3

system operation

PROGRAMMER'S CONSOLE OPERATION

The switches and indicators on the PDP-8/E programmer's console are designed to allow manual control over the detailed operation of the computer and present a convenient indication of program conditions within the machine. The PDP-8/E may be programmed manually, by means of switches on the programmer's console, and program execution may be started, stopped, monitored, or toggled between various modes of operation. The console switches also provide a convenient means of selecting a memory location for examination and selectively modifying the content of memory.

The indicator lights on the programmer's console provide a continuous display of the logical state of major registers, busses and control flip-flops inside the PDP-8/E, as well as several important registers contained in commonly used processor options, such as the extended arithmetic element. Figure 3-1 shows the KC8-EA Programmer's Console, which is typical of the models available. Table 3-1 describes the function of the various switches and indicators. This table is intended as a reference for the advanced programmer or system operator; most users will want to be thoroughly familiar with the remainder of this chapter before attempting to operate the programmer's console.

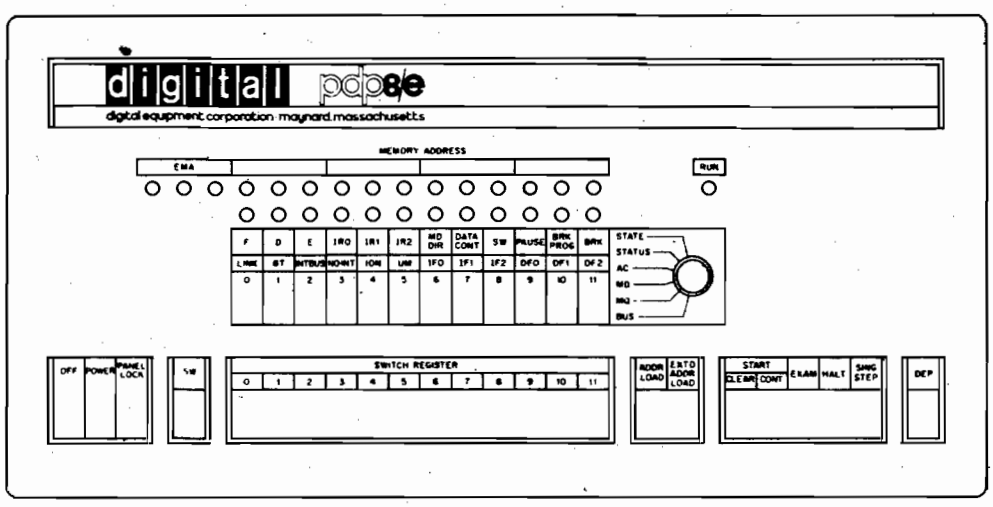


Figure 3-1. PDP-8/E Programmer's Console

Table 3-1 Programmer's Console Control and Indicator Functions

CONTROL OR INDICATOR	FUNCTION
OFF/POWER/ PANEL LOCK	In the counter-clockwise, or OFF position, this key operated switch disconnects all primary power to the computer. In the POWER, or vertical position, it applies power to the computer and all manual controls. In the PANEL LOCK, or clockwise position, it applies power to the computer, the switch register, the SW switch and the RUN indicator only. In this position, a running program is protected from inadvertent switch operation.
SW	When the SW switch is up, the OMNIBUS SW line is disabled (logical 1, or voltage level high). When it is down, the SW line is asserted. This switch is used by certain peripheral options such as the MI8-E Bootstrap Loader.
ADDR LOAD	Pressing the ADDRess LOAD switch loads the contents of the SR into the CPMA register and enables the FETCH major state for the next processor cycle (which will begin when the RUN indicator is lit).
EXTD ADDR LOAD	Pressing the EXTendeD ADDRess LOAD switch loads the content of SR bits 6-8 into the instruction field register and the content of SR bits 9-11 into the data field register. The instruction and data field registers are contained in the KM8-E Memory Extension and Time Share option.
CLEAR	Pressing the CLEAR switch generates an INITIALIZE pulse that loads a binary 0 into bits 0-11 of the AC, the link, all I/O device flag registers and all interrupt system flip-flops. This is equivalent to executing a CAF instruction.
CONT	Pressing the CONTinue switch sets the RUN flip-flop and issues a MEM START L signal to begin program execution at the memory location addressed by the current content of the CPMA register.
EXAM	Pressing the EXAMine switch loads the contents of the memory location addressed by the current content of the CPMA register into the MB register and then increments the CPMA and PC registers. Repeated operation of this switch permits the content of sequential memory locations to be examined.

HALT

Pressing the HALT switch clears the RUN flip-flop, causing the computer to stop at the beginning of the next FETCH cycle. Operating the computer with the HALT switch depressed causes one complete instruction to be executed whenever the CONTinue switch is pressed.

SING STEP

Pressing SINGle STEP clears the RUN flip-flop and causes the computer to stop at TS1 of the next machine cycle. Operating the computer with the SINGle STEP switch depressed causes one machine cycle to be executed whenever the CONTinue switch is pressed.

DEP

Lifting the spring-loaded DEPosit switch loads the content of the SR into the MB register and into memory at the address specified by the current content of the CPMA register, then increments the CPMA and PC registers. Use of the DEPosit switch facilitates manual storage of information in sequential memory locations.

EMA

The Extended Memory Address register displays the content of the 3-bit EMA bus (EMA0-2) contained on the OMNIBUS. EMA0-2 normally carries the memory field designation of the memory field being accessed.

MEMORY ADDRESS

The MEMORY ADDRESS register displays the content of the 12-bit MA bus (MA0-11) contained on the OMNIBUS. It combines with the EMA register to provide the 15-bit address of the next memory location to be accessed.

RUN

The RUN indicator lamp is lighted to show that the RUN flip-flop is set, and all machine circuits are activated and capable of executing instructions.

Indicator Selector Switch

This 6-position rotary switch designates which of six possible registers (or combinations of registers) is to be gated into the adjacent 12-bit display. Setting the Indicator Selector Switch to:

BUS

Displays the logical state of the 12-bit DATA bus (DATA0-11) contained on the OMNIBUS.

MQ

Displays the content of the Multiplier Quotient register.

MD

Displays the logical state of the 12-bit MEMORY DATA bus (MD0-11) contained on the OMNIBUS. This bus normally carries the content of the last memory location addressed by the EMA and MEMORY ADDRESS registers.

AC	Displays the content of the accumulator.
STATUS	Each display lamp is lighted to indicate the designated condition:

INDICATOR LAMP/BIT POSITION	TURNED ON TO INDICATE:
0	The link is set.
1	The Greater Than Flag (GTF) is set. The GTF is contained in the KE8-E Extended Arithmetic Element.
2	The OMNIBUS interrupt request line is asserted.
3	The interrupt inhibit flip-flop is set. The interrupt inhibit flip-flop is contained in the KM8-E Memory Extension and Time Share option.
4	The interrupt system is enabled.
5	The USER MODE line is asserted. Signal USER MODE L originates in the time share portion of the KM8-E Memory Extension and Time Share option to disable execution of all OSR, LAS, IOT and HLT instructions when the computer is operated in a timesharing environment.
6-8	Displays the content of the 3-bit instruction field register (IF0-2) contained in the KM8-E Memory Extension and Time Share option.
9-11	Displays the content of the 3-bit data field register (DF0-2) contained in the KM8-E Memory Extension and Time Share option.

STATE	Each display lamp is lighted to indicate the designated condition:

INDICATOR LAMP/BIT POSITION	TURNED ON TO INDICATE:
0	FETCH major state is enabled.
1	DEFER major state is enabled.
2	EXECUTE major state is enabled.
3-5	Displays the content of the 3-bit instruction register (IR0-2).

6	Displays the state of the MD DIR line on the OMNIBUS. Signal MD DIR is high (and the lamp is lighted) during operations that read data from memory. MD DIR is low (and the lamp is extinguished) during operations that write data into memory.
7	Displays the state of the BREAK DATA CONT line on the OMNIBUS. Signal BREAK DATA CONT is low (and the lamp is lighted) during an ADM operation.
8	The SW line on the OMNIBUS is asserted. This can only occur when the programmer's console SW switch is depressed.
9	The PAUSE line on the OMNIBUS is asserted. Signal I/O PAUSE L is generated during IOT instruction execution.
10	The BREAK IN PROG line on the OMNIBUS is asserted, indicating that one or more devices are requesting a data break. The highest priority device will begin a DMA operation at the beginning of the following cycle.
11	The BREAK CYCLE line on the OMNIBUS is asserted, indicating that the processor is currently performing a DMA operation under the control of a peripheral device.

MEMORY ORGANIZATION

PDP-8/E memory is divided into 4096-word blocks called memory fields. The memory fields are numbered sequentially from field 0, which is the first 4096 words of memory supplied with the basic system, up to field 7, if a full 32K of memory is installed. Within each memory field, the 4096 storage locations are numbered sequentially, in octal, from 0000 to 7777. This 4-digit octal number is called the 12-bit address of the memory location. In any given memory field, every storage location has a unique 12-bit address.

Each memory field is further subdivided into 32 pages of 128 words each. Memory pages are numbered sequentially, in octal, from page 0 (which contains addresses 0000 to 0177) to page 37 (addresses 7600 to 7777). Within each memory page, the 128 locations on the page are numbered sequentially, in octal, from 0 to 177. This number is called the page address of the memory location. Page addresses are not redundant; the page address of a memory location is simply the octal value of the low-order 7 bits of the 12-bit address.

The first five bits of a 12-bit memory address are called the page bits. The octal value of the page bits for any memory address is identical to the number of the memory page on which the address is located. The last seven bits of the 12-bit address are called the page address bits.

The octal value of the page address bits for any memory address is identical to the page address of the memory location. Thus, location 4716 is at page address 116 on page 23, while location 2257 is at page address 057 on page 11.

Unlike memory fields, which may be physically separated by being located on different modules plugged into the OMNIBUS, memory pages do not correspond to any physical separation within memory. The computer has no way of recognizing which page of memory it is executing

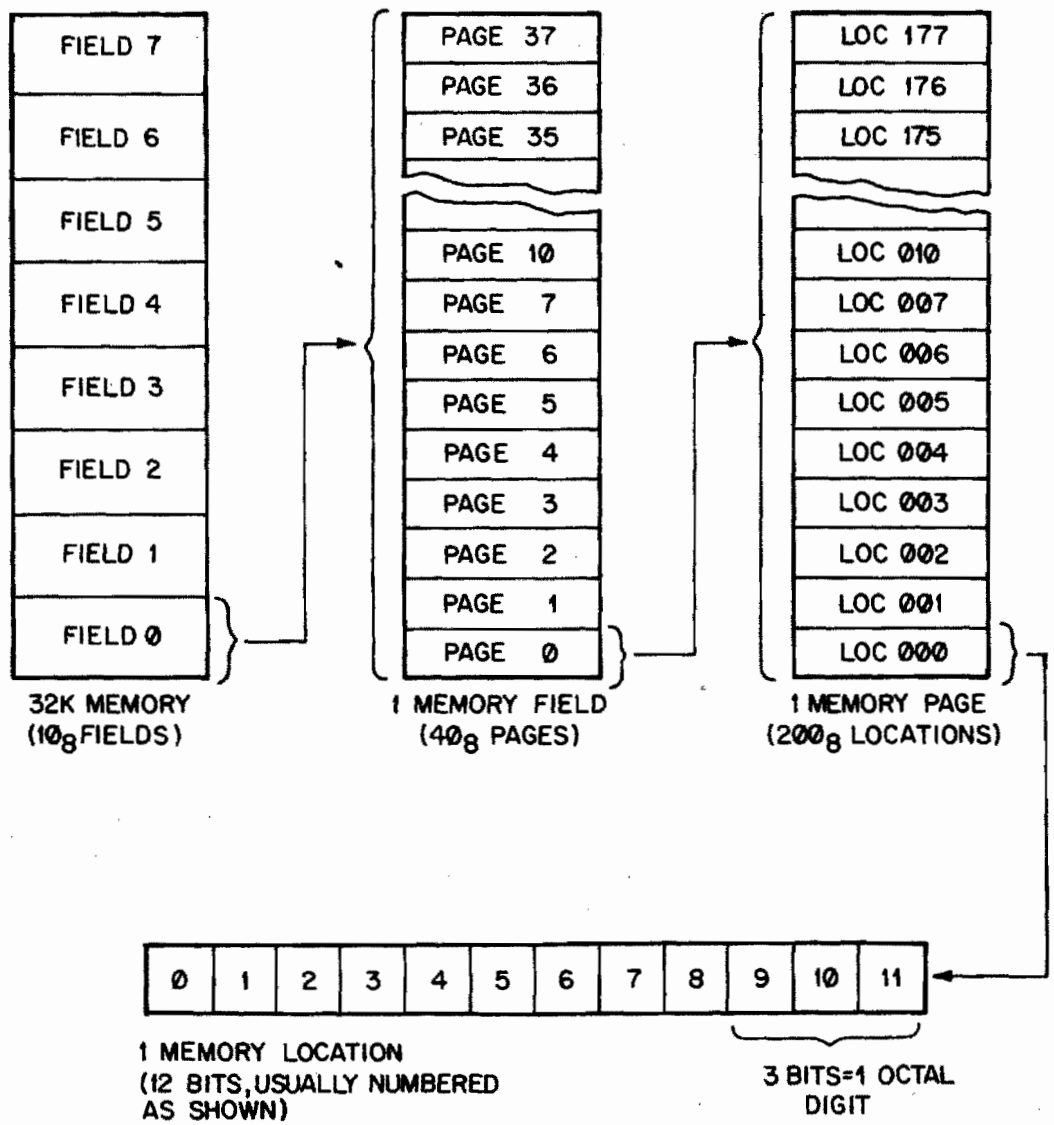


Figure 3-2. PDP-8/E Memory Organization

in, and it is not cognizant of executing across a page boundary. Memory pages represent a more or less artificial subdivision of memory that facilitates understanding the PDP-8/E memory reference instruction decoding process.

The individual bits of a PDP-8/E memory word are usually numbered, for reference purposes, as shown in Figure 3-2. The bits of major registers are numbered in the same manner, but the abbreviated register name is prefixed to the number for identification purposes. Thus, bit 0 is always the high-order bit of a memory word, while AC0 is the high-order bit of the accumulator and PC11 is the low-order bit of the program counter.

MEMORY AND PROCESSOR INSTRUCTIONS

A PDP-8/E instruction is a single, 12-bit word, stored in memory, that tells the computer to perform a specific operation or sequence of operations. Like most stored program computers, the PDP-8/E makes no distinction between instructions and data; it will manipulate instructions as though they are stored variables or attempt to execute data as instructions if it is programmed to do so. The 12-bit value that tells the computer to execute a specified instruction is called the octal code for that instruction. In addition to its unique octal code, every instruction has an assigned mnemonic, which is a 3- or 4-character name that may be supplied to an assembler program to generate the corresponding octal code. There are three general classes of PDP-8/E instructions, each of which is handled somewhat differently by the central processor.

Memory reference instructions, or MRI's are instructions that cause the computer to operate on the content of a memory location, or to use the content of a memory location to operate on the accumulator. Every MRI specifies an operation, which is coded in the first 3 bits of the instruction, and the address of an operand, which is coded in the last 9 bits. There are five PDP-8/E memory reference instructions. Typical applications of MRIs include depositing the content of the AC at a specified address in memory, or jumping to a subroutine with a specified entry address.

Augmented instructions cause the computer to perform a logical (non-arithmetic) operation on the content of one of the major registers. Typical applications of augmented instructions include rotating the AC right or left, testing the content of the AC or link, loading an I/O device buffer from the AC and operating the I/O device, or initializing and operating the interrupt system. Since augmented instructons do not reference a memory address, all 12 bits of the instruction are available for coding the precise operation or sequence of operations to be performed.

There is one housekeeping instruction that comprises the third class of PDP-8/E instructions. This instruction is similar to the MRIs, in that it references a memory address, but similar to the augmented instructions in the manner in which it is executed. It is used to load the PC with a specified memory address, so that the instruction stored at this address will be the next instruction to be executed.

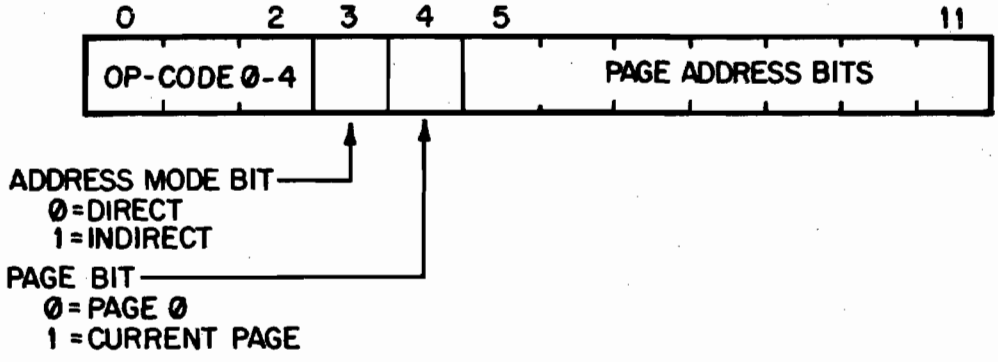


Figure 3-3. Memory Reference Instruction Format

Memory Reference Instructions

Every memory reference instruction contains an operation code, or OP-code, that occupies the first 3 bits of the instruction and an address code that occupies the last 9 bits. This format is illustrated in Figure 3-3. The OP code of an MRI is one of the digits 0 to 4, corresponding to one of five possible operations. The address code specifies the address of an operand, if the instruction is directly addressed, or the address of a pointer to the operand, if the instruction is indirectly addressed.

Bit 3 of an MRI is called the address mode bit. If this bit is set (contains a 1), the MRI is indirectly addressed. This means that the address code of the MRI specifies the page address of a memory location in which the 12-bit address of the operand is stored. If the address mode bit is not set, the instruction is directly addressed. In this case, the address code specifies the page address at which the operand itself is stored.

Bit 4 of an MRI is called the page bit, and bits 5-11 are the page address bits. If the page bit is set, the page address bits contain a page address in the memory page on which the MRI itself is stored (called the current page). If the page bit is not set, the page address bits contain a page address on page 0. In either case, the address specified by the page bit and the page address bits will be the address of the operand, if the MRI is directly addressed, or the address of a memory location that contains the 12-bit address of the operand, if the instruction is indirectly addressed.

In this manner, an MRI may address any one of 400 (octal) locations directly, unless it is stored on page zero. If the MRI is stored in one of the locations 0000-0177, the current page is page zero and the MRI may only address 200 (octal) locations directly. An MRI may address any of 7777 (octal) locations indirectly, however the pointer to the addressed location must reside on page 0 or the current page.

Table 3-2 lists the mnemonics for the five memory reference instructions, their octal codes, and the operations they perform. Only the first 3 bits of the octal codes are listed explicitly; the remaining 9 bits make up the address code, which depends upon where in memory the operand for the MRI is stored.

Table 3-2 Memory Reference Instructions

MNEMONIC	OCTAL	OPERATION
AND	**0XXX**	**Logical AND.** The content of the memory location specified by XXX is combined with the content of the AC by a bitwise logical AND operation. The result is left in the AC, the operand is restored to memory, and the original content of the AC is lost. This instruction, often called "extract" or "mask", may also be considered as a bit-by-bit binary multiplication.
TAD	**1XXX**	**Two's Complement Add.** The content of the memory location specified by XXX is combined with the content of the AC by two's complement addition. The result is left in the AC, the operand is restored to memory, and the original content of the AC is lost. If there is a high-order carry from AC0, the link is complemented.
ISZ	**2XXX**	**Increment and Skip if Zero.** The content of the memory location specified by XXX is incremented by 1 and restored to memory. If the content of the referenced location becomes zero, the PC is incremented by 1 to skip the next sequential instruction. If the content of the referenced location does not become zero, the next instruction is executed.
DCA	**3XXX**	**Deposit and Clear the Accumulator.** The content of the AC is stored in the memory location specified by XXX and the AC is set to zero. The original content of the referenced memory location is lost.
JMS	**4XXX**	**Jump to Subroutine.** The content of the PC is stored in the memory location specified by XXX. The PC is then loaded with 1 more than the address of this location (XXX+1), so that the instruction stored in the memory location following the referenced location is the next instruction to be executed. The content of the AC is not affected.

The Housekeeping Instruction

The only housekeeping instruction is the JMP instruction, with an OP-code of 5, whose format is illustrated in Figure 3-4. Table 3-3 lists the octal code for this instruction and describes the operation it performs.

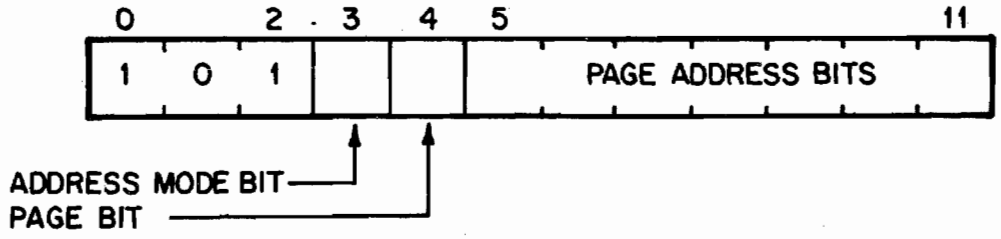


Figure 3-4. Jump Instruction Format

Table 3-3 The Housekeeping Instruction

MNEMONIC	OCTAL	OPERATION
JMP	**5XXX**	**Jump.** The 12-bit address of the memory location specified by XXX is loaded into the PC, so that the instruction stored at this address will be the next instruction to be executed. The original content of the PC is lost. The content of the AC is not affected.

AUGMENTED INSTRUCTIONS

The two augmented instructions are the input/output transfer instruction and the operate instruction. Input/output transfer instructions, which have an OP-code of 6, provide for communication between the central processor and all peripheral devices. They are also used to communicate with the interrupt system. Operate instructions, with an OP-code of 7, are used to perform logical operations on the content of the major registers.

The Input/Output Transfer Instruction

Input/output transfer (IOT) instructions are used to initiate the operation of peripheral devices and to transfer data between peripherals and the central processor. Figure 3-5 shows the format of an IOT instruction. Bits 0-2 contain the OP-code, which must be 6 to specify an IOT instruction. Bits 3-8 contain a device selection code that is transmitted to every peripheral device whenever the IOT instruction is executed. Device selectors within the peripheral devices monitor these device codes. When a peripheral device recognizes a device code as that peripheral's assigned code, the device accepts the last three bits of the IOT instruction.

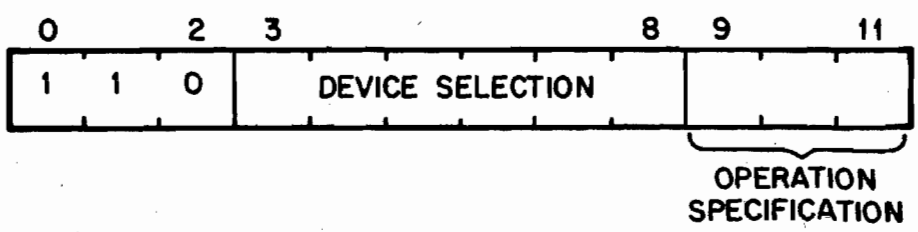


Figure 3-5. IOT Instruction Format

Bits 9-11 of an IOT instruction contain the operation specification code. These bits may be set to specify one of up to eight operations. If a peripheral device is capable of performing more than eight different operations, it is necessary to assign two or more device codes to the peripheral device.

The Operate Instruction

The operate instruction consists of 3 groups of microinstructions. Group 1 microinstructions, which are identified by the presence of a 0 in bit 3, are used to perform logical operations on the content of the accumulator and link. Group 2 microinstructions, which are identified by the presence of a 1 in bit 3 and a 0 in bit 11, are used primarily to test the content of the accumulator and link, then conditionally skip the next sequential instruction. Group 3 microinstructions have a 1 in bit 3 and a 1 in bit 11. They are used to perform logical operations on the content of the accumulator and multiplier quotient registers.

Operate microinstruction from any group may be microprogrammed with most other operate microinstructions of the same group. The octal code for a microprogrammed combination of two (or more) microinstructions is the bitwise logical OR of the octal codes for the individual microinstructions. When more than one operation is microprogrammed into a single instruction, the operations are performed in a prescribed sequence, with logical sequence 1 microinstructions performed first, then logical sequence 2 microinstructions, and so on. Two operations with the same logical sequence number are performed simultaneously.

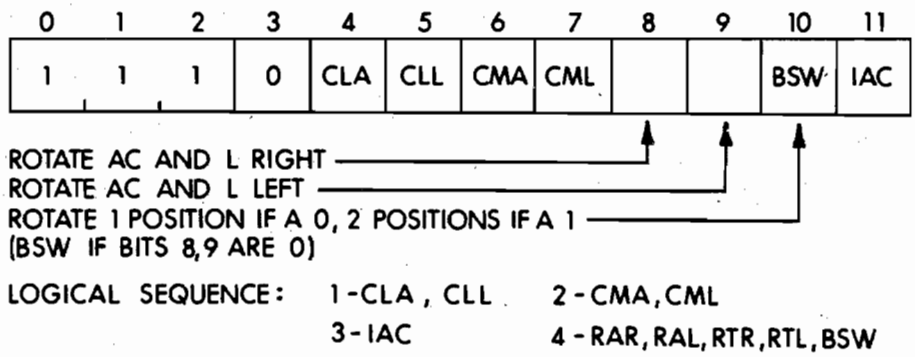


Figure 3-6. Group 1 Operate Microinstructions

GROUP 1 MICROINSTRUCTIONS

Figure 3-6 shows the format of a group 1 microinstruction. The OP-code must be 7, to indicate an operate instruction, and bit 3 must contain a 0, to indicate a group 1 microinstruction. Any one of bits 4 to 11 may be set (loaded with a binary 1) to indicate a specific group 1 microinstruction. If more than one of these bits is set, the instruction is a microprogrammed combination of group 1 microinstructions, which will be executed according to the logical sequence shown in Figure 3-6.

Table 3-4 lists the group 1 microinstructions, their assigned mnemonics and the operations they perform. Two or more of these microinstructions may be microprogrammed into a single 12-bit instruction, as long as the instruction does not contain more than 1 of the logical sequence 4 microinstructions (RAR, RAL, RTR, RTL and BSW). This restriction should not impose any constraint on the user, since the five logical sequence 4 microinstructions perform mutually incompatible operations.

Table 3-4 Group 1 Operate Microinstructions

MNEMONIC	OCTAL	OPERATION
NOP	**7000**	**No Operation.** This instruction causes a 1-cycle delay in program execution, without affecting the state of the computer. It may be used for timing synchronization or as a convenient means of deleting another instruction from a program.
IAC	**7001**	**Increment Accumulator.** The content of the accumulator is incremented by 1.
BSW	**7002**	**Byte Swap.** The content of the six low-order bits of the AC is exchanged with the content of the six high-order bits. That is, AC0 is exchanged with AC6, AC1 is exchanged with AC7, etc. The content of the link is not affected.
RAL	**7004**	**Rotate Accumulator Left.** The content of AC1-11 is shifted into AC0-10. The content of AC0 is shifted into the link, and the content of the link is shifted into AC11.
RTL	**7006**	**Rotate Two Left.** Equivalent to two consecutive RAL operations.
RAR	**7010**	**Rotate Accumulator Right.** The content of AC0-10 is shifted into AC1-11. The content of the link is shifted into AC0, and the content of AC11 is shifted into the link.
RTR	**7012**	**Rotate Two Right.** Equivalent to two consecutive RAR operations.
CML	**7020**	**Complement Link.** The content of the link is complemented.
CMA	**7040**	**Complement Accumulator.** The content of each bit of the AC is complemented. This has the effect of replacing the content of the AC with its one's complement.
CLL	**7100**	**Clear Link.** The link is loaded with a binary 0.
CLA	**7200**	**Clear Accumulator.** Each bit of the AC is loaded with a binary 0.

Table 3-5 lists four microprogrammed combinations of group 1 microinstructions which are used so frequently that they have been assigned their own mnemonics. Note that the octal codes for a microprogrammed combination of operate microinstructions is the bitwise logical OR of the octal codes of the individual microinstructions. Other frequently used combinations of operate microinstructions are listed in the appendix of this handbook.

Table 3-5 Microprogrammed Combinations of Group 1 Microinstructions

MNEMONIC	OCTAL	OPERATION
CIA	**7041**	**Complement and Increment Accumulator.** The content of the AC is replaced with its two's complement. This is a microprogrammed combination of CMA and IAC.
STL	**7120**	**Set the Link.** The link is loaded with a binary 1. This is a microprogrammed combination of CLL and CML.
STA	**7240**	**Set the Accumulator.** Each bit of the AC is loaded with a binary 1. This is a microprogrammed combination of CLA and CMA.
GLK	**7204**	**Get the Link.** The AC is cleared and the content of the link is shifted into AC11 while a 0 is shifted into the link. This is a microprogrammed combination of CLA and RAL.

GROUP 2 MICROINSTRUCTIONS

Figure 3-7 shows the format of a group 2 microinstruction. The operation code must be 7, to indicate an operate instruction, and bit 3 must contain a 1 while bit 11 must contain a 0, to indicate a group 2 microinstruction. Bits 4-10 may be set to indicate a specific group 1 microinstruction. If more than one of bits 4-7 or 9-10 is set, the instruction is a microprogrammed combination of group 2 microinstructions, which will be executed according to the logical sequence shown in Figure 3-7. Table 3-6 lists the group 2 microinstructions, their mnenomics, and the operations they perform.

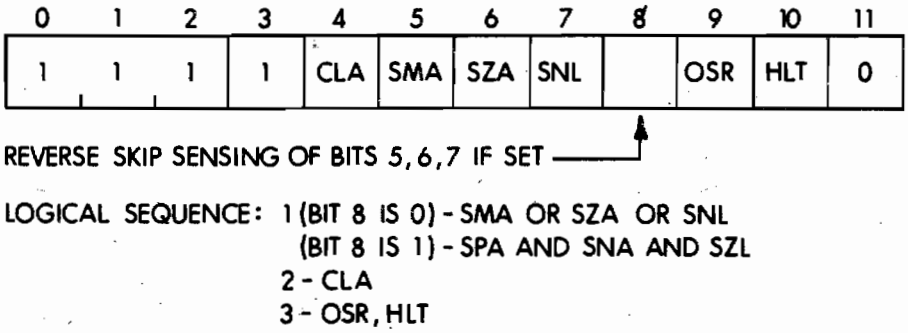


Figure 3-7. Group 2 Operate Microinstructions

Table 3-6 Group 2 Microinstructions

MNEMONIC	OCTAL	OPERATION
HLT	**7402**	**Halt.** Clears the run flip-flop so that program execution stops at the end of TP4 of the current machine cycle.
OSR	**7404**	**Logical OR with Switch Register.** The content of the programmer's console switch register (SR) is combined with the content of the AC by a bitwise logical OR operation. The result is left in the AC and the original content of the AC is lost. The content of the SR is not affected.
SKP	**7410**	**Skip.** The content of the PC is incremented by 1, to skip the next sequential instruction.
SNL	**7420**	**Skip on Non-Zero Link.** The content of the link is sampled. If the link contains a 1, the content of the PC is incremented to skip the next sequential instruction. If the link contains a 0, the next instruction is executed.
SZL	**7430**	**Skip on Zero Link.** The content of the link is sampled. If the link contains a 0, the content of the PC is incremented to skip the next sequential instruction. If the link contains a 1, the next instruction is executed.
SZA	**7440**	**Skip on Zero Accumulator.** The content of each bit of the AC is sampled. If every bit contains a 0, the content of the PC is incremented to skip the next sequential instruction. If any bit contains a 1, the next instruction is executed.
SNA	**7450**	**Skip on Non-Zero Accumulator.** The content of each bit of the AC is sampled. If any bit contains a 1, the content of the PC is incremented by 1 to skip the next sequential instruction. If every bit contains a 0, the next instruction is executed.
SMA	**7500**	**Skip on Minus Accumulator.** The content of AC0 is sampled. If AC0 contains a 1, indicating that the AC contains a negative two's complement number, the content of the PC is incremented to skip the next sequential instruction. If AC0 contains a 0, the next instruction is executed.

Table 3-6 Group 2 Microinstructions (Cont.)

MNEMONIC	OCTAL	OPERATION
SPA	**7510**	**Skip on Positive Accumulator.** The content of AC0 is sampled. If AC0 contains a 0, indicating that the AC contains a positive two's complement number (or zero), the content of the PC is incremented to skip the next sequential instruction. If AC0 contains a 1, the next instruction is executed.
CLA	**7600**	**Clear Accumulator.** Each bit of the AC is loaded with a binary 0.

Skip microinstructions may be microprogrammed with CLA, OSR or HLT microinstructions, and also with other skip microinstructions that have the same value in bit 8. Skip microinstructions which have a 0 in bit 8 may not be microprogrammed with skip microinstructions which have a 1 in bit 8, however.

When two or more skip microinstructions are microprogrammed into a single instruction, the resulting condition on which the decision will be based is the logical OR of the individual conditions when bit 8 is 0, or the logical AND of the individual conditions when bit 8 is 1 (see Figure 3-7).

Table 3-7 lists every legal combination of skip microinstructions, along with the resulting condition upon which the decision to skip or execute the next sequential instruction is based. This table does not include microprogrammed combinations of skip microinstructions and the CLA, OSR or HLT microinstructions.

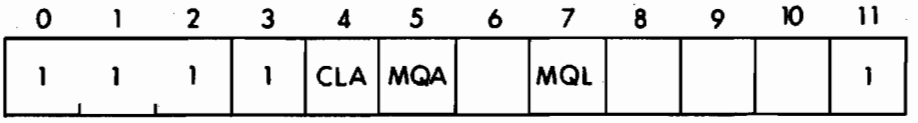
Table 3-7 Microprogrammed Combinations of Group 2 Microinstructions

MNEMONIC			OCTAL	OPERATION
SZA	**SNL**		7460	Skip if AC = 0 or L = 1 or both.
SNA	**SZL**		7470	Skip if AC = 0 and L = 0.
SMA	**SNL**		7520	Skip if AC < 0 or L = 1 or both.
SPA	**SZL**		7530	Skip if AC $\geq$ 0 and L = 0.
SMA	**SZA**		7540	Skip if AC $\leq$ 0.
SPA	**SNA**		7550	Skip if AC > 0.
SMA	**SZA**	**SNL**	7560	Skip if AC $\leq$ 0 or L = 1 or both.
SPA	**SNA**	**SZL**	7570	Skip if AC > 0 and L = 0.

GROUP 3 MICROINSTRUCTIONS

Group 3 microinstructions are used to transfer data between the AC and multiplier quotient (MQ) registers. Although these microinstructions are intended primarily for use with the KE8-E Extended Arithmetic Element, they are also useful when the MQ is employed as a temporary storage register, even if an EAE is not installed.

Figure 3-8 shows the format of a group 3 microinstruction. The operation code must be 7, to indicate an operate instruction, while bits 3 and 11 must both contain a 1, to indicate a group 3 microinstruction. Any one of bits 4, 5 or 7 may be set to indicate a specific group 3 microinstruction. If more than one of these bits is set, the instruction is a microprogrammed combination of group 3 microinstructions.



LOGICAL SEQUENCE: 1 - CLA
2 - MQA, MQL
3 - ALL OTHERS

Figure 3-8. Group 3 Operate Microinstructions

Table 3-8 lists the three group 3 microinstructions, their assigned mnemonics, and the operations they perform. This table also lists two useful microprogrammed combinations of group 3 microinstructions.

Table 3-8 Group 3 Microinstructions

MNEMONIC	OCTAL	OPERATION
CLA	**7601**	**Clear Accumulator.** Each bit of the AC is loaded with a binary 0.
MQL	**7421**	**Multiplier Quotient Load.** The content of the AC is loaded into the MQ. The AC is cleared, and the original content of the MQ is lost.
MQA	**7501**	**Multiplier Quotient into Accumulator.** The content of the MQ is combined with the content of the AC by a bitwise logical OR operation, and the result is loaded into the AC. The original content of the AC is lost, but the original content of the MQ is not affected. Note that this instruction provides the programmer with a direct inclusive OR operation.

Table 3-8 Group 3 Microinstructions (Cont.)

MNEMONIC	OCTAL	OPERATION
SWP	**7521**	**Swap Accumulator and Multiplier Quotient.** The content of the AC and the content of the MQ are exchanged. This is a microprogrammed combination of MQA and MQL.
CAM	**7621**	**Clear Accumulator and Multiplier Quotient.** Each bit of both the AC and the MQ loaded with a binary 0. This is a microprogrammed combination of CLA and MQL.

INSTRUCTION EXECUTION AND TIMING

The major state generator provides control signals that may enable one of three major states during each memory cycle.

The FETCH major state is used to fetch an instruction from memory. FETCH is enabled whenever execution of an instruction was completed at the end of the last memory cycle. During a FETCH cycle, the processor reads an instruction from the memory location whose address is contained in the PC and decodes the first 3 bits of the instruction. If the instruction is an augmented instruction (OP-code 6 or 7) it is executed during the FETCH cycle. If the instruction is a JMP or memory reference instruction it is decoded further. Directly addressed JMP instructions will also be executed during the FETCH cycle; however, indirectly addressed JMP instructions and all MRIs require at least one additional cycle.

If an indirectly addressed JMP or MRI instruction was read from memory during the last FETCH cycle, the DEFER major state will be enabled during the following cycle. If a directly addressed MRI was read, the EXECUTE major state will be enabled next.

The DEFER major state is used to decode indirect memory references. During a DEFER cycle, the processor computes the 12-bit address of the memory location specified by bits 4-11 of the indirectly addressed JMP or MRI instruction and reads the address of an operand from this location. If the referenced location is an autoindex register, its content is incremented by 1 during the DEFER cycle, and the incremented value is taken as the operand address. Execution of an indirectly addressed JMP instruction will be completed during the DEFER cycle, but if the instruction is an indirectly addressed MRI, the EXECUTE major state must be enabled to complete execution during the following cycle.

Memory reference instruction execution is always completed during an EXECUTE cycle. The EXECUTE major state is entered from FETCH, when a directly addressed MRI is read from memory, or from DEFER, when the current instruction is an indirectly addressed MRI. In either case, instruction execution will be completed by the end of the EXECUTE cycle, and the FETCH major state will be enabled during the following cycle.

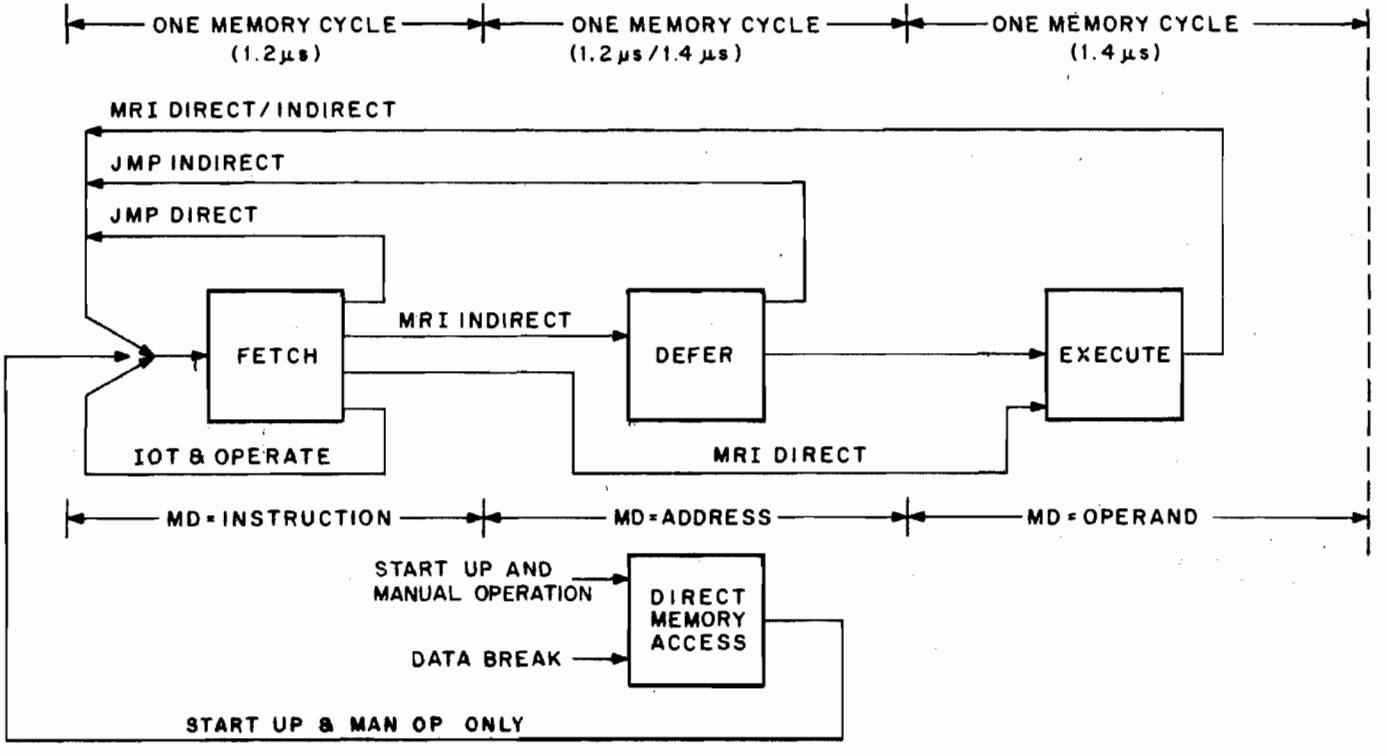


Figure 3-9. Major State Flow Diagram

A fourth major state, Direct Memory Access or DMA, is defined when neither FETCH, DEFER nor EXECUTE is enabled. The DMA state is entered during data break transfers and during manual operation of the switches on the programmer's console. Figure 3-9 is a diagram that illustrates major state flow as a function of instruction type. This diagram indicates which major states will be entered during execution of any given type of instruction.

Memory Timing

The timing generator produces four time state signals, designated TS1 through TS4, and four time pulse signals, designated TP1 through TP4. The timing diagram of Figure 3-10 illustrates the relationship between the time state and time pulse signals for a fast (1.2 microseconds) memory cycle. A slow (1.4 microseconds) memory cycle is produced when the EXECUTE major state is enabled or when the DEFER major state is enabled and the current instruction is an indirectly addressed MRI with autoindexing. Slow cycle timing is identical to fast cycle timing except that TS2 is extended for an additional 0.2 microseconds. All time state and time pulse signals are gated out onto the OMNIBUS where they are used as control signals throughout the system.

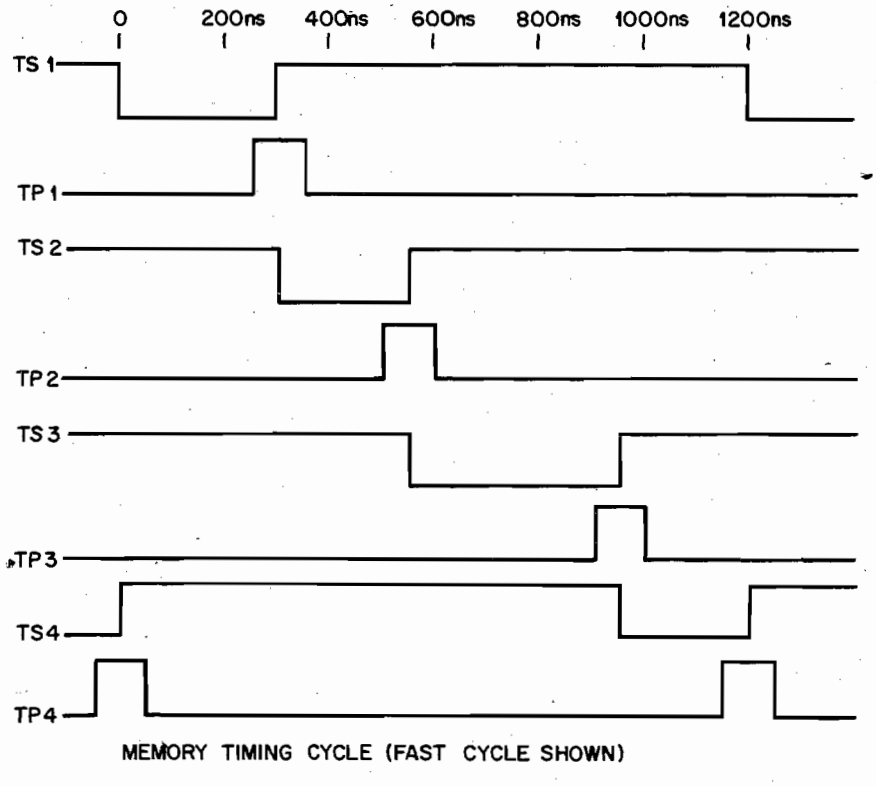


Figure 3-10. PDP-8/E Memory Timing Diagram (Fast Cycle)

FETCH Major State

Figure 3-11 is a simplified flow chart showing the general sequence of operations that occurs during every FETCH cycle. Notice that FETCH is always a fast cycle, and that the major state to be enabled during the next cycle depends on the type of instruction that is read from memory during the FETCH cycle.

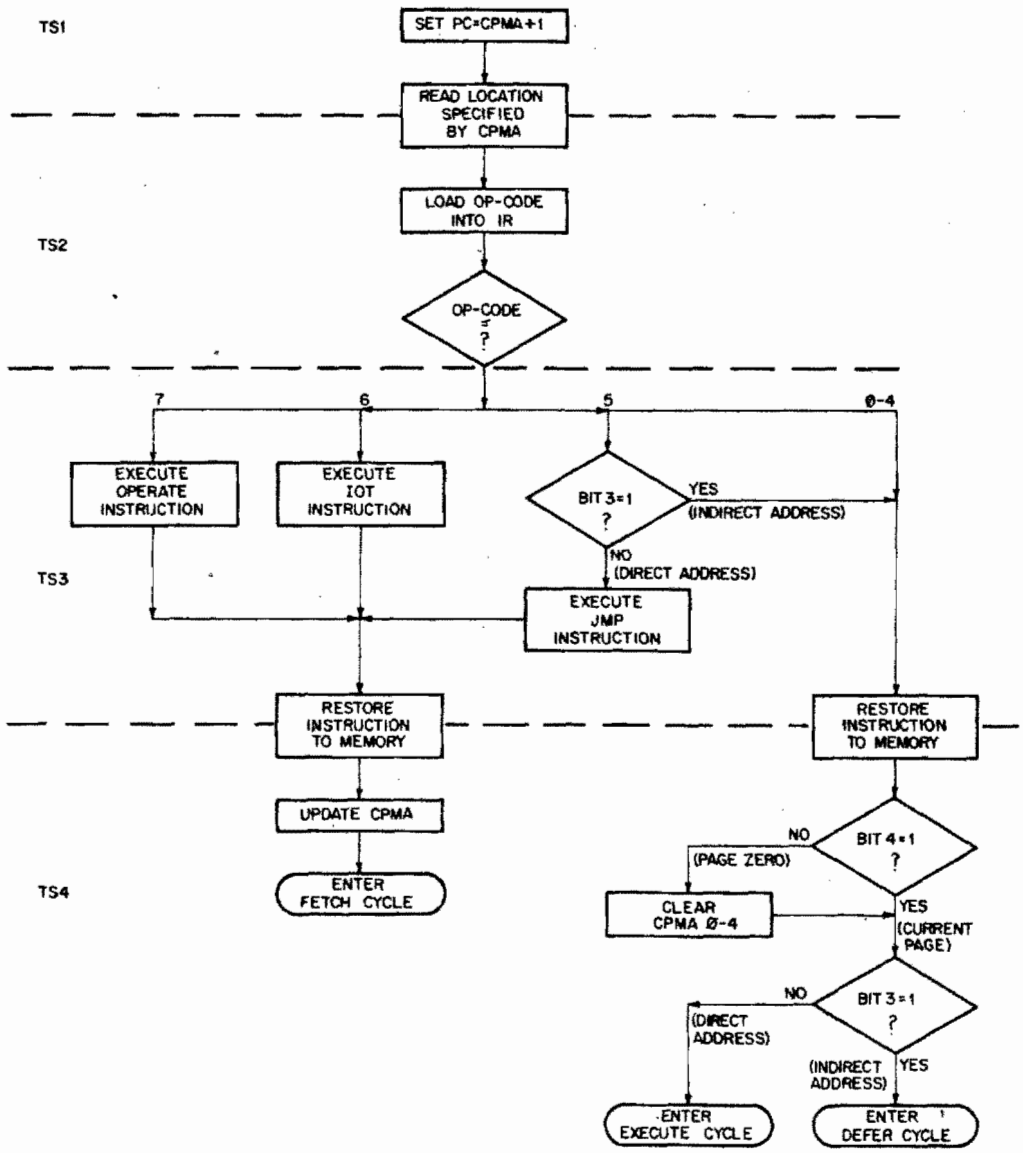


Figure 3-11. FETCH Major State

DEFER Major State

Figure 3-12 is a simplified flow chart showing the sequence of operations that occurs during a DEFER cycle. DEFER is enabled whenever the current instruction is an indirectly addressed JMP or memory reference instruction. It will be a slow cycle if the current instruction references one of the autoindex registers (locations 0010 to 0017) or a fast cycle in any other case. DEFER is always entered from the FETCH major state. A DEFER cycle will be followed by a FETCH cycle, if the current instruction is an indirect JMP, or by an EXECUTE cycle, if the current instruction is an indirectly addressed MRI.

EXECUTE Major State

Figure 3-1 is a simplified flow chart showing the sequence of operations that occurs during an EXECUTE cycle. EXECUTE is entered from FETCH, if the current instruction is a directly addressed MRI, or from DEFER, if the instruction is an indirectly addressed MRI. An EXECUTE cycle is always followed by a new FETCH cycle.

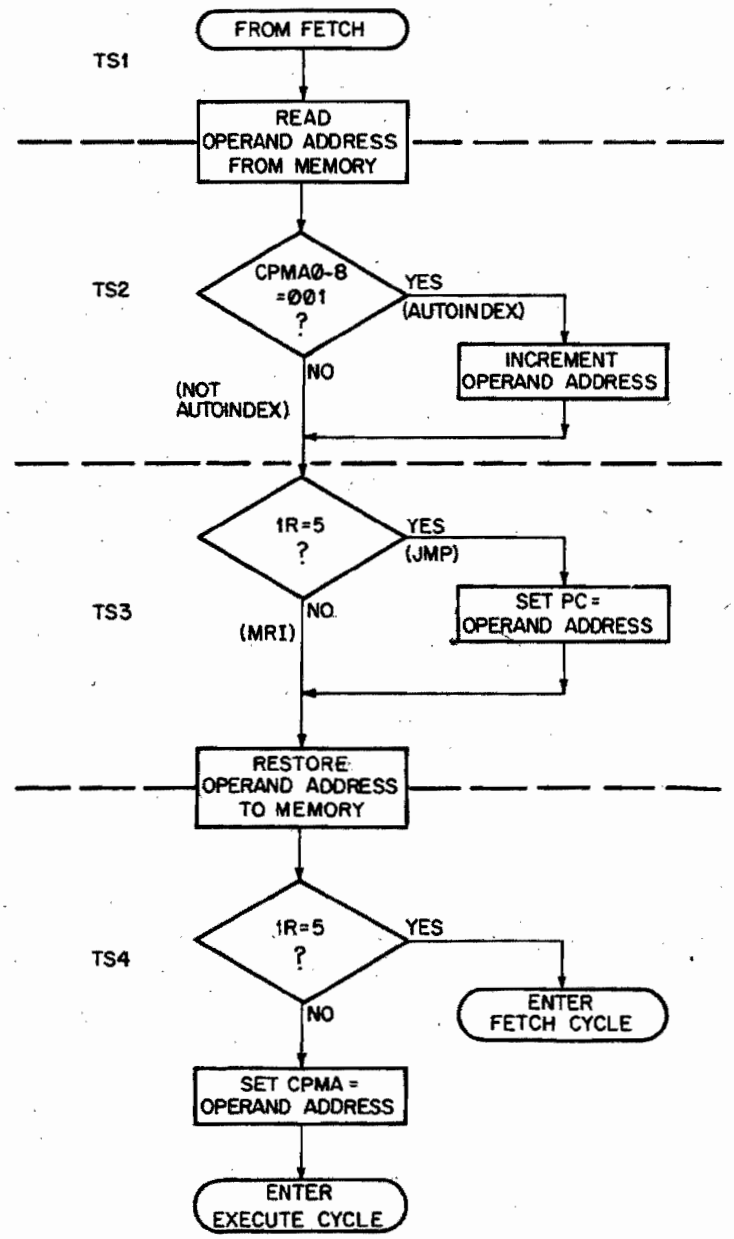


Figure 3-12. DEFER Major State

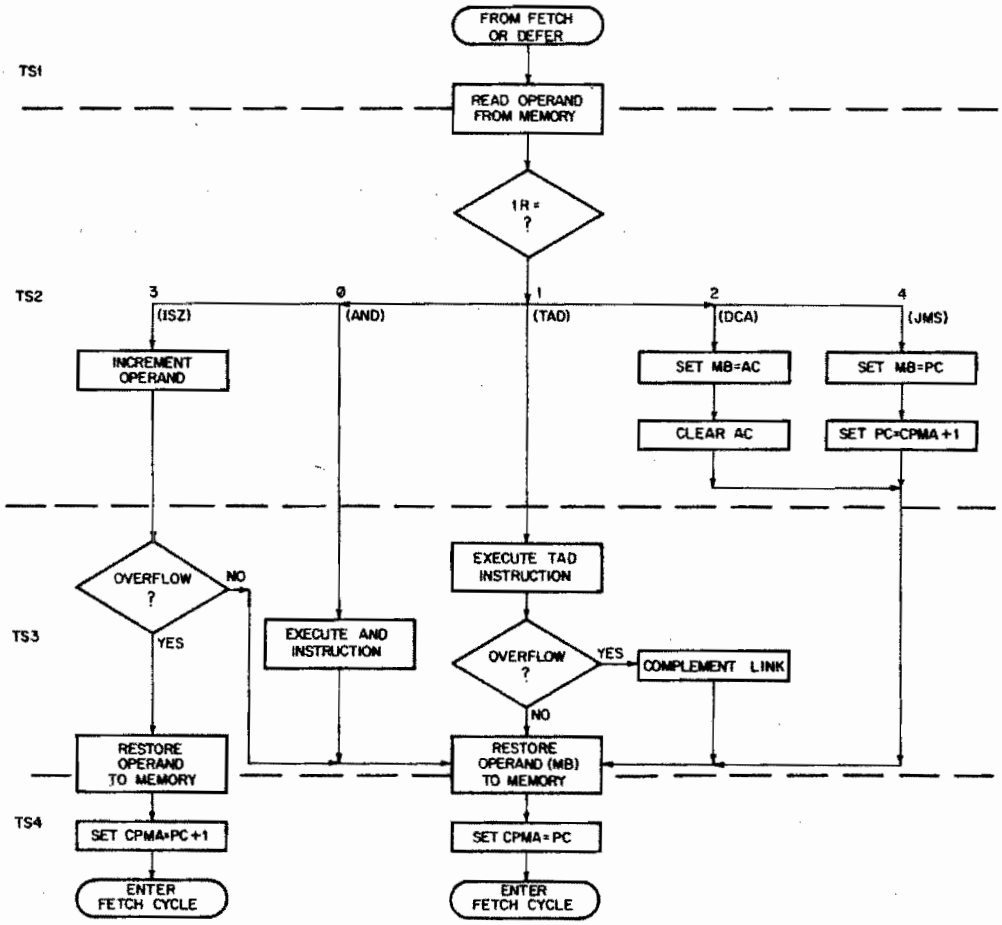


Figure 3-13. EXECUTE Major State