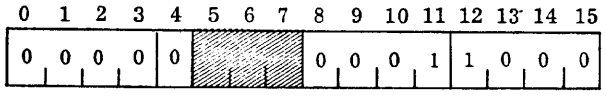
Format BR BRANCH IF OVERFLOW ON

Valid Modifiers: IX XQ PZ IO

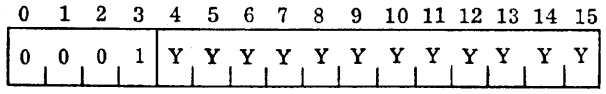
The overflow indicator is tested. If the test result is true, the next instruction is taken from the ROM location specified in the operand field. If the test result is false, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by Branch instructions. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the overflow indicator is tested. If it is true, a Branch to LAG occurs. If it is false, the next sequential instruction is executed.



ROM INSTRUCTION, EVEN ADDRESS



ROM INSTRUCTION, ODD ADDRESS

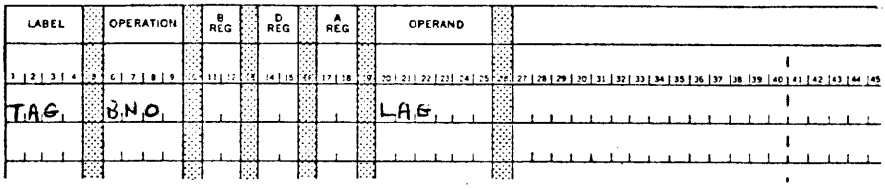
Y = ROM Address

BNO

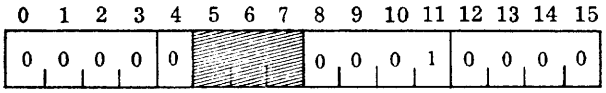
Format BR BRANCH IF NO OVERFLOW

Valid Modifiers: IX XQ PZ IO

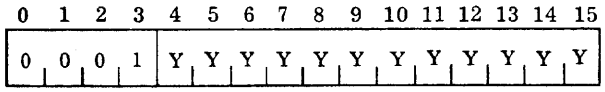
The overflow indicator is tested. If the test result is false, the next instruction is taken from the ROM location specified in the operand field. If the test result is true, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed by Branch instructions. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the overflow indicator is tested. If it is false, a Branch to LAG occurs. If it is true, the next sequential instruction is executed.



ROM INSTRUCTION, EVEN ADDRESS



ROM INSTRUCTION, ODD ADDRESS

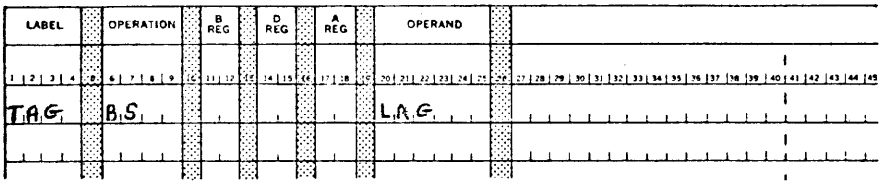
Y = ROM Address

BS

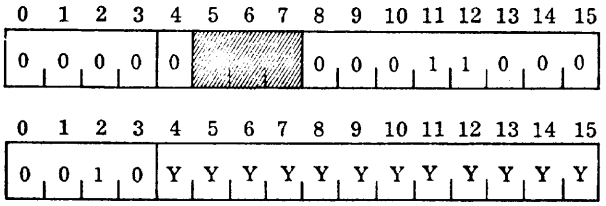
Format BR BRANCH IF SHIFT ON

Valid Modifiers: IX XQ PZ IO

The shift indicator is tested. If the test result is true, the next instruction is taken from the ROM location specified in the operand field. If the test result is false, the next sequential instruction is executed. The operand field may contain a label or an absolute address. Registers and machine conditions are not changed. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the shift indicator is tested. If it is true, a Branch to LAG occurs. If it is false, the next sequential instruction is executed.



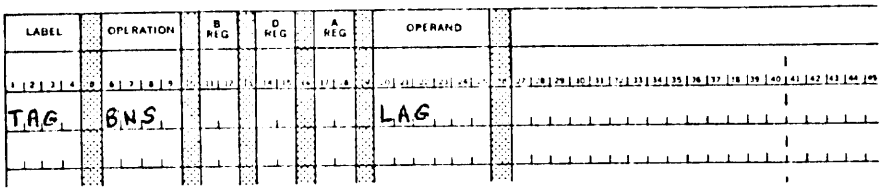
Y = ROM Address

BNS

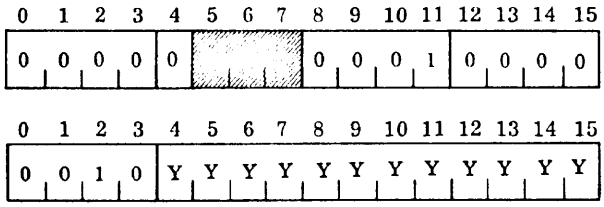
Format BR BRANCH IF NO SHIFT

Valid Modifiers: IX XQ PZ IO

The shift indicator is tested. If the test result is false, the next instruction is taken from the ROM location specified in the operand field. If the test result is true, the next sequential instruction is executed. The operand field must contain either label or an absolute address. Registers and machine conditions are not changed. Logical indexing applies if IX modifier is specified. The least significant bit of Y is ignored and interpreted as zero.



In this example, the shift indicator is tested. If it is false, a Branch to LAG occurs. If it is true, the next sequential instruction is executed.



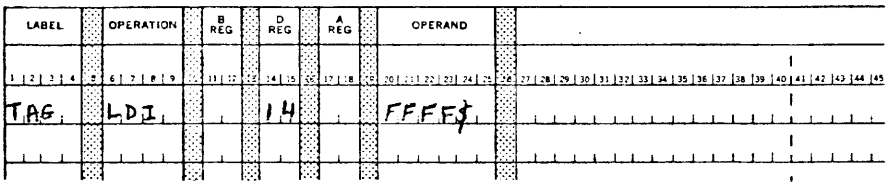
Y = ROM Address

LDI

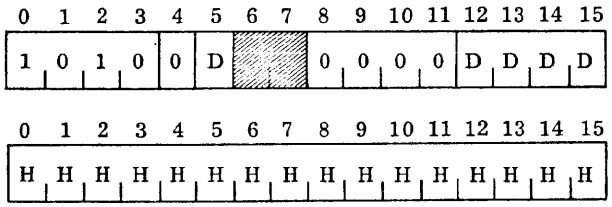
Format RI LOAD IMMEDIATE

Valid Modifiers: PZ IO MR MW

The left-justified hexadecimal constant or label in the operand field is stored in the D-register.



In this example, the hexadecimal constant FFFF is loaded into Register 14.



D = D-Register

H = HEX Constant

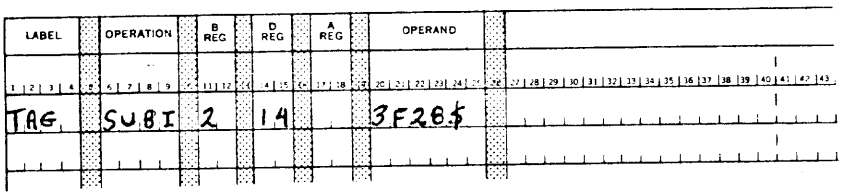
SUBI

Format RI SUBTRACT IMMEDIATE

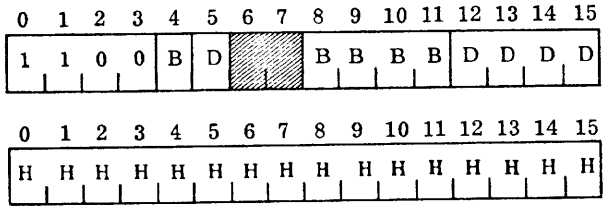
Valid Modifiers: PZ IO MR MW

The two's complement of the operand field is used as the operand of an ADDI instruction. The contents of the B-register are added to the two's complement of the hexadecimal constant in the operand field. The sum is stored in the D-register. The operand must contain either a left-justified, hexadecimal constant, or a label. Addition is carried out in two's complement format. The carry condition is set to correspond to the carry from bit 0 and the overflow is set to correspond to the Exclusive OR of the carries from bits 1 and 0.

The B-register is left unchanged by this instruction.



In this example, the contents of Register 2 are added to the two's complement of the hexadecimal constant 3F28 and the sum appears in Register 14.



B = B-Register

D = D-Register

H = HEX Constant

HEX

HEXADECIMAL CONSTANT

No Valid Modifiers

This instruction permits tables or constants to be stored. Data specified in the operand field is stored in the current address and the current address plus one. The data may overflow the operand field. Either labels or constants (not mixed) may be specified. Constants must be terminated by a $. Both must be left-justified in the field. After a HEX pseudo-op, two 16-bit words may be specified (by labels only) using a comma (,) as a separator. A slash (/) separating two labels indicates that the low order 8 bits of each label are linked to form a 16-bit word.

LABEL	OPERATION	B REG	D REG	A REG	OPERAND
TAG	HEX				2438AB46$
GAG	HEX				2A2$
RAG	HEX				DOG,BOG
2A2$	HEX				DOG/BOG,HOG/FOG

In the first example, the hexadecimal constant 2438AB46 is stored in double word TAG.

In the second example, the hexadecimal constant 02A2 is stored in double word GAG + 1 and GAG is 0000.

In the third example, the address of label DOG is stored in RAG and the address of label BOG is stored in RAG + 1.

In the fourth example, the double word at address 2A2 and 2A3 is set up in the following order:

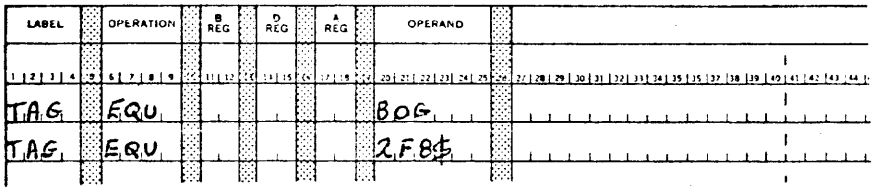
- Bits 0-7 of 2A2 = the low order 8 bits of DOG.
- Bits 8-15 of 2A2 = the low order 8 bits of BOG.
- Bits 0-7 of 2A3 = the low order 8 bits of HOG.
- Bits 8-15 of 2A3 = the low order 8 bits of FOG.

EQU

EQUATE LABEL

No Valid Modifiers

This instruction equates a label with an absolute address or with another label. The label field must contain a label. The operand field may contain a predefined label or a hexadecimal address constant. If a hexadecimal constant is used, it must be even, left-justified, followed by a $.



In the first example, the label TAG is equated with the label BOG. BOG must have been previously defined or an error will be indicated. The two labels may then be used interchangeably.

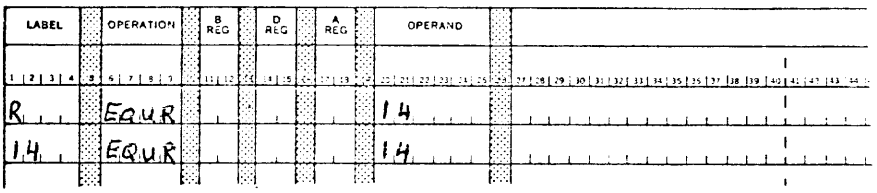
In the second example, the label TAG is equated to 2F8.

EQUR

EQUATE REGISTER LABEL

No Valid Modifiers

This instruction will EQUATE a one- or two-character name with a register. The name is specified in the label field and the register is specified in the operand field. Before any register may be used in a program, it must be equated to a name. Legal characters for a name are any alphanumeric characters. EQUR statements are usually kept at the beginning of the program.



In the first example, Register 14 is named R. Any reference made to Register 14 must be made by the name R.

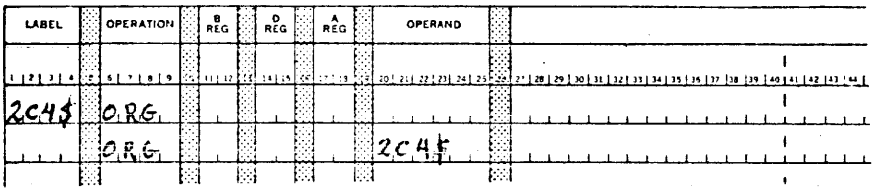
In the second example, Register 14 is named 14. Any reference made to Register 14 must be made by the name 14.

ORG

ORIGINATE ASSEMBLY

No Valid Modifiers

The Microassembler Instruction Counter address is set to the value specified either in the label field or in the operand field. The value specified in either of these fields must be a hexadecimal constant, left-justified, followed by a $. This hexadecimal constant must be an even value.



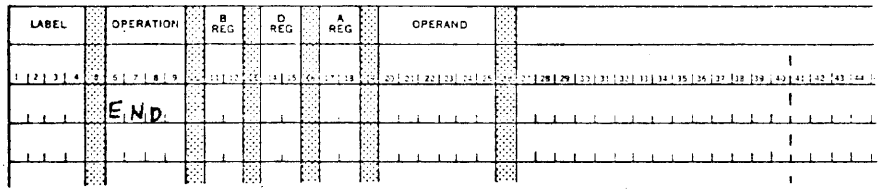
In either of these examples, the address of the next instruction will be 2C4 and subsequent instructions will continue in sequence from there.

END

END ASSEMBLY

No Valid Modifiers

This instruction terminates the program. An END causes the Microassembler to end pass one and begin pass two. This instruction is required to end a program. A label may not be used with this instruction.

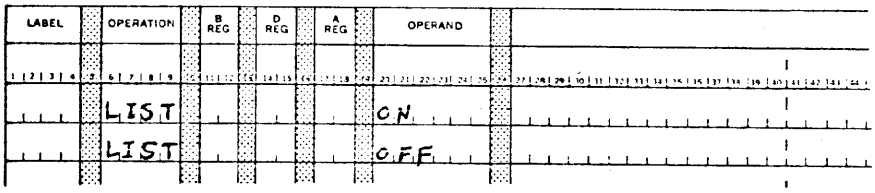


LIST

LISTING PRINTER CONTROL

No Valid Modifiers

This instruction causes listing on the selected device to be either started or stopped. If the operand field contains ON, listing begins. If the operand field contains OFF, listing stops. If no LIST instruction is used, a List On Condition is assumed. The LIST instruction may be used to list small portions of long assemblies.



In the first example, listing starts on the selected listing device.

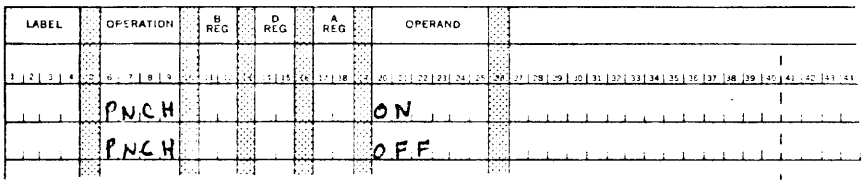
In the second example, listing stops.

PNCH

PUNCH CONTROL

No Valid Modifiers

This instruction causes punching of binary output on the selected device. If the operand field contains ON, punching begins. If the operand field contains OFF, punching stops. If no PNCH instruction is used, a Punch On Condition is assumed. The PNCH instruction may be used to punch small portions of a large assembly.



In the first example, punching starts.

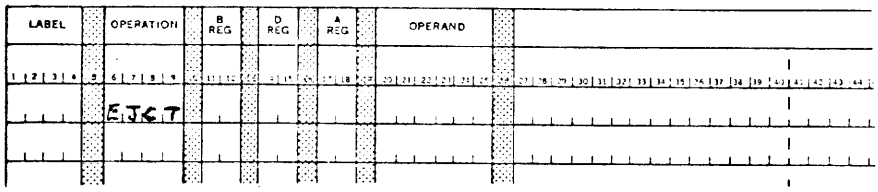
In the second example, punching stops.

EJCT

EJECT PAGE

No Valid Modifiers

This instruction causes the selected listing device to terminate the present page and begin the next page.



APPENDIX A

META 4 COMPUTER SYSTEM
PROGRAMMING TECHNIQUES AND EXAMPLES

APPENDIX A

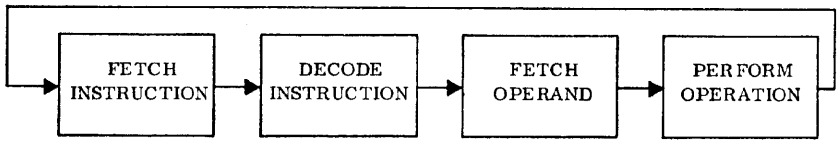
META 4 COMPUTER SYSTEM PROGRAMMING TECHNIQUES AND EXAMPLES

EMULATION

• Core memory instruction sets for other computers are readily implemented in the Digital Scientific META 4 Computer's Read-Only Memory (ROM). Firmware performs the operations which would be carried out by hardware in computers that lack control memories for sequencing.

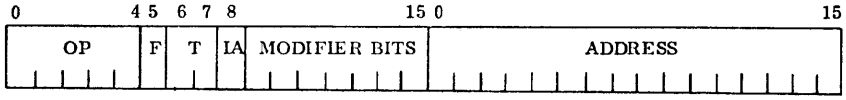
Emulating an IBM 1130 computer typifies the programming techniques for 16-bit systems. The sequences shown here have been prepared directly from instruction descriptions in the programming manual and from other sources such as timing charts.

The basic functions of any emulation are diagrammed below:



Formats for the instruction to be emulated are as follows:

Long Instruction Format (IBM 1130)



- OP (Operation) Code. These five bits specify the operation to be performed.

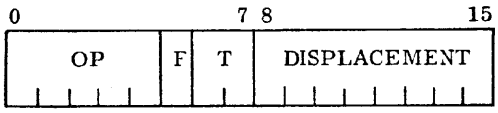
- F (Format). The F bit controls the instruction format: 0 = short format; 1 = long format.

- T (Tag). These two bits specify the register to be used in effective address generation: 00 indicates the I-register; 01 indicates XR1; 10 indicates XR2; and 11 indicates XR3.

- IA (Indirect Address). A zero indicates a direct address (contained in the second word). A 1 bit designates an indirect address.

- Modifier Bits. Bit positions 9 through 15 have various uses as modifiers.

Short Instruction Format (IBM 1130)



- The first eight bits of the short format are the same as those of the long format. The second eight bits contain the displacement, which is added to data in the register specified by the tag bits to form the effective address. Bit 8 is treated as a sign bit and is extended into bit positions 0 through 7 to obtain a 16-bit number. Negative numbers are expressed as two's complement.

Instruction Decoding

- The instructions decode readily using a table look-up on eight bits of the instruction, which establishes not only the operation to be performed, but also the format and the method of operand-effective address generation.

Since most operations require an operand, a table can be devised to direct the reading of the operand and the subsequent operation to be performed. Using the Subtract instruction as an example, assume that Subtract is located at 044_{16} of the ROM and that Operand Read routines are located as follows:

SUBROUTINE NAME		ROM ADDRESS
ORSI	(Operand Read, Short Format, Relative to I-Register)	F00
ORS1	(Operand Read, Short Format, Index Register 1)	F06
ORS2	(Operand Read, Short Format, Index Register 2)	F0C
ORS3	(Operand Read, Short Format, Index Register 3)	F12
OPRL	(Operand Read, Long Format)	F40

The table is written in META 4 Computer Micro-assembler language using the HEX pseudo-op.

The Microassembler generates the following table for Subtract instructions:

```
LOC.   INST.    LAB.  OP   BR DR AR  OPRAND MODIFIERS AND COMMENTS
0E90   44004406 E90SHEX              S/ORSI ,S/ORS1          SUBTRACT
0E92   440C4412     HEX              S/ORS2 ,S/ORS3
0E94   44404440     HEX              S/OPRL,S/OPRL
0E96   44404440     HEX              S/OPRL,S/OPRL
```

A META 4 Computer address from the table may not exceed eight bits; therefore, the most significant portion of the address is supplied by the program using the logical index facility.

System Conditions

- Core storage location 500 contains 9210_{16}. This is a Subtract instruction.
- The accumulator contains 300_{16}.
- The operand in location 520 contains 150_{16}.

  Index Register 2 contains 510_{16}.

META 4 Computer Registers

- The META 4 Computer registers with their mnemonics are:

```
ERR LOC.  INST.   LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

                        PNCH          OFF
                  *PROGRAM TO SIMULATE THE IBM 1130
                  *INSTRUCTION SET
                  *
      000         0     EQUR          0              ADDRESS ZERO
      001         C     EQUR          1$             COUNTER REGISTER
      002         L     EQUR          2$             LINK REGISTER
      003         S     EQUR          3              SCRATCH ACCUMULATOR
      004         M     EQUR          4              MEMORY ADDRESS REG
      005         D     EQUR          5              MEMORY DATA REGISTER
      006         Y     EQUR          6              IOCC OUT,INTERRUPT IN
      007         Z     EQUR          7              I/O DATA IN AND OUT
      014         A     EQUR          14             ACCUMULATOR
      015         Q     EQUR          15             ACCUMULATOR EXTENSION
      016         U     EQUR          16             TEMP ACCUMULATOR
      017         I     EQUR          17             INSTRUCTION ADDR REG
      018         X     EQUR          18             STATUS REG
      019         O     EQUR          19             OPERAND REGISTER
      01A         1     EQUR          1A             INDEX 1
      01B         2     EQUR          1B             INDEX 2
      01C         3     EQUR          1C             INDEX 3
      01D         K     EQUR          1D             PRIORIYY MASK REG
      01E         H     EQUR          1E$            CHARACTERISTIC REG
      01F         P     EQUR          1F             PRIORITY REG
```

APPENDIX B

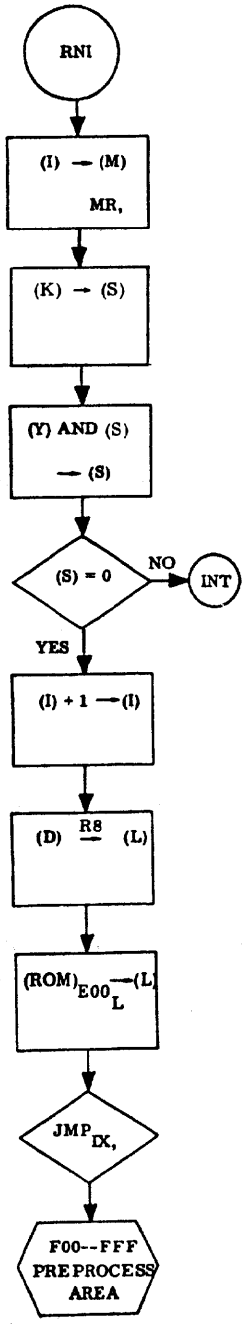
META 4 SYSTEM SAMPLE PROGRAMS AND SAMPLE FLOWCHARTS

APPENDIX B

META 4 SYSTEM SAMPLE PROGRAMS AND SAMPLE FLOWCHARTS

```
LOC.  INST.    LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

1BC 29740000 RNI   COPY I  M           MR.          MOVE I TO MEM ADDRESS
             *                                      AND READ NEXT INST.
1BE 28D30000       COPY K  S                         MOVE MASK TO DOUBLE BUS REG
1C0 10633000       AND  Y  S  S                      AND MASK WITH RAW INT
1C2 003C01D0       BNZ  S        INT     W.          VALID INTERRUPT
1C4 CC770001       ADDI I  I     1$                  INCREMENT I
1C6 20520050       COPY D  L             R8.         SHIFT OP CODE INTO L
1C8 F0020E00       LOAD 0  L     E00$                LOAD THE LINK FROM
                   *               THE TABLE STARTING AT
                   *               E00 AND INDEXED BY
                   *               THE CONTENTS OF L
1CA 00020F00       JMP           F00$   IX           JUMP TO OPERAND READ AREA
```



Copy the Instruction Address register, I (500), into the Memory Address register, M. Initiate a Read from location 500 in core storage.

Copy K (Single-Bus Accumulator) to S (Double-Bus Accumulator).

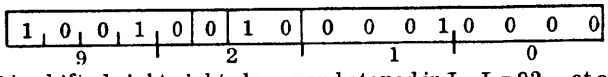
During the time required for memory to react, interrupts may be tested without time penalty. In this case, assume that the priority mask in the K-register when logically AND'ed with the raw interrupts in the Y-register produce a zero, which is stored in the S-register for subsequent testing.

The S-register is tested for zero; if S is zero, any interrupts which may be in T are of a lower priority than the one being serviced and are, therefore, deferred.

The I-register is also incremented without time penalty:

I (500) +1 → I (501).

When the Memory Data register, D, becomes available it contains the instruction. In this case, a short format is illustrated and is as follows:



$D = 9210_{16}$. D is shifted right eight places and stored in L. $L = 92_{16}$ at end of instruction.

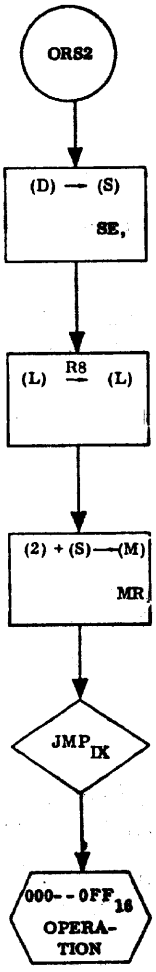
A Load instruction with the address field set to $E00_{16}$ is used to read the contents of the table starting at E00. The Load instruction is indexed logically by the Link register giving an effective Read-Only Memory (ROM) address of $E92_{16}$. In this example, the contents of ROM location E92 are placed in the Link register. The Link register, L, now contains $440C_{16}$. Load is a two-cycle instruction.

A Jump-to-F00 indexed by the Link causes program execution in the ROM to continue at location $F0C_{16}$. ($440C_{16}$ OR $F00_{16}$ = $4F0C_{16}$. The ROM Address register is 12 bits; therefore, the effective address is $F0C_{16}$.)

OPERAND READ, SHORT FORMAT

```
LOC.  INST.   LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

F00           F00$ ORG
              *THE AREA BETWEEN F00 AND FFF CAN BE ADDRESSED BY THE LEAST SIGNIFICANT
              *8 BITS OF THE LINK REGISTER,INDEXED LOGICALLY BY F00$.
              *THIS AREA IS DESIGNATED AS P1,AND IS USED FOR PRE-PROCESSING SUCH AS
              *COMPUTING OPERAND ADDRESSES ETC.
              *
              *                                              ORSI COMPUTES THE OPERAND ADDRESS
              *                                              OF SHORT FORMAT INSTRUCTIONS
              *                                              (REL TO I) AND INITIATES THE
              *                                              READ OF THE OPERAND
              *
F00 20530040 ORSI COPY D  S            SE,                   EXTEND SIGN OF DISP
F02 20220050      COPY L  L            R8                    SHIFT OPERAND EXECUTION ADDRESS
              *                                              INTO LOW 8 OF LINK
F04 41347880      ADD  S  M  I         MR,J                COMPUTE OPERAMD ADDR.,FETCH
              *                                              OPERAND AND JUMP TO
              *                                              ADDRESS CONTAINED IN LINK.
              *
              *ORS1,ORS2 AND ORS3 ARE IDENTICAL TO ORSI
              *EXCEPT FOR INDEX REGISTER USED TO
              *COMPUTE THE EFFECTIVE ADDRESS (EA) OF
              *THE OPERAND
F06 20530040 ORS1 COPY D  S            SE,
F08 20220050      COPY L  L            R8,
F0A 49A43800      ADD  1  M  S         MR,J
F0C 20530040 ORS2 COPY D  S            SE,
F0E 20220050      COPY L  L            R8,
F10 49B43800      ADD  2  M  S         MR,J
F12 20530040 ORS3 COPY D  S            SE,
F14 20220050      COPY L  L            R8,
F16 49C43800      ADD  3  M  S         MR,J
              *
```



The instruction in the Memory Data register, D , is now copied into a Scratch register, S , with sign extension specified.

$D = 9210_{16}$ $S = 0010_{16}$. S now contains only the displacement.

The exit to the operation subroutine is prepared. The Link register, $L = 440C_{16}$, is shifted right eight places, and the new contents of $L = 0044_{16}$, are the address of the Subtract subroutine.

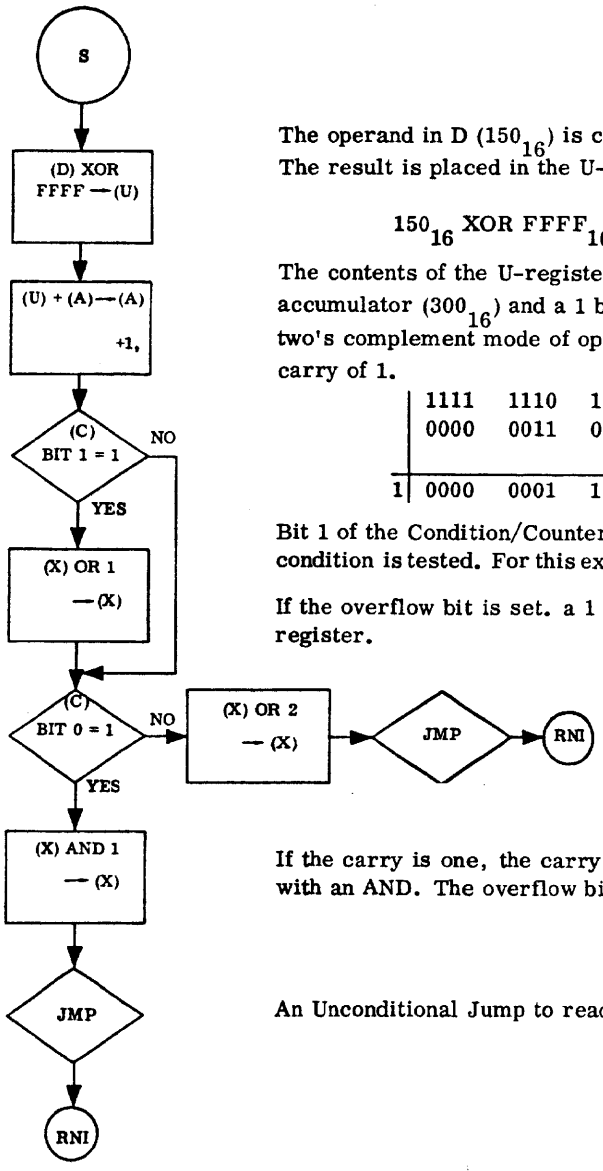
Index Register 2 = 510_{16} is added to $S = 10_{16}$ giving 520_{16} as the effective core address of the operand. This address is placed in the Memory Address register and a Read of the operand is initiated.

A Jump-to-LOC 000 indexed by the contents of the Link register results in an effective Jump-to-LOC 044 of ROM for execution of the Subtract when the operand becomes available in the Memory Data register, D.

SUBTRACT

```
LOC.  INST.   LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

044 B456FFFF S     XORI D  U      FFFF$          1'S COMP DATA
046 4C644081       ADD  U  A  A         +1       ADD WITH PLUS 1 = SUB
048 0010104C SOV   BRZ  C     1    *+2           BR IF NO OVERFLOW
04A AC880001       ORI  X  X      1$            SET OVFL INDICATOR
04C 00100052       BRZ  C     0    *+3           BR IF CARRY = 0
04E 9C880001       ANDI X  X      1$            CLEAR CARRY INDICATOR
050 000001BC       JMP            RNI
052 AC880002       ORI  X  X      2$            SET CARRY INDICATOR
054 000001BC       JMP            RNI
```



The operand in D (150_{16}) is complemented by an Exclusive OR with all 1 bits. The result is placed in the U-register:

$$150_{16} \text{ XOR } FFFF_{16} = FEAF_{16}.$$

The contents of the U-register ($FEAF_{16}$) are added to the contents of the accumulator (300_{16}) and a 1 bit is forced into the carry-in position for the two's complement mode of operation: $FEAF_{16} + 150_{16} + 1 = 1B0_{16}$ with a carry of 1.

	1111	1110	1010	1111	FEAF
	0000	0011	0000	0000	300
				1	1
1	0000	0001	1011	0000	

Bit 1 of the Condition/Counter register, C, contains the overflow status. This condition is tested. For this example.

If the overflow bit is set. a 1 bit is forced into the simulated 1130 status register.

Bit 0 of the C-register contains carry status. If the META 4 carry is a zero, the carry bit of the simulated Status register is forced to a 1 and the overflow bit (bit 15) is unchanged. In this example, the META 4 carry is a 1 and control transfers to the path marked "yes." Note that emulation of the subtract with the complement and add in the META 4 results in the 1130 subtract carry condition being the inverse of that which occurs in the META 4.

If the carry is one, the carry bit (bit 14) of X and all the unused bits are cleared with an AND. The overflow bit (bit 15) is unchanged.

An Unconditional Jump to read the next instruction terminates the instruction.

Multiply and divide instructions for the IBM 1130 emulation are not flowcharted, but are listed together with comments. The Instruction and Operand Fetch routines are identical to those for subtract. The actual multiply and divide arithmetic instructions are a relatively small portion of the code, even though execution of the loop 16 times under control of the counter represents a large portion of the execution time. The coding is primarily internal housekeeping, tests to emulate certain failure modes of the IBM 1130, conversion of the operands to positive numbers for arithmetic processing, and establishing the correct arithmetic processing, and establishing the correct arithmetic sign of the answer.

MULTIPLY

```
LOC.  INST.   LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

080 A0010010 M     LDI      C      10$                    LOAD 16 INTO COUNTER
082 A0020110       LDI      L      MTOP
084 08480100       BNZ  A        0  MNEG                   BR IF MULTIPLIER NEG
086 2C450210       COPY A   Q             SO,R1           COPY MULTIPLIER INTO Q,RIGHT
                   *                                      SHIFTED 1 PLACE TO CONDITION
                   *                                      A SUBSEQUENT MULTIPLY STEP
088 00000104       JMP              MNEG+2
                   *
                   *
                   *MNEG IS A CONTINUATION OF THE MULTIPLY
                   *ROUTINE IN P2 AREA
100 BC45FFFF MNEG  XORI A   Q       FFFF$                  COMP MULTIPLIER TO(Q)
102 4C550211       ADD  Q   Q   0          +1,R1,SO,       +1 FOR 2'S COMP
                   *                                      ALSO SHIFT MULTIPLIER
                   *                                      TO CONDITION A SUBSEQUENT
                   *                                      MULTIPLY STEP
104 30534080       XOR  D   S   A                         SIGN BIT OF S IS SIGN OF PROD
106 A4040000       LDI      A      0$
108 0058010C       BNZ  D        0  *+2                  BR IF DATA NEG
10A 24560800       COPY D   U             J               POS MULTIPLICAND
10C B456FFFF       XORI D   U     FFFF$                   COMP MULTIPLIER
10E CC660001       ADDI U   U     1$
110 5C644290 MTOP  MULT U   A   A          SO,R1,          SHIFT AND ADD
                   *                                      CONDITIONED BY PREVIOUS CONTENTS
                   *                                      OF THE SHIFT FLIP-FLOP
112 2C550F10       COPY Q   Q             SO,R1,SI,D,J,   FORM LEAST
                   *                                      SIGNIFICANT PORTION OF RESULT,
                   *                                      ALSO SHIFT NEXT BIT OF MULTIPLIER
                   *                                      TO SHIFT FLIP-FLOP THEN LOOP
                   *                                      UNTIL COMPLETION
114 003001BC       BRZ  S        0  RNI                  EXIT IF POS
116 BC55FFFF       XORI Q   Q     FFFF$                   COMP RESULT
118 BC44FFFF       XORI A   A     FFFF$
11A CC550001       ADDI Q   Q     1$
11C 4C440002       ADD  A   A   0          CI
11E 000001BC       JMP              RNI
```

DIVIDE

```
LOC.  INST.   LAB.  OP   BR DR AR OPRAND MODIFIERS AND COMMENTS

08A A0010010 D     LDI      C      10$                    LOAD 16 INTO COUNTER
08C A0020134       LDI      L      DTOP                   TOP OF LOOP TO LINK
08E 28430000       COPY A   S                             SIGN OF DIVIDEND TO S
090 08480120       BNZ  A        0 DNEG                   BR IF DIVIDEND NEG
092 005C012A       BNZ  D          DNEG+5 W               BR IF DIVISOR NOT ZERO
094 AC880001 OVFL  ORI  X    X     1$                     SET OVFL BIT
096 000001BC       JMP             RNI

                                                          DNEG IS A CONTINUATION OF THE DIVIDE
                                                          ROUTINE IN P2 AREA
120 00540094 DNEG  BRZ  D          OVFL   W               BR IF DIVISOR ZERO
122 BC44FFFF       XORI A    A     FFFF$                  COMP DIVIDEND
124 BC55FFFF       XORI Q    Q     FFFF$
126 CC550001       ADDI Q    Q     1$
128 4C440002       ADD  A    A  0         CI
12A 24560000       COPY D    U                            SAVE MEMORY DATA
12C 34396080       XOR  S    O  U                         PRODUCT OF SIGNS TO O
12E 08680134       BNZ  U          DTOP   0$             BR IF DIVISOR NEGATIVE
130 BC66FFFF DNZ   XORI U    U     FFFF$                  COMPLEMENT DIVISOR TO
132 CC660001       ADDI U    U     1$                     PERFORM SUBTRACT
134 6C644280 DTOP  DIV  U    A  A       SO,              TRIAL SUBTRACT OR
                   *                                     SUBTRACT
136 2C550320       COPY Q    Q          L1,SO,SI          SHIFT DIVIDEND
                   *                                     BIT IN AND SHIFT DIVIDEND BIT OUT
138 2C440D20       COPY A    A          L1,SI,D,J,        SHIFT DIVIDEND
                   *                                     AND LOOP TO COMPLETION
13A 6C644280       DIV  U    A  A       SO,               LAST CYCLE REM NOW OK
13C 2C560320       COPY Q    U          L1,SO,SI          COMPLETE QUOT TO L
13E 00102094       BRZ  C        2 OVFL                   SHIFT FF =0 = OVFL
140 00300146       BRZ  S          *+3    0$             BR IF REM. POS.
142 BC44FFFF       XORI A    A     FFFF$
144 CC440001       ADDI A    A     1$
146 2C450000 RPOS  COPY A    Q                            REMAINDER TO Q
148 08680154       BNZ  U          UPOS   0$             QUOT IS POS.
14A 08900094 UNEG  BRZ  O          OVFL   0$             IF QUOT IS NEG AND SIGN OF
                   *                                     QUOT. IS POS,IT IS AN OVERFLOW
14C CC698001       SUBI U    O     7FFF$                  IF THE FORMED QUOT
                   *                                     IS NEGATIVE AND THE SIGN OF THE
                   *                                     QUOTIENT IS NEGATIVE AN OVERFLOW
                   *                                     IS INDICATED EXCEPT FOR
                   *                                     -2 TO THE 15TH
14E 089C0094       BNZ  O          OVFL   W               NOT -2 TO 15TH
150 CC640001 QNEG  ADDI U    A     1$                     CONVERT 1'S COMP QUOT
                   *                                     TO 2'S COMP QUOTIENT
152 000001BC       JMP             RNI                    EXIT
154 08980150 UPOS  BNZ  O          QNEG   0$             SIGN OF QUOT IS POSITIVE
156 BC64FFFF       XORI U    A     FFFF$                  QUOTIENT TO A
158 000001BC       JMP             RNI                    EXIT
```

APPENDIX C

POWERS OF TWO

APPENDIX C

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25

APPENDIX D

HEXADECIMAL-TO-DECIMAL CONVERSION TABLE

APPENDIX D

HEXADECIMAL-TO-DECIMAL CONVERSION TABLE

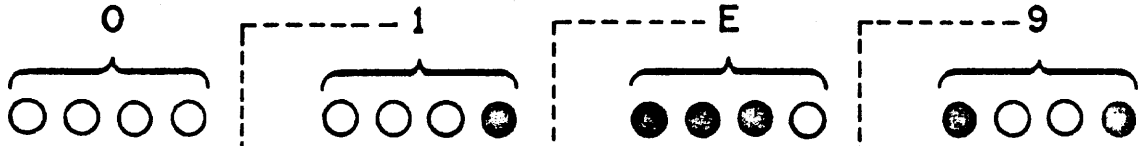
The table in this appendix provides for direct conversion of decimal and hexadecimal numbers in these ranges:

Hexadecimal	Decimal
000 to FFF	0000 to 4095

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

Hexadecimal	Decimal
1000	4096
2000	8192
3000	12288

Hexadecimal	Decimal
4000	16384
5000	20484
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	53248
E000	57344
F000	61440



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

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

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

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

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

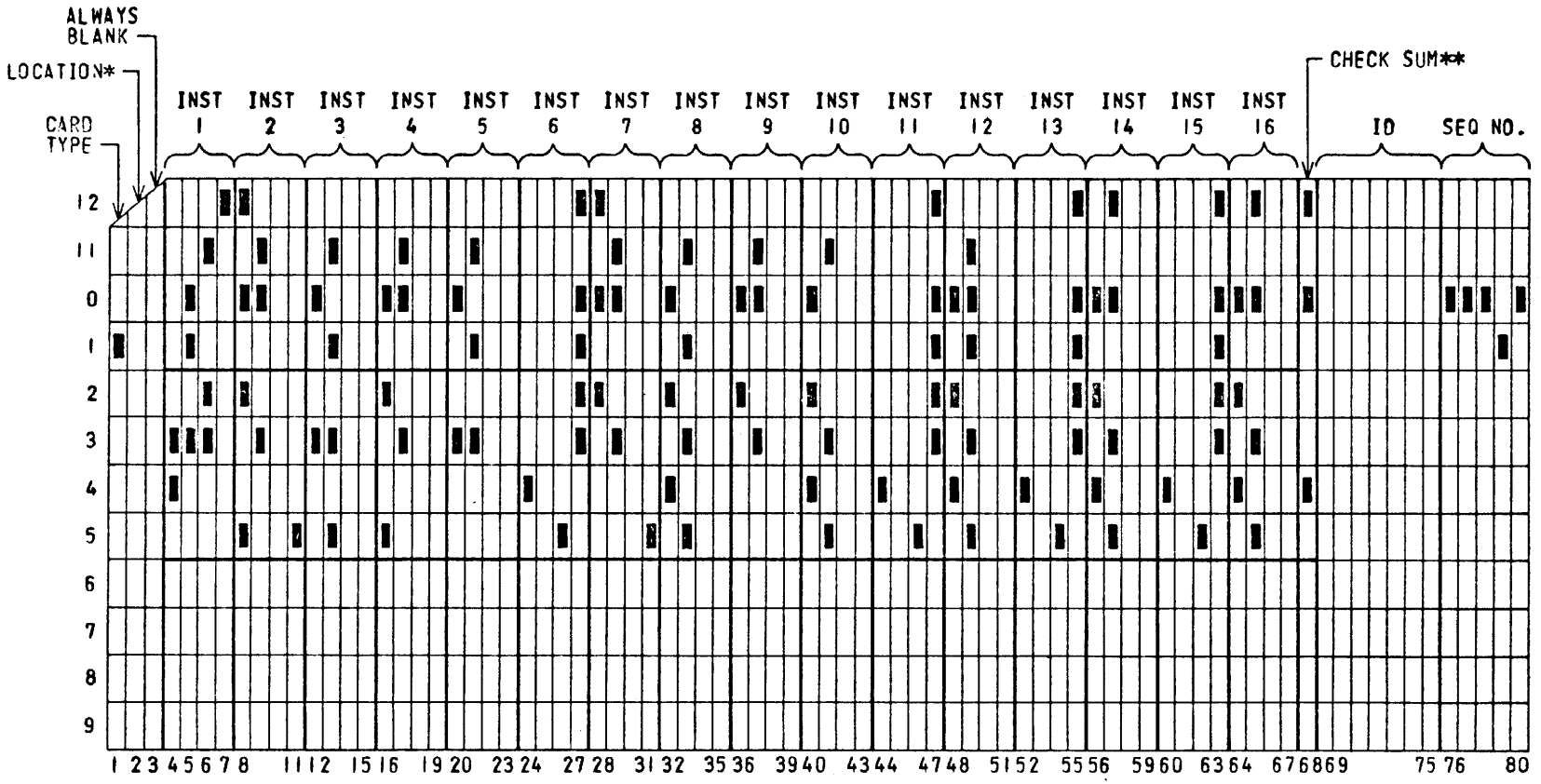
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610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

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

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

APPENDIX E

META 4 SYSTEM OBJECT CARD FORMAT



INSTRUCTIONS ARE READ:

2 CHARACTERS PER CARD COLUMN, WITH 4 COLUMNS PER ROM WORD. THE MOST SIGNIFICANT BITS ARE CARD BIT LOCATIONS 12 AND 2.

THIS CARD IS READ AS FOLLOWS:

TYPE 1, LOCATION 00, SEQ NO. 00010

INST NO.		INST NO.	
1	06344CC0	9	28640000
2	A9640001	10	2A450000
3	24550000	11	020001BC
4	29640000	12	2A750000
5	24540000	13	020001BC
6	020001BC	14	2AA50000
7	A8640001	15	020001BC
8	2A550000	16	2AB50000

* LOCATION IS THE 2 MOST SIGNIFICANT DIGITS OF THE ROM LOCATION WHEN THE CARD STARTS. THE LEAST SIGNIFICANT DIGIT WILL ALWAYS BE ZERO.

** CHECK SUM IS THE BINARY SUM OF COLUMNS 3-67.

APPENDIX F

MICROASSEMBLER OPERATION

APPENDIX F

MICROASSEMBLER OPERATION

•The Microassembler is stored on the META 4 System disc under the name of M4ASM. The following cards are needed to load and execute M4ASM:

```
//bJOB
//bXEQ M4ASM
```

These cards are followed by the source deck to be assembled.

MICROASSEMBLER THEORY

•The META 4 System Microassembler operates in a two-pass mode. On the first pass, a label table and a register table are stored in core. Any errors encountered during pass one are flagged and printed on the selected listing device.

On pass two, instructions are read back from the disc and actual assembly takes place. Listing occurs during this pass and any errors found are flagged and listed. Object output is generated during this pass and is punched on the selected output device.

When the Microassembler is loaded, the following message is printed on the console printer:

Select mode with console entry switches

•SW-0	Paper Tape Source
•SW-1	Paper Tape Object
•SW-2	List on Console Printer
•SW-13	List Type-2 Comments
•SW-14	List Type-3 Comments

Press start.

In the normal operation, with no switches set, source input is from cards, object output is on cards, and listing is on the line printer. Deviations from this configuration may be made by selecting the proper console entry switches. Once selections are made and the START switch is pressed, switches may be reset. At the end of pass two when assembly, output, and listing are completed, the program exits back to the Monitor.

MICROASSEMBLER ERROR CODES

•Any errors found during pass one are flagged and printed before the program listing starts. Any errors found during pass two are flagged on the listing under the column labeled ERR. Some of the errors found in pass one will be encountered again in pass two; therefore, they will be flagged twice, once before the listing and again on the listing.

The Microassembler converts mnemonics, labels, and constants into bit patterns for the META 4 ROM.

Microassembler Error codes are listed in Appendix G.

FIRMWARE CODING MICROCODING

•Coding may be done on the Digital Scientific Firmware Assembler Coding Form. This form provides the proper column and field identification for the META 4 System language. A sample of the coding form is shown on the following page. The fields are described below.

Label Fields

•Columns 1-4 contain an absolute location, a label, a comment indicator, or spaces. Absolute locations are represented in even hexadecimal notation, left-justified within the field and followed by a $. Labels are left-justified within the field and are terminated by a space. Labels may contain any combination of four alphanumeric characters.

Operand Field

•Columns 6-9 contain an operation mnemonic or pseudo-op, left-justified within the field. (Operation mnemonics and pseudo-ops are described in Section 2.0 of this manual.)

Source and Destination Field

•Columns 11-18 contain the source and destination registers to be used in the instruction:

- •Columns 11-12 B Source
- •Columns 14-15 D Destination
- •Columns 17-18 A Source or Branch Pointer

Operand Field

•Columns 20-25 are used as a data field for BR and RI format instructions and must contain a label or a hexadecimal constant. Labels and constants are left-justified within the field and are terminated by a space or a $. A hexadecimal constant must be followed by a $.

Modifiers and Comments Field

•Columns 27-72 are used for modifiers and comments. Modifiers are added to instructions in order to provide variations to the basic instruction set. Modifiers must start in column 27 and may be listed in any order, separated by commas (,). The first space encountered after column 26 indicates that the rest of the field is dedicated to comments.

Identifier Field

•Columns 73-80 are used for statement identifier or sequence numbers. This field is listed just as it appears on the source cards. The Microassembler takes no action on this field.

Digital Scientific META 4 T.M. COMPUTER
FIRMWARE ASSEMBLER CODING FORM

PROGRAM _____ DATE _____

PROGRAMMED BY _____ SHEET _____ OF _____

LABEL	OPERATION	B REG	D REG	A REG	OPERAND	MODIFIERS AND COMMENTS	IDENT
1 2 3 4	5 6 7 8 9	10 11 12	13 14 15	16 17 18	19 20 21 22 23 24 25	26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	73 74 75 76 77 78 79 80

DSC8A 12/69

MICROASSEMBLER CODING FORM

REGISTER DESIGNATION

•Before any register can be used in a program, it must be named. This is done with an EQUR statement.

COMMENTS

•Three types of comment statements are available:

- •Type 1 has an asterisk (*) in column 1
- •Type 2 has a comma (,) in column 1
- •Type 3 has a period (.) in column 1

All three types have columns 2-80 available for comments. Any printable character may be used in comment statements.

Only type 1 comments appear on a normal program listing. Types 2 and 3 are suppressed unless specifically selected at the beginning of the assembly. Selection of a type 2 or 3 comment may be made during manual mode selection. If switch 13 on the programmer's control panel is set, type 2 comments are printed. If panel entry switch 14 is set, type 3 comments are printed.

Comments may also be written in columns 28-72. These comments will always be printed and may contain any combination of alphanumeric or special characters.

APPENDIX G

META 4 MICROASSEMBLER ERROR CODES

APPENDIX G

META 4 MICROASSEMBLER ERROR CODES

•Microassembler errors are flagged according to the following table:

A- ILLEGAL A-REGISTER

B- ILLEGAL B-REGISTER

D- DUPLICATE SYMBOL

F- FORMAT ERROR

H- HEX CONVERSION ERROR

I- ILLEGAL D-REGISTER

M- ILLEGAL MODIFIER

O- INVALID OPERATION

R- UNDEFINED REGISTER

S- SYMBOL TABLE OVERFLOW

U- UNDEFINED SYMBOL

APPENDIX H

CABLE CONNECTIONS

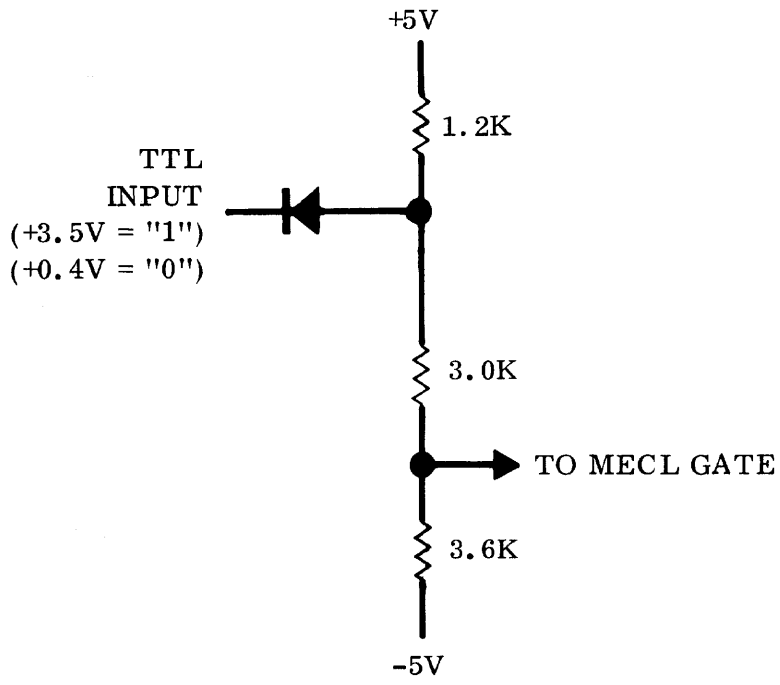
APPENDIX H

CABLE CONNECTIONS

•A list of signals on each 44-pin connector of an I/O port is shown on the following page. The mating connector is a Viking 2VH22/1JN5.

Output signals (4 control flip-flop outputs and 16 data outputs) are supplied from Motorola type 1039 MECL-to-TTL integrated circuit level shifters.

Input signals (2 Ready signals, Output-Enable, and 16 data inputs) are each connected to a level shifter network shown schematically below.



In the following list, the logic terms are TTL with 3.5 volts high and +0.4 volts low. The active state (logical one) listed for each signal is low.

PIN	SIGNAL	PIN	SIGNAL
A1	GND	B1	Not Used
A2	Input Ready	B2	Reserved
A3	Output Resume	B3	$\overline{\text{Acknowledge}}$
A4	Output Enable	B4	Reserved
A5	Reserved	B5	$\overline{\text{Go}}$
A6	$\overline{\text{Input Bit 13}}$	B6	Output Bit 13
A7	$\overline{\text{Input Bit 14}}$	B7	$\overline{\text{Output Bit 14}}$
A8	$\overline{\text{Input Bit 12}}$	B8	$\overline{\text{Output Bit 12}}$
A9	$\overline{\text{Input Bit 0}}$	B9	$\overline{\text{Output Bit 0}}$
A10	$\overline{\text{Input Bit 1}}$	B10	$\overline{\text{Output Bit 1}}$
A11	$\overline{\text{Input Bit 2}}$	B11	$\overline{\text{Output Bit 2}}$
A12	$\overline{\text{Input Bit 15}}$	B12	$\overline{\text{Output Bit 15}}$
A13	$\overline{\text{Input Bit 3}}$	B13	$\overline{\text{Output Bit 3}}$
A14	$\overline{\text{Input Bit 11}}$	B14	$\overline{\text{Output Bit 11}}$
A15	$\overline{\text{Input Bit 7}}$	B15	$\overline{\text{Output Bit 7}}$
A16	$\overline{\text{Input Bit 10}}$	B16	$\overline{\text{Output Bit 10}}$
A17	$\overline{\text{Input Bit 6}}$	B17	$\overline{\text{Output Bit 6}}$
A18	$\overline{\text{Input Bit 9}}$	B18	$\overline{\text{Output Bit 9}}$
A19	$\overline{\text{Input Bit 5}}$	B19	$\overline{\text{Output Bit 5}}$
A20	$\overline{\text{Input Bit 4}}$	B20	$\overline{\text{Output Bit 4}}$
A21	$\overline{\text{Input Bit 8}}$	B21	$\overline{\text{Output Bit 8}}$
A22	Not Used	B22	GND

APPENDIX I

MICROPROGRAMMER'S PANEL

APPENDIX I

MICROPROGRAMMER'S PANEL

•The Digital Scientific Microprogrammer's Panel is used only to operate the META 4 Processor's microprogram in a step-by-step fashion for firmware or hardware debugging. It is not required for normal operation of the system.

The panel data lamps display binary data from internal logic signals and the panel switches select operating modes and provide entry data.

Any machine instruction can be executed from the panel's data switches. This capability allows loading or viewing any of the 32 machine registers, viewing any of the 4096 read-only memory (ROM) data words, and executing data transfers with any input or output device or with core memory.

Control Switches located at the bottom of the Microprogrammer's Panel are:

•START	Forces execution of one instruction cycle.
•RUN/STEP	Disables timing clocks after each instruction cycle when STEP is selected; allows continuous operation when RUN is selected.
•CLEAR	Forces the ROM address register to zero and resets the control flip-flop on memory and I/O register boards.
•LOAD	Loads the ROM data register from the 32 data switches and inhibits ROM data. The Up position is latching; the Down position is momentary; and the Center position is OFF.
•ON/OFF	Enables/disables the other control switches.

The D/ADDR/ROM switch at the top of the panel selects the function of the 32 data indicators. The D-bus together with the ROM address register is displayed in the lower position.

MICROPROGRAMMER'S PANEL

The ROM address displayed when D/ADDR is selected is the next instruction to be executed and not the current instruction.

The Control Indicators at the bottom of the panel are:

•INPUT	Indicates that an input instruction with the PZ bit is waiting for an external signal.
•OUTPUT	Indicates that an output instruction with the PZ bit is waiting for an external signal.
•PANEL	Indicates that the instruction is waiting for the START switch to be depressed.
•LOAD	Indicates the second cycle of a register load will control ROM address selection.
•JUMP	Indicates that a Branch in the program sequence will occur.
•INSTR	Indicates that a successful Branch test will use part of the current instruction word to form the Branch address.
•LINK	Indicates that a successful Branch test will use the link contents to form part of the Branch address.
•SHIFT	Displays bit 2 of Register 1.
•CARRY	Displays bit 0 of Register 1.
•OVERFLOW	Displays bit 1 of Register 1.

REGISTER DATA ENTRY

•Data may be entered into a register in STEP mode by setting an ORI instruction into the data switches with the PZ/MW and IO/MR switches set to zero, the B-bus address set to zero, the D-bus address set to register to be loaded, and the data to be entered in the ODD word of the instruction. After the data switches are set, depress LOAD to enter the instruction into the ROM data register and then depress START to execute the instructions which enter data into the register.

REGISTER DATA DISPLAY

•Register data may be displayed in the STEP mode by setting an ORI instruction into the data switches with the PZ/MW and IO/MR switches set to zero, the B-bus address set to the register to be displayed, and the ODD word of the instruction set to zero. After the data switches are set, depress LOAD to enter the instruction into the ROM data register and read the data with the D/ADDR/ROM switch set to D/ADDR.

ROM INSTRUCTION OR DATA DISPLAY AND PROGRAM START

•A ROM double-word may be displayed by entering a BRZ instruction into the data switches with all bits zero except for the address of the ROM double-word in the ODD word of the instruction. After the data switches are set, depress LOAD to enter the instruction into the ROM data register and then depress START to display the contents of the double word with the D/ADDR/ROM switch set to ROM.

Each additional depression of the START switch will execute one instruction beginning with the instruction first displayed. For high-speed execution, place the RUN/STEP switch in the RUN mode.

COMMENT SHEET

DIGITAL SCIENTIFIC META 4[T.M.] SERIES 16 COMPUTER SYSTEM

REFERENCE MANUAL

Publication No. 7032MO

FROM: Name: _____

Business Address: _____

COMMENTS: (Describe errors, suggested additions or deletions, etc.; please include page number.)

Cut Along Line

Form No. DSC7

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.
FOLD ON DOTTED LINES AND STAPLE

Staple

Staple

Fold

Fold

First Class
Permit No. 6225
San Diego, Calif.

BUSINESS REPLY MAIL
No Postage Stamp Necessary If Mailed in the United States

— POSTAGE WILL BE PAID BY —

DIGITAL SCIENTIFIC CORPORATION
11455 Sorrento Valley Road
San Diego, California 92121

Attention: Marketing Department, Publications Group

Cut Along Line

Fold

Fold

Staple

Staple