

D. Vonada



INTEROFFICE MEMORANDUM

DATE: 6 November 1967

SUBJECT: PDP-X Processor Description

TO: Distribution Lists A,B,C. FROM: H. Burkhardt

Attached is a revised version of PDP-X Technical Memorandum #13 which is now obsolete and should be destroyed. Many areas of the central processor architecture have been changed, revised or completely changed. It is, therefore, suggested that this document be read fully.

The areas of major change are:

1. The elimination of the floating-point registers from the memory address space.
2. The addition of several trap conditions and a general re-arrangement of the PSW.
3. The addition of an additional mode on the DIV and LDIV instructions allowing for integer operations.
4. The elimination of the IOX instruction and the addition of the IORC and IOWC instructions.
5. The addition of several IOD class instructions including HALT and CONSOLE SWITCH operations.
6. A change in the memory protection instructions.
7. The addition of an additional multiplexor channel priority (level 8).
8. A change in the way in which a device is granted an interrupt. This includes a change in the Status Registers of all devices previously described in TM's 7, 8, 9, 15 (i.e., the change of the LOW bit to a HIGH bit).
9. A reduction to 200₈ (from 400₈) of the number of memory words reserved for the handling of device interrupts.

This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.

PDP-X Technical Memorandum # 29

Title: PDP-X Processor Description

Author(s): H. Burkhardt
L. Seligman

Index Keys: Architecture
Instruction Set
Processor

Distribution
Keys: A,B,C

Obsolete: Technical Memorandum # 13

Revision: None

Date: October 16, 1967

Index

1.0 Introduction

1.1 Models

2.0 Processor Architecture

2.1 Instruction Format

2.2 Data Formats

2.2.1 Fixed-Point Arithmetic Operations

2.2.2 Logical Operations

2.2.3 Floating-Point Operations

2.2.4 Character Operations

2.2.5 Byte Pointer

2.2.6 Character Set

2.3 Addressing

2.3.1 Addressing Exceptions

2.3.2 Address Calculation

2.4 General Registers

2.5 Program Status Word

2.5.1 Traps

2.5.2 Condition Codes

2.6 Instructions

2.6.1 Basic Instructions

2.6.2 Extended Operation Class

2.6.3 Extended Arithmetic Group

2.6.4 Character Group

2.6.5 Logical Compare and Modify Group

2.6.6 Push Down Group

2.6.7 IO Instructions

2.7 Priority (Interrupt) System

2.8 Protection Feature

2.8.1 Instruction Protection

2.8.2 Memory Protection

2.8.3 Monitor Calls

2.8.4 Instructions for Memory Protection System

2.8.5 Summary

3.0 IO System

3.1 Devices and Controllers

3.2 Modes of Data Transfer

3.3 Operation of the Multiplexor Channel and Interrupt

3.3.1 Program Controlled Interrupt Service

3.3.2 Multiplexor Channel

3.4 Basic Peripheral Structure

- 3.5 Paper Tape Peripherals
 - 3.5.1 Paper Tape Reader/Punch
 - 3.5.2 Keyboard/Printer
 - 3.5.3 Priority Assignments
 - 3.5.4 Paper Tape Peripherals Status Bytes
 - 3.6 Device Assignment Table
 - 3.7 IO Bus
 - 3.7.1 General Characteristics
 - 3.7.2 Operation
 - 3.7.3 Line Definitions
 - 3.8 IO Configurations
- 4.0 Appendices
- 4.1 Assembly Language Conventions
 - 4.2 Instructions (Alphabetic)
 - 4.3 Peripheral Structure (Continued)
 - 4.3.1 Flow Chart Conventions
 - 4.3.2 Output Device - No Interrupt
 - 4.3.3 Output Device - Interrupt Mode
 - 4.3.4 Input Device - No Interrupt
 - 4.3.5 Input Device - Interrupt Mode
 - 4.3.6 Programming Example - Punch Routine
 - 4.4 Reserved Memory

1.0 Introduction

PDP-X is a modern, very high performance, third generation, binary, two's complement computer family designed for the small computer market. Upward and downward (limited) program compatibility permits easy system growth and enhances application programming. Standard IO and Memory interfaces are used for all processor models and all peripheral devices. The architecture lends itself to fourth generation hardware implementation and the development of multiprocessor systems.

The system architecture of the PDP-X computer family is described below. Two particular members of the family are suggested; however, details of their implementation are beyond the scope of this document. PDP-X/I may be thought of as a PDP-8 class machine, PDP-X/II as a PDP-9 class machine.

1.1 Models

The smallest PDP-X model has two priority levels, main program and interrupt level, both general register sets reside in core memory. The Model I instruction set consists of only the basic instructions; all EOP (Extend Operation) class instructions trap. Memory is expandable from 4K to 32K. All peripheral equipment will run on Model I within the constraints of the single interrupt level.

The Model I processor may not be upgraded to a Model II processor. All of the other components of the system, however, may be used with a Model II processor.

The medium-sized PDP-X model has two sets of hardware general registers for its two priority levels: main program and interrupt level. The Model II instruction set consists of both the basic instructions and extended instructions. The basic configuration includes High-Speed Paper Tape, 33 KSR Teletype, and an 8K memory.

- a. The Priority Interrupt System may be added. This option adds six sets of general registers, the priority interrupt structure, and special instructions to modify the state of the interrupt system.
- b. The Protection Option may be added. This option consists of user mode/executive mode, the memory paging system and certain special instructions to alter the state of the paging hardware. The Priority Interrupt system is required before this option may be added.
- c. Power Fail, Memory Parity, Machine Check hardware may be added.
- d. The memory system may be increased to 32K or to 128K, if the Protection Option is installed.

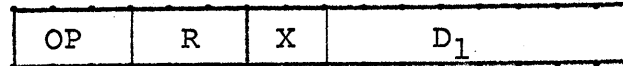
Model	Register Sets	Instruction Set	Standard Memory	Standard IO	Interrupt Structure	Protection System	Add Time	Multiply Time (Signed)	Index Time
I _a	2, core	Basic	4k - 2μ	33 ASR	2 levels	None	8μ	subroutine	2μ
I _b	2, core	Basic	4k - .8μ	33 ASR	2 levels	None	3.2μ	subroutine	.8μ
II _a *	2, hardware	Extended Set	8k - .8μ	33 ASR High-Speed Paper Tape	2 levels	None	1.6μ	<7μ	0
II _b	4 or 8 hardware	Extended Set	8k - .8μ	33 ASR High-Speed Paper Tape	4 or 8 Fully Nested Levels	None	1.6μ	<7μ	0
II _c	4 or 8 hardware	Extended Set	16k - .8μ	33 ASR High-Speed Paper Tape	4 or 8 Fully Nested Levels	User/Exec Modes Paging	1.6μ	<7μ	0

*Model II includes a byte multiplexor channel.

2.0 Processor Architecture

2.1 Instruction Format

short form



basic op

long form



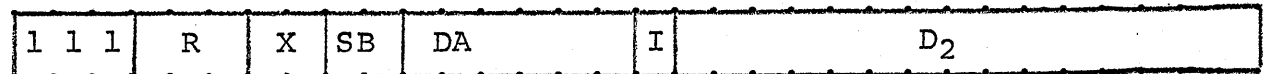
extended

op form

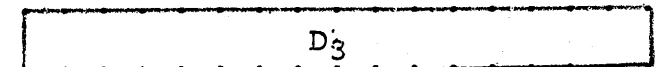


IO

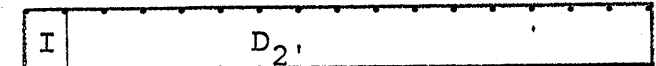
instruction



immediate word in EOP, IO or long form



indirect word



2.1 (Cont.)

mnem	bits	definition
OP	3	basic operation code specifying major instruction class
R	3	general register specification or sub-function selection for non-accumulator reference instructions
X	2	index register and address mode selector
D ₁	8	short form address
D ₂	15	long form address
D ₂	15	indirect word
D ₃	16	immediate operand
I	1	indirect addressing specification
EOP	8	extended operation code specifying instruction
DA	6	IO device address (device selection)
SB	2	selects one of four (optional) IO busses. SB is normally 0

2.2 Data Formats

The hardware and software capabilities include operations on single and double precision fixed-point data, single precision logical data, and character bytes. On the larger processors, most of the operations will be performed by the hardware. On smaller processors, the operations may be performed by resident subroutines.

2.2.1 Fixed-Point Arithmetic Operations

The basic arithmetic operand is the 16-bit fixed-point binary word. To preserve precision, all products and dividends are 32 bits long.

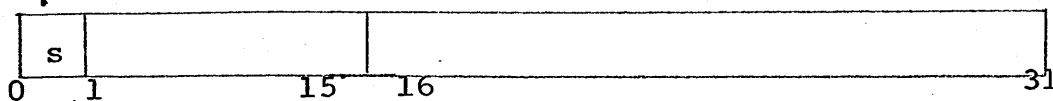
Since the 16-bit word size accommodates a 15-bit address, fixed-point arithmetic can be used both for integer operand arithmetic and for address arithmetic. Since integer and addressing arithmetic often requires repeated references to operands or to intermediate results, the use of multiple registers is advantageous in arithmetic sequences and address calculations.

Additions, subtractions, multiplications, divisions and comparisons are performed upon one operand in a register and another operand either in a register or main storage. Two's complement notation is used to facilitate multi-precision arithmetic.



Signed (2's complement) single precision integer.

$$-(2^{15} - 1) \leq \text{word} \leq (2^{15} - 1)$$



Signed (2's complement) double precision integer.

$$-(2^{31} - 1) \leq \text{word} \leq (2^{31} - 1)$$

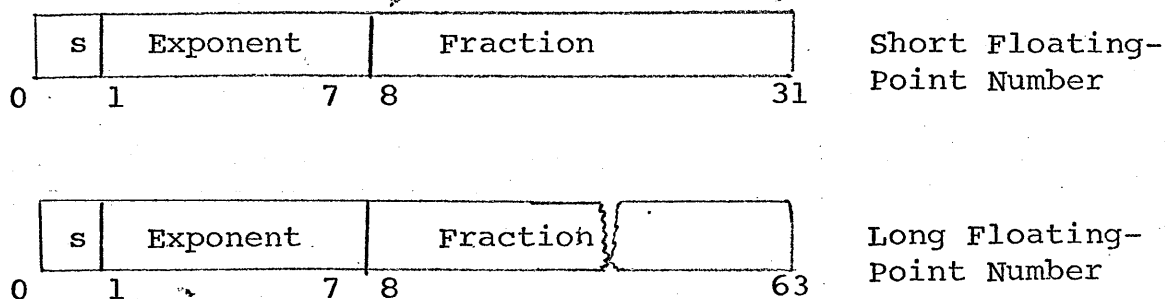
The address of a double precision quantity is the address of the high-order word. This address must be even (i.e., Effective Address₁₅ = 0).

2.2.2 Logical Operations

Logical Operations are performed on 16-bit binary words with one operand in a register and another operand either in a register or in storage.

2.2.3 Floating-Point Operations

Floating-point numbers occur in either of two fixed length formats - short and long. These formats differ only in the lengths of the fractions



Operands are either 32 (two words) or 64 (four words) bits long. The short form, equivalent to seven decimal places of precision, permits a maximum number of operands to be placed in storage and gives the shortest execution times. The long form gives up to 17 decimal places of precision.

The fraction of floating-point number is expressed in hexadecimal (base 16) digits each consisting of four binary bits and having the values 0-15. In the short format, the fraction consists of six hexadecimal digits occupying bit positions 8-31. In the long format the fraction has 14 hexadecimal digits occupying bit positions 8-63. The fraction is always in positive form, the sign bit (if a 1) indicates that the number is negative.

The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit (between bits 7 and 8). To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1-7 of both formats, is used to indicate the power. The characteristic is treated as an excess 64 number with a range -64 to +63 corresponding to 0-127. This permits representation of decimal numbers with magnitudes in the range of

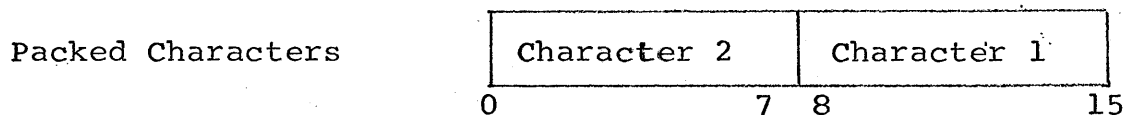
2.2.3 (Cont.)

10^{-78} to 10^{75} . Four 64-bit floating-point registers are provided. (This is optional; if the floating-point operations are performed by resident subroutines, the simulated registers are internal to the routines.) Arithmetic operations are performed with one operand in a register and another either in a register or from storage. The result is developed in a register. The availability of several floating-point register eliminates much storing and loading of intermediate results.

The addresses of short-form floating-point numbers must be even ($EFA_{15} = 0$). The addresses of long form floating-point numbers must be evenly divisible by four ($EFA_{14-15} = 0$). In either case, the address is the address of the high-order word containing the exponent.

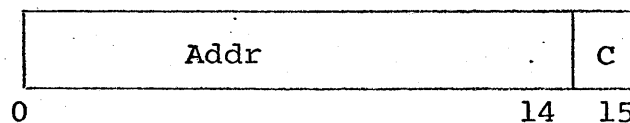
2.2.4 Character Operations

Two 8-bit bytes (characters) may be stored as a single computer word.



Data transfers to and from external devices are done through an 8-bit channel. If, during an I/O read or write operation the device requests a two-byte transfer, character 1 is received (transmitted) and then character 2. If the device does not request a two-byte transfer, only character 1 of the effective word is effected.

2.2.5 Byte Pointer



The left 15 bits of the byte pointer select the word address of a byte pair. Bit 15 selects one of the 2 bytes:

0 = right byte

1 = left byte

2.2.6 Character Set

The character set used for PDP-X peripherals (with a few exceptions) is the USASCII character set depicted below

	000	020	040	060	100	120	140	160
00	NUL	DLE	SP	0	è	P	`	p
01	SOH	DC1	!	1	A	Q	o	q
02	STX	DC2	"	2	B	R	b	r
03	ETX	DC3	#	3	C	S	c	s
04	EOT	DC4	\$	4	D	T	d	t
05	ENQ	NAK	%	5	E	U	e	u
06	ACK	SYN	&	6	F	V	f	v
07	BEL	ETB	'	7	G	W	g	w
10	BS	CAN	(	8	H	X	h	x
11	HT	EM	)	9	I	Y	i	y
12	LF	SUB	*	:	J	Z	j	z
13	VT	ESC	+	;	K	[	k	{
14	FF	FS	,	<	L	\	l	
15	CR	GS	-	=	M	]	m	}
16	SO	RS	.	>	N	↑	n	~
17	SI	US	/	?	O	←	o	DEL

(The octal for a given character is obtained by adding the column heading and the row heading)

Ex: M 100 + 15 = 115

: 040 + 01 = 041)

2.2.6 (Cont.)

The abbreviations of the more common special characters is given below:

BEL	Bell, rings bell on 33, 35, 37 Teletypes
HT	Horizontal Tab, spaces carriage to next tab stop on 35 or 37 Teletypes
LF	Line Feed, moves paper up one printing position
VT	Vertical Tab
FF	Form Feed
CR	Carriage Return, return carriage to beginning of line
DEL	Delete Code

See descriptions of particular IO devices to determine their response to these and other special characters.

2.3 Addressing

Addresses are generated by either long or short format instructions. In either case, the processor forms a 15-bit Effective Address (EFA) which it sends to the memory system. The left byte (high order 7 bits) of the address is called the field; there are 128 fields of 256 words each.

The available addressing modes are:

- a. direct (no indexing) to any word in memory
- b. relative ($\pm 127_{10}$) to the instruction
- c. immediate (the next location is the effective address)
- d. indexed
- e. linked (the subroutine linkage register is used to pick up arguments or to make returns)

In the short format, the displacement (D1) is treated as a sign extended two's complement number ($|D1| \leq 177_8$), unless the addressing mode is direct. In this case, D1 is a field 0 address. Long format instructions are specified whenever $D1 = 200_8$ or whenever the instruction implicitly forces this format (all IO and extended op code instructions).

The address mode is, thus, a function of format (short or long) and of the X bits of the instruction:

Addressing Table (Effective Address Calculation)

X	Short	Long (Instruction = IO or EOP or $D1 = 200_8$)
0	D1 (Field 0)	D2 (Direct)
1	$\pm D1 + PC$ (Relative)	$PC + 1$ (Immediate)
2	$\pm D1 + R2$ (Linked)	$D2 + R2$ (Linked)
	indexed by subroutine linkage register	indexed by subroutine linkage register
3	$\pm D1 + R3$ (Indexed)	$D2 + R3$ (Indexed)

2.3 (Cont.)

One of the primary uses of index registers arises from their ability to modify instruction addresses. For this to occur, the instruction must specify the particular register that is to take part in the modifying activity. This is done by placing the appropriate bit configuration in the X field (bit positions 6 and 7) of the instruction.

PDP-X has 2 index registers and provides for relative addressing. Index register coding is:

- 0 non-indexing
- 1 relative addressing (index by program counter) or immediate mode in long form
- 2 index registers
- 3 index registers

The assignment of the appropriate bit configuration in the X field selects the index register to be used in the modifying activity. The instruction is then executed as if its address field contained the stated address plus (two's complement) the contents of the index register. For example, assume that storage location 01000 contains the instruction ADD 02000 and that this instruction has a 2 in its X field. If the contents of index register 2 are 117, the number stored in location 2117 is added into the accumulator when the ADD instruction is executed. However, location 01000 still contains the instruction ADD 2000 in its original form. Address 02117 is called the effective address.

The basic addressable unit is the word (two bytes, 16 bits), although certain instructions do reference bytes or double-words. The contents of the effective address is called the Effective Word (EFW). Words in storage are consecutively numbered starting with 0. The 15-bit address field accommodates a maximum of 32,768 words. When only a part of the maximum storage capacity is available in a given installation, the available storage is continguously addressable from 0.

2.3 (Cont.)

A non-existent memory trap occurs when any effective address points to a location beyond the installed capacity. The EFW of such an address is zero. The invalid address is recognized and a program trap occurs with bit 4 of the Program Status double-Word (PSW) set.

2.3.1 Addressing Exceptions

When the effective address is 0 or 1 or the R bits of the instruction are either 0 or 1, i.e., the PSW, special considerations apply. Refer also to section 2.5.

Address 0 may not be arbitrarily changed. Any attempt to modify bits 10 through 12, the RG section, is ignored. All bits may be read, bits other than 10 through 12 may be written.

Address 1 may not be either read or modified under program control. Any instruction which generates an address of 1 (EFA or R) causes an address exception trap and causes the Program Counter (PC) to be unpredictably modified.

2.4 General Registers

Each level of priority contains a set of 16 general registers, a Register Group (RG), 2 of which may be used as accumulators and index registers, 4 of which may be used just as accumulators and the Program Status double-Word. Two of these words, the Program Status double-Word (PSW) occupy registers 0 and 1. These registers occupy field 0 words 0 to 7 in the memory space as well as the R bits in the instruction; hence, register-to-register instructions are possible. The registers may be stored, loaded, added into, etc., depending on the operation code of the particular instruction used. The second group of 8 registers contains the trap locations for unimplemented EOP instructions, push-down words, and hardware traps. These may be modified or read as memory words but are not explicitly referenced as accumulators. In Model II, the first 8 registers may be fast (flip-flop) registers instead of core memory as in Model I.

register	use
0 R_0	status word, contains condition code, etc.
1 R_1	status word, contains program counter (PC) of the currently active process
2 R_2	accumulator, subroutine linkage register, or secondary index
3 R_3	accumulator or main index register
4 R_4	accumulator
5 R_5	accumulator
6 R_6	accumulator
7 R_7	accumulator
8_{10} 10_8	EOP, receives the updated program counter
9_{10} 11_8	EOP, receives instruction itself
10_{10} 12_8	EOP, receives effective address

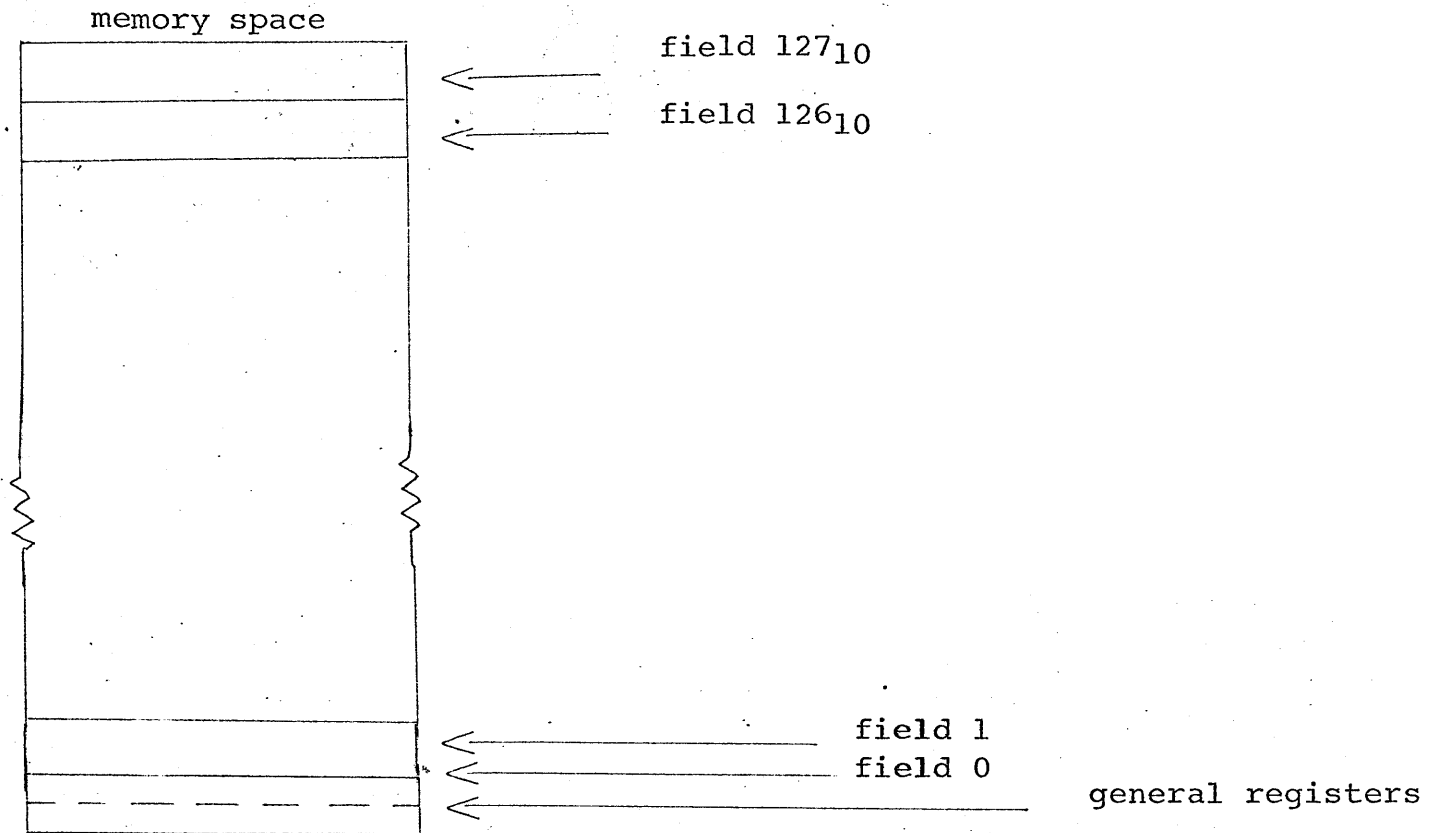
2.4 (Cont.)

register	use
11 ₁₀ 13 ₈	EOP, contains the entry point into the EOP handler, loaded into PC
12 ₁₀ 14 ₈	contains the push-down pointer
13 ₁₀ 15 ₈	contains the push-down counter
14 ₁₀ 16 ₈	TRAP, receives the program counter
15 ₁₀ 17 ₈	TRAP, contains the entry point into the TRAP handler, loaded into PC

For each level of machine priority, both background and IO, there exists a set of general registers; in addition, the hardware insures that the applicable set is available at locations 0-15₁₀ in (apparent) memory address space. Thus, the general registers need not be stored and restored during interrupts. (See section 2.7.)

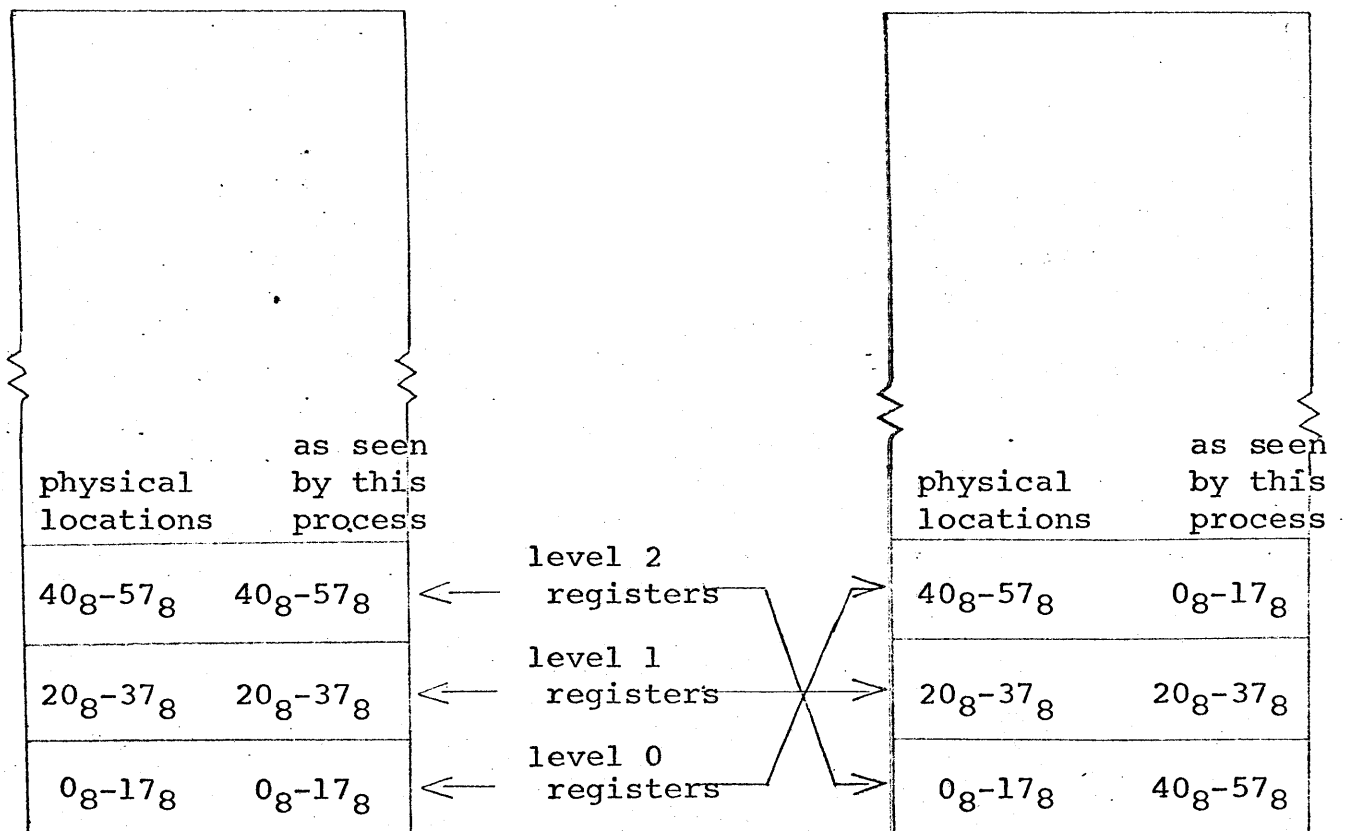
The set of general registers map onto the main memory in field 0. The following figure shows the entire memory space; the figure on the right is an exploded view of physical memory. Apparent memory is the memory space as seen by the running process; this differs from physical memory in the location of its general registers as is shown for a priority level 2 process in the bottom figure. The hardware operates as follows:

- a. whenever an address in the range of 0-15₁₀ is encountered, the RG bits are added to the address in bit positions 9-11₁₀.
- b. whenever bits 9-11 of the address are equal to the RG bits and address bits 1-8 are zeros, bits 9-11 of the address are cleared.



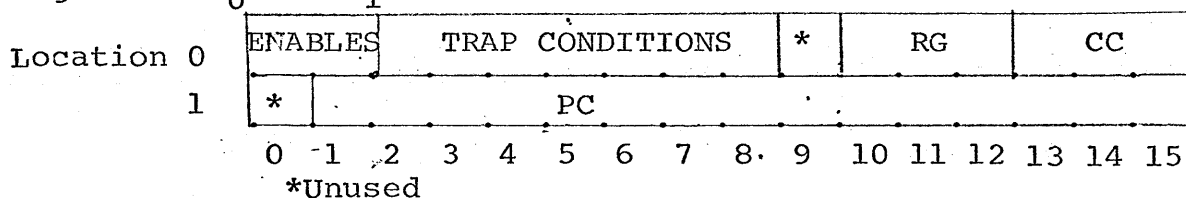
field 0 as seen
by level 0 process

field 0 as seen
by level 2 process



2.5 Program Status Word

The collection of bits that constitute the state of the processor between instructions is called, collectively, the Program Status Word (PSW). This state word occupies the double-word at memory locations 0 and 1 of the active process, corresponding to general registers R_0 and R_1 .



Bit(s)	Definition
0	Arithmetic trap enable (bit 2)
1	Error trap enable (bits 3-6)
2	Arithmetic (add, divide, shift, floating, etc.); enabled if bit 0 = 1
3	Push-down list error; enabled if bit 1 = 1
4	Non-existent memory (reference to an address not in the memory system); enabled if bit 1 = 1
5	Address exception; enabled if bit 1 = 1
6	IO error (IO device did not respond to IO instruction or a bus parity occurred); enabled if bit 1 = 1
7	Privileged instruction (attempt to execute a system instruction while in user mode)
8	Read only violation (attempt to write into a protected memory area)
9	Unused
10-12	Priority of active process (current <u>Register Group</u>)
13	Condition code bit 0

2.5 (Cont.)

Bit(s)	Definition
14	Condition code bit 1
15	Condition code bit 2
16	Unused
17-31	Program counter of active process

2.5.1 Traps

Program status word bits 0-8 constitute the trap indicators and the error trap enable bits; a trap occurs when an unusual condition is detected by the hardware. The trap source bit, and the new register group is given in the table below. Refer to the section on the protection feature for further explanation of the privileged instruction and protection violation traps.

During a trap, the program counter (usually points to the instruction causing the error source) is stored in register 14₁₀, a new address, contained in register 15₁₀, is loaded into the program counter and the appropriate PSW trap bit is set. The trap enable bits (0, 1) are then cleared.

In the case of a protection violation trap (bits 7-8) where priority changes, the change in register groups occurs before the PC is stored and reloaded.

PSW bit	Source	New Priority and RG
2	Arithmetic Error	Same
	ADD: Magnitude of sum greater than register capacity	
	SUB: Magnitude of dif- ference greater than register capacity	
	DIV, LDIV: Magnitude of quotient greater than register capacity	

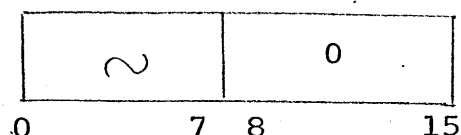
2.5.1 (Cont.)

PSW bit	Source	New Priority and RG
---------	--------	---------------------

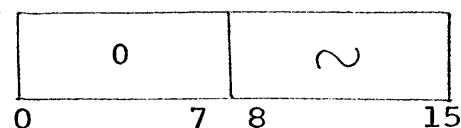
SHFT: (Arithmetic left shift only) sign bit (0) changed during shift

3	Push-Down List Error	Same
---	----------------------	------

Execution of PUSH type instruction with counter word of form



Execution of POP type instruction with counter word of form



(See section 2.6.6)

4	Non-existent Memory	Same
---	---------------------	------

Attempt to address a word not available in the memory system, i.e., no memory responds to a processor request. In a time-sharing environment, this error indicates an attempt to use a page whose assignment was never requested (see Section 2.8)

5	Address Exception	Same
---	-------------------	------

Attempt to reference location 1, the program counter of the currently active process. The effect of instruction execution depends upon the model. The contents of the PC may be lost.

2.5.1 (Cont.)

PSW bit	Source	New Priority and RG
6	IO Error	Same
	The IO device addressed by an IO class instruction did not respond (see Section 2.6.7) or a parity error was detected on the IO bus (optional).	
7	Privileged Instruction	Monitor
	Attempt by user mode (see protection system Section 2.8) program to execute an IO class instruction.	
8	Read-Only Violation	Monitor
	Attempt by the user mode program to write into a read-only page (see Section 2.8).	

2.5.2 Condition Codes

The condition codes (PSW bits 13, 14, 15) are used to determine conditional branches as the result of arithmetic and certain other operations. The codes are normally not changed during load, store, and branch instructions; the appendix contains an instruction list which indicates which instructions effect these PSW bits.

These bits, when changed, reflect the result of the operation just performed; previous condition code information is lost. Hence, the code bits must be tested before the next instruction which changes these bits is executed.

<u>BIT</u>	<u>Normal Meaning</u>
13	Carry during add, borrow during subtract, end bit for certain rotate operations
14	Negative result
15	Nonzero result

2.6 Instructions

Instructions may be divided into two groups, basic and extended. The basic instructions appear in both models and may be in either long or short format. Extended instructions are implemented in Model II, they trap when executed in Model I; extended instructions exist in long format only. The instruction class is determined by the three Op Code bits (0,1,2) of the instruction word. EOP (extended) instructions are characterized by a 110 pattern in the Op Code and the specific operation in the D_1 bits.

Instructions may also be classified by the type of the operand effected. These include:

<u>Class:</u>	<u>Operand:</u>
arithmetic	signed words
logical	unsigned words
floating	floating point double/ quaduple words
branch	address pointers
IO	IO system

2.6.1 Basic Instructions

Class	OP	mnem	Definition
load	0	LDA	<u>Loa</u> D specified <u>Acc</u> umulator (R) with the effective word; the condition code bits remain unchanged.
store	1	STA	<u>ST</u> ore specified <u>Acc</u> umulator (R) into memory at the effective address; the condition code bits remain unchanged.
and	2	AND	<u>AND</u> specified accumulator (R) with the effective word, place result in specified accumulator; condition code 0 remains unchanged 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise
add	3	ADD	<u>ADD</u> contents of specified accumulator (R) with the effective word following the rules of two's complement arithmetic, place result in specified accumulator; condition code 0 set if carry out of bit 0, cleared otherwise 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise
			The arithmetic error bit (bit 2 of the PSW) is set if the sign of the result does not agree with the signs of the operands (add overflow).
branch	4		General conditional branch and subroutine linkage instruction; R bits specify particular operation. If the branch is taken, the effective address is loaded into the program counter. When R = 7 the program counter, updated to point to the instruction following the branch, is saved in accumulator/index register 2; the previous contents of 2 are lost. The condition code bits remain unchanged for all branch instructions.

2.6.1 (Cont.)

Class	OP	mnem	Definition
			R Condition
		BCN	0 branch if condition code bit 0 set (<u>B</u> branch if <u>C</u> arry <u>N</u> onzero)
		BM	1 branch if condition code bit 1 set (<u>B</u> branch if <u>M</u> inus result)
		BN	2 branch if condition code bit 2 set (<u>B</u> branch if <u>N</u> onzero result)
		B	3 unconditional <u>B</u> branch
		BCZ	4 branch if condition code bit 0 <u>NOT</u> set (<u>B</u> branch if <u>C</u> arry <u>Z</u> ero)
		BP	5 branch if condition code bit 1 <u>NOT</u> set (<u>B</u> branch if <u>P</u> ositive result)
		BZ	6 branch if condition code bit 2 <u>NOT</u> set (<u>B</u> branch if <u>Z</u> ero result)
		BAL	7 <u>B</u> branch <u>A</u> nd <u>L</u> ink

2.6.1 (Cont.)

A subroutine is a program segment that may be used many times during the computation of a program but is written only once in the whole code. As the computer proceeds down the main program the control will occasionally jump to this subroutine and then, after doing the subroutine, will jump back to the main program where it let off. This 'detour from the main program through the subroutine may occur several times during the computation of the program. Hence a subroutine must have an entrance, a way of getting into it, and an exit, a way of getting out of it. Each time an entrance is made to a subroutine, some initial conditions must be set up that are characteristic of the place in the main program from which the entrance was made. For instance, if the subroutine calculates some function, the initial values of the independent variables at that point in the main program must be given to the subroutine. In addition, as an entrance to a subroutine is made, the exit must be set up; i.e., the subroutine must be told where to transfer back to the main program.

BAL is the basic subroutine call instruction in PDP-X and it is used in the following manner:

```

BAL      SUBR      ; CALL

ARG 1          ; FIRST ARGUMENT
.
.
.
ARG N          ; Nth ARGUMENT

---           ; RETURN, INSTRUCTION EXECUTED

              ; WHEN SUBROUTINE DONE, MORE

              ; THAN 1 RETURN IS POSSIBLE JUST

              ; AS THERE MAY BE MORE THAN

              ; 1 ARGUMENT

SUBR:  ---           ; FIRST INSTRUCTION OF SUBROUTINE
```

2.6.1 (Cont.)

```
LDA 4, J (2)    ; PICK UP J + 1th ( J ≤ N)  
---            ; ARGUMENT  
B      N (2)    ; RETURN TO CALLING ROUTINE
```

In order to nest subroutines, the subroutine linkage register must be stored in a temporary storage memory word, usually within the subroutine.

If the arguments were addresses, rather than data words, the instruction to pick up data would need to specify indirect addressing, i.e.:

```
LDA 4, @ J (2)
```

2.6.1 (Cont.)

Class	OP	mnem	Definition
modify	5		General memory modification instruction; the R bits specify a particular operation. Condition code bit 0 is changed only by the two rotate instructions (R = 4,5). Condition bits 1 and 2 are set as follows for <u>all</u> modify instructions: 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise Note: in short format these instructions may modify the accumulators or other general registers.
			<u>R Operation</u>
TST	0		<u>TeST</u> , no operation but condition code bits 1 and 2 are set to reflect the state of the effective word.
COM	1		logical <u>COM</u> plement, the effective word is complemented on a bit-by-bit basis.
INC	2		<u>INC</u> rement, one is added to the effective word.
NEG	3		<u>NEG</u> ate, the effective word is negated (complemented then incremented).
RR	4		<u>Ro</u> late <u>R</u> ight, the effective word and condition code bit 0 are rotated together as a 17-bit register one place to the right, loading condition code bit 0 from bit 15 and bit 0 of the memory word from condition code bit 0. (See Section 2.6.3 rotate with CCO.)
RL	5		<u>Ro</u> late <u>L</u> eft, the effective word and condition code bit 0 are rotated left together as a 17-bit register, loading condition code bit 0 from bit 0 of the memory word and bit 15 of the memory word from condition code bit 0. (See section 2.6.3 rotate with CCO.)

2.6.1 (Cont.)

<u>Class</u>	<u>OP</u>	<u>mnem</u>	<u>Definition</u>
			<u>R Operation</u>
		SWP	6 <u>SWaP</u> bytes, the left and right bytes of the effective word are interchanged.
		CLR	7 <u>CLear</u> , the effective word is set to a zero.

2.6.2 Extended Operation Class

Class	OP	mnem	Definition
EOP	6		Extended Operation code class; forced long format; D_1 bits specify particular operation to be performed. The effect on the condition code bits depends upon the particular operation performed.

D_1 codes 0 through 63_{10} are reserved for 64_{10} programmed operators. These codes specify UUO's (UnUsed Operation codes). Codes 0 through 31_{10} are reserved for user program/monitor communication since they can function as protected entry points. (See Section 2.8 on protection features.)

If the operation specified has not been implemented in the machine or if it is UUO, a trap occurs as follows:

location 8_{10} receives the updated program counter

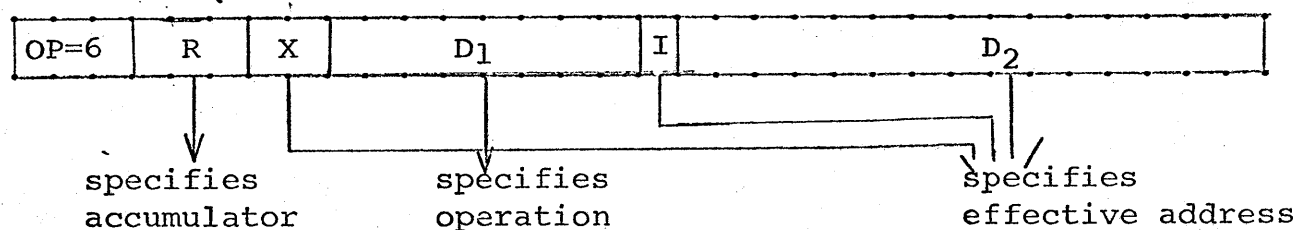
9 EOP instruction

10 effective address

11 contains the entry point into the EOP handler. This word is loaded into the program counter.

Note: the effective address must be in the allowable range or bit 4 of the PSW will be set and a trap may occur.

EOP Class Instruction Double-word:



2.6.3 Extended Arithmetic Group

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
112	SUB	<u>SUB</u>tract the effective word from the contents of the specified accumulator (R) following the rules of two's complement arithmetic; place result in the specified accumulator. condition code bit 0 set if carry out of bit 0, cleared otherwise 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise The arithmetic error bit (bit 2 of the PSW) may be set as in ADD.
101	MUL	<u>MUL</u>tiply. The effective word is algebraically multiplied by the low order word of the double-word specified by R. If R is even, the double-word product replaces the double-word at R and is properly signed. If the specified accumulator (R) is odd, the high order part of the double-word product is discarded. The multiplier and multiplicand are taken to be signed. No overflow is possible. The condition codes remain unchanged. Example: • MUL 4, 300 Multiplies the contents of register 5 by the contents of register 300 leaving the double-precision result in register 4 (high-order) and in register 5 (low-order). MUL 5, 300 Multiplies the contents of register 5 by the contents of register 300 leaving the low-order result in register 5. The high-order result is discarded.

2.6.3 Extended Arithmetic Group

D ₁	mnem	Definition
100	LMUL	Logical <u>M</u> ultiply. The effective word is logically multiplied by the multiplier as described for MUL, above, except that the operands are taken to be positive 16-bit logical quantities.
103	DIV	<u>D</u> IVide. The signed arithmetic double-word beginning at the specified accumulator (R) is algebraically divided by the effective word. The signed quotient developed replaces the low-order word of the dividend, the remainder, signed the same as the dividend replaces its high-order part. When the relative magnitude of dividend and divisor is such that the quotient cannot be expressed by a 16-bit signed integer, a divide overflow trap occurs, no division takes place, and the dividend may be lost.

If R is odd, the corresponding even word is filled to produce a two's complement double-precision dividend. The divide then proceeds normally. This is equivalent to an integer division.

Examples:

DIV 4, 1137

Divide the signed (two's complement) double-word in locations 4,5 by the contents of register 1137. Place the remainder in location 4 and the quotient in location 5.

DIV 5, 1137

Divide the signed (two's complement) integer in location 5 by the contents of register 1137. Place the remainder in location 4 and quotient in location 5.

2.6.3 Extended Arithmetic Group

D ₁	mnem	Definition
----------------	------	------------

The condition codes remain unchanged.

102	LDIV	<u>Logical DIVide</u> . The logical double-word beginning at the selected register (R) is divided by the effective word. The operation performed is the same as DIV except that the operands are treated as 16 and 32-bit positive integers. The results are also 16-bit positive integers.
-----	------	---

The condition codes remain unchanged.

If R is odd, the corresponding even word is filled with zeros to produce a double-precision logical word. The divide then proceeds normally.

111	CMP	Algebraic <u>CoMPare</u> . The specified accumulator (R) and the effective word are algebraically compared as signed integers. Neither accumulator nor effective word are changed, but the condition code is set according to the result. condition code bit 0 remains unchanged 1 set if register < memory word, cleared if ≥ memory word 2 set if register ≠ memory word, cleared otherwise
-----	-----	--

110	LCMP	<u>Logical CoMPare</u> . The specified accumulator (R) and the effective word are logically compared as 16-bit positive integers. Operation proceeds and condition code is set as in CMP.
-----	------	---

2.6.3 (Cont.)

D ₁	mnem	Definition
113	SHFT	<u>SHiFT</u>. The contents of the specified accumulator (R) is shifted as indicated by the effective word. The right half (byte) of the effective word is used as a signed shift count. A positive value indicates a left shift. Bits 6 and 7 of the effective word indicate the type of shift to be performed. (Condition code bit 0 is changed only by the Rotate with CCO mode.) Condition code bits 1 and 2 are set as follows for all shift instructions: 1 if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise 00 - arithmetic shift - performs two's complement multiplication by powers of two. The sign is unchanged. When going to the right, the sign is shifted into bit 1. Ones or zeros leaving bit 15 are lost. When going to the left, zeros enter bit 15. The arithmetic error bit (bit 2 of the PSW) is set if, during shifting, the sign bit is changed (left shift only). 01 - rotate with CCO - bits leaving one end enter condition code bit 0. CCO enters at the other end. 10 - rotate - bits leaving one end enter at the other end. 11 - logical shift - bits leaving one end are lost and zeros enter the other end.

Shift Control Word

SM = Shift Mode

SC = Signed Shift Count

00 - Arithmetic Shift

>0 : Left

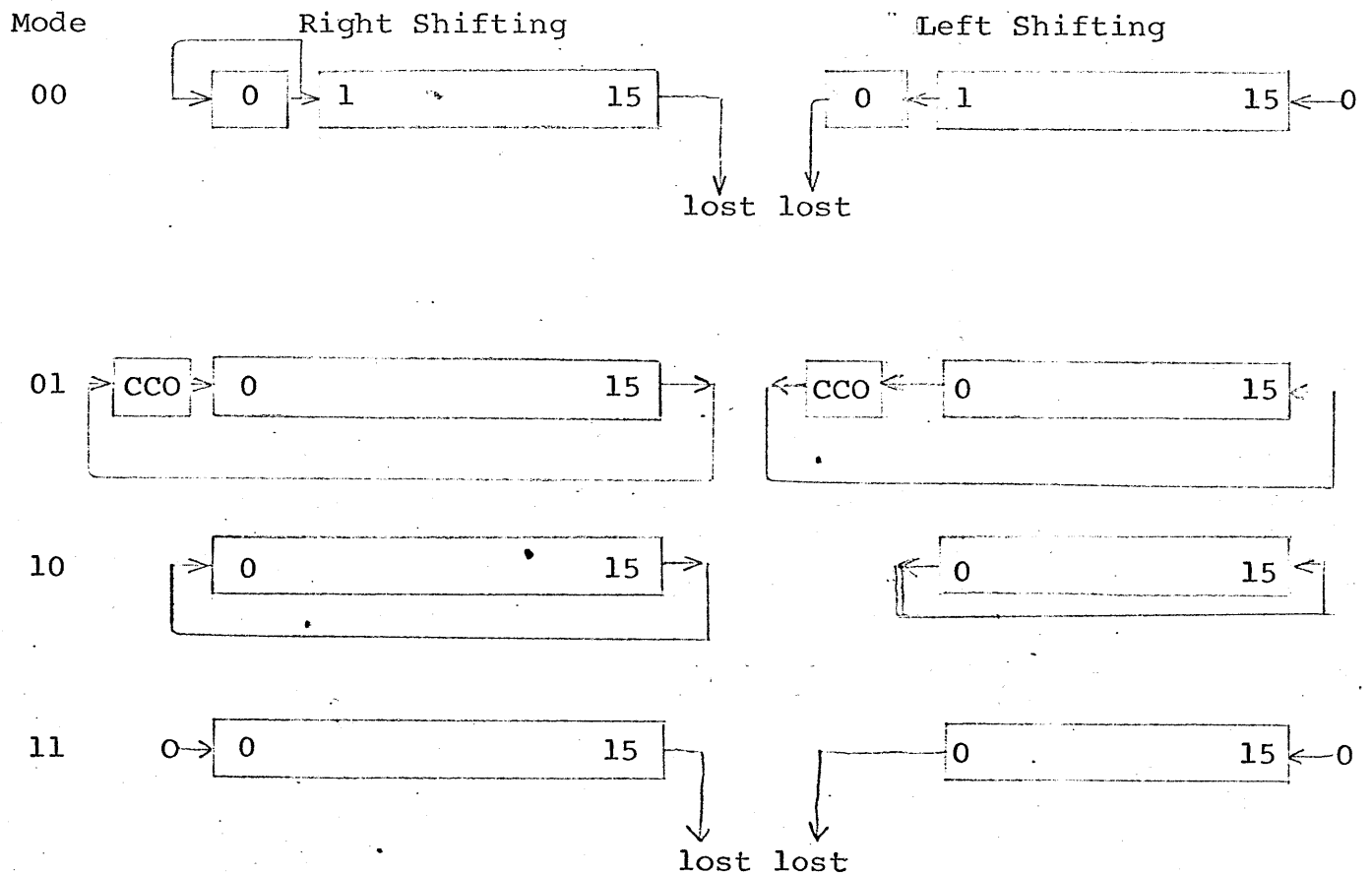
01 - Rotate with CCO

<0 : Right

10 - Rotate

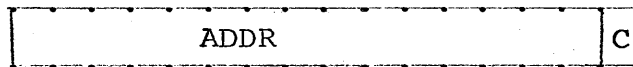
=0 : No Shift

11 - Logical Shift



2.6.4 Character Group

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
114	LDC	<u>L</u> oad <u>C</u> haracter. The effective word is used as a character pointer to locate an 8-bit byte. This byte is loaded into the right half of the specified accumulator (R). The left half is cleared. The addressed memory word is left unchanged. The condition code remains unchanged.
115	STC	<u>S</u> to <u>r</u> e <u>C</u> haracter. The effective word is used as a character pointer to locate an 8-bit byte. The right half of the specified accumulator (R) is stored at the indicated character position. The other half of the addressed word is unaltered. The content of R remains unchanged. The condition code remains unchanged.

CHARACTER (BYTE) POINTER WORD

ADDR = address C = character selection
 0-right byte
 1-left byte

Note: This is identical to the BA word in the multiplexor channel.

2.6.5 Logical Compare and Modify Group

Bits of the specified accumulator (R) that are masked by bits of a memory word may be tested and/or modified to determine a conditional branch. The bits to be tested and/or modified are selected by ones in the effective word. Condition code bit 2 is cleared if all of the selected bits of the specified accumulator (R) are zero; otherwise, it is set to a one. Condition code bit 1 is set to a one if bit 0 is selected and bit 0 of the specified accumulator (R) is a one; otherwise, it is cleared. The selected bits of the specified accumulator (R) are then modified or not depending upon the operation being performed. Condition code bit 0 is undisturbed.

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
104	TSTN	<u>TeST</u> but change <u>N</u> othing. Test the content of the specified accumulator (R) against the effective word. Set condition code bits 1 and 2 according to the result.
105	TSTZ	<u>TeST</u> and <u>Z</u> ero selected bits. Test the content of the specified accumulator against the effective word. Set condition code bits 1 and 2 according to the results. Clear selected bits in the specified accumulator (R) (i.e., for every one in the effective word, clear the corresponding bit in the specified accumulator). This performs the logical extract operation.
106	TSTO	<u>TeST</u> and set selected bits to <u>O</u> nes. Test the content of the specified accumulator (R) with the effective word. Set condition code bits 1 and 2 according to the result. Set selected bits in the specified accumulator (R) (i.e., for every one in the effective word, set the corresponding bit in the specified accumulator). This performs the logical function inclusive or.

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
107	TSTC	<u>Te</u> <u>ST</u> and <u>C</u> omplement selected bits. Test the content of the specified accumulator with the effective word. Set condition code bits 1 and 2 according to the result. Complement selected bits in the specified accumulator (R) (i.e., for every one in the effective word, complement the corresponding bit in the specified accumulator). This performs the logical function exclusive or.

2.6.6 Push-Down Group

Words are pushed into (popped from) memory under control of pointer and counter words. General register 14_8 is the push-down pointer. Its contents indicate the first free location on the push-down list. General register 15_8 is the push-down counter which insures that the space allotted to the list is not exceeded. Maximum push-down list length is 256 words. Exceeding the list capacity while either storing or retrieving causes a push-down error trap with bit 3 of the PSW set. The registers are incremented/decremented on each instruction.

The list control registers are initialized by placing the starting (lowest) address of the list in the pointer word and the length of the list (positive integer $\leq 255_{10}$) in the counter word. During a push-type instruction, the pointer is used then incremented; the counter left byte is incremented; the right byte decremented. During a POP-type instruction, the pointer is decremented then used; the counter left byte is decremented, the right byte is incremented. A trap occurs whenever a byte whose value is 0 is decremented.

All 8 (specified by the R field) Push/Pop class instructions manipulate the pointer and counter words. PUL and POL move 2 words onto or off of the push-down list.

The Push-Down List (PDL) pointer (location 14_8) always points to the next free cell on the push-down list. The right-hand byte of the push-down list counter (bits 8-15 of location 15_8) contains a count of the number of pushes that may be executed before the list overflows. The left-hand byte of the push-down list counter (bits 0-7 of location 15_8) contains a count of the number of pops that may be executed before the list underflows.

The PDL pointer is initialized to point to the first cell of the push-down list, the right half of the PDL counter is initialized to contain the length of the list ($\leq 255_{10}$) and the left half is initialized to contain zero.

2.6.6 (Cont.)

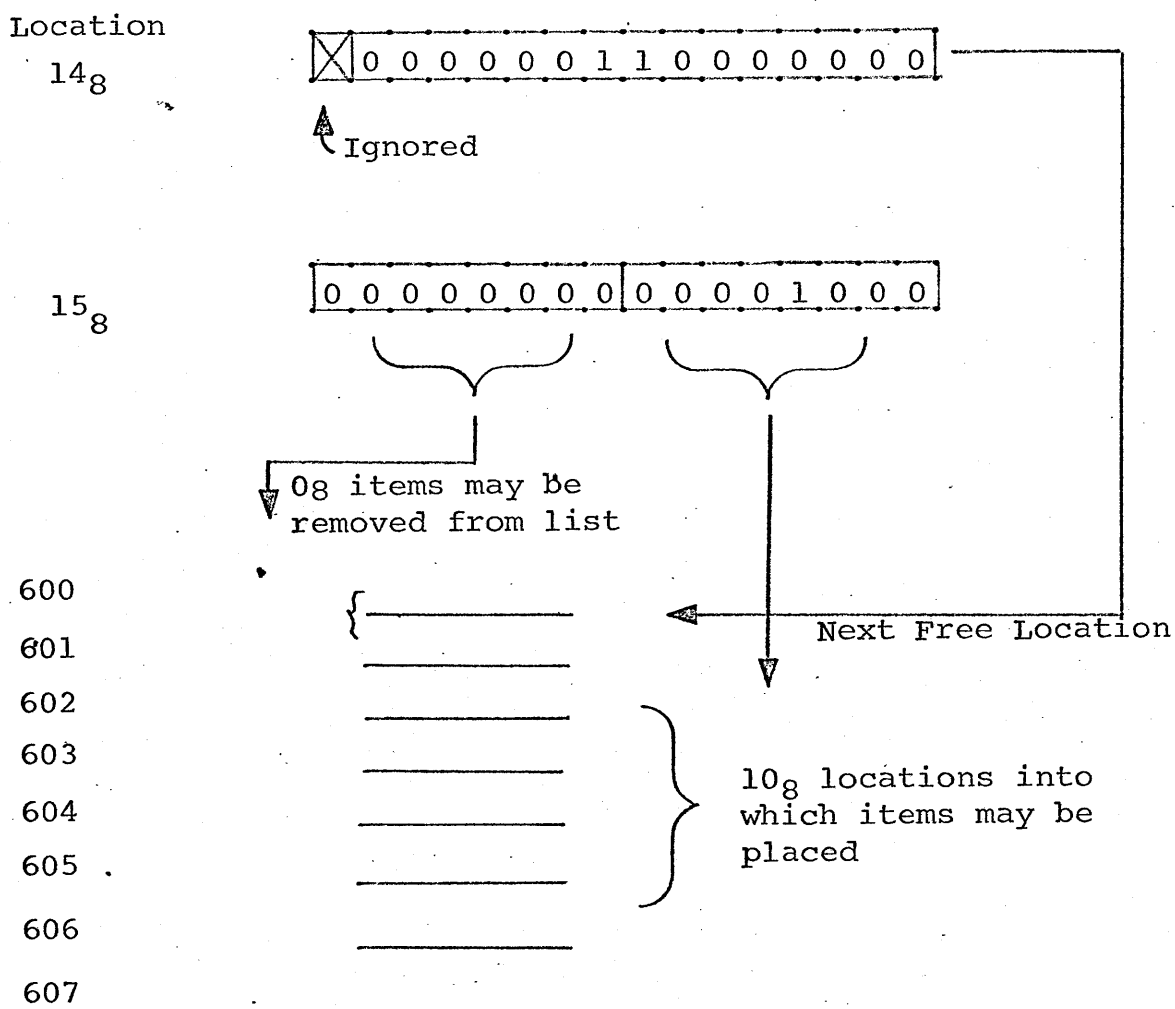
For example, if the PDL were to start at location 600_8 and is 10_8 locations long, the initialization sequence could be:

```

LDA 4,    [600]
STA 4,     14      ; STORE POINTER
LDA 4,     [10]
STA 4,     15      ; STORE COUNTER

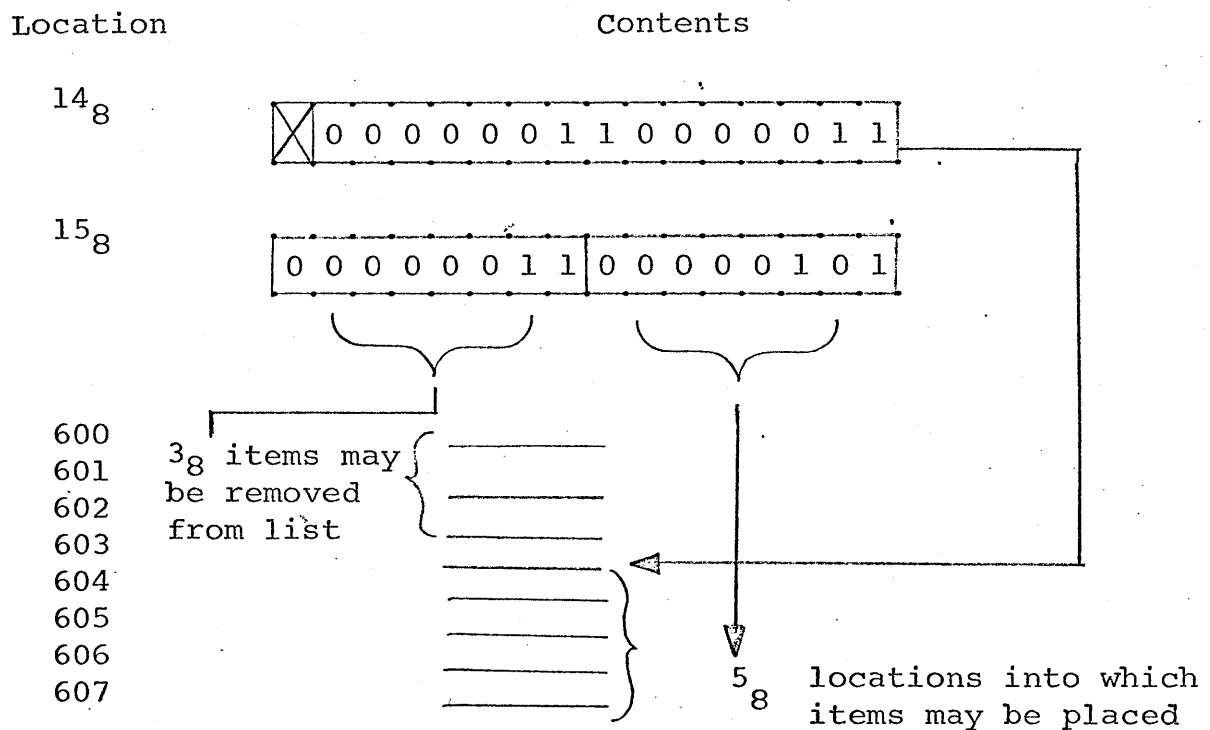
```

The push-down list counter and pointer would then be:



2.6.6 (Cont.)

After the execution of instructions that placed three items on the PDL, the situation would be:



D ₁	mnem	R	Definition
116	PUC	0	<u>P</u> U <u>s</u> h <u>C</u> ount. The pointer and counter are modified as for a push type instruction but no data is put onto the list.
	PUSH	1	<u>P</u> U <u>s</u> h. The effective word is placed in the next location on the push-down list.
	PUB	2	<u>P</u> U <u>s</u> h and <u>B</u> ran <u>c</u> h. The program counter is placed in the next location on the push-down list. The effective address replaces the program counter.
	PUL	3	<u>P</u> U <u>s</u> h, <u>B</u> ran <u>c</u> h and <u>L</u> ink. The subroutine linkage register is placed in the next location on the push-down list.

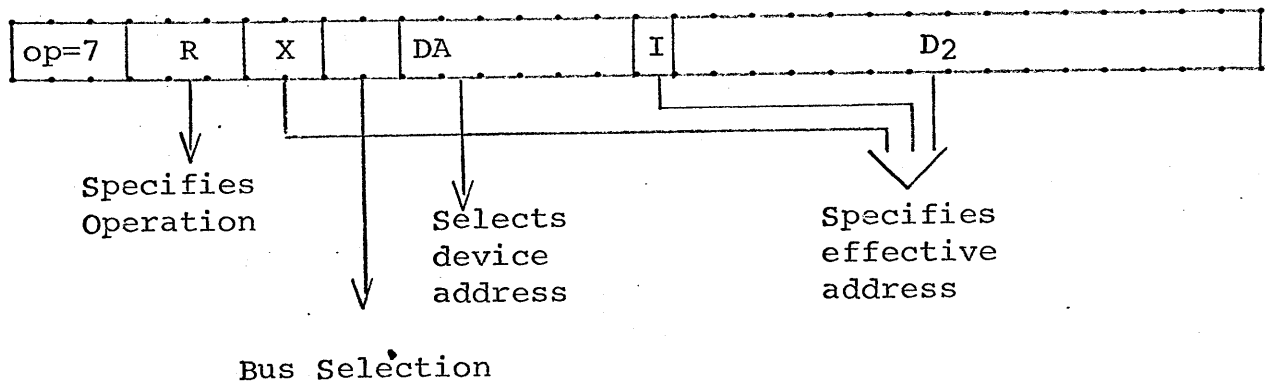
2.6.6 (Cont.)

<u>D₁</u>	<u>mnem</u>	<u>R</u>	<u>Definition</u>
			The program counter is placed in the subroutine linkage register. The program counter is placed in the next location on the push-down list. The effective address replaces the program counter.
	POC	4	<u>POp Count</u> . The pointer and counter are modified as for a POP type instruction but no data is removed from the list.
	POP	5	<u>POP</u> . The last word placed on the push-down list is moved to the content of the effective address.
	POB	6	<u>POp</u> and <u>Branch</u> . The sum of the effective word and the last word placed on the push-down list replaces the program counter. This is the return instruction for PUB.
	POL	7	<u>POp</u> , <u>branch</u> , and <u>Link</u> . The sum of the effective word and the last word placed on the push-down list replaces the program counter. Then, the new last word on the push-down list is then placed in the subroutine linkage register. This is the return instruction for PUL.

PUB and PUL may be used as generalized subroutine calls; PUB when no arguments are passed, PUL when arguments are passed. These calling sequences are "nestable" and re-cursive allowing for re-entrance. The corresponding return instructions are POB and POL. Notice the PUL places two words on the push-down list and that POL removes two words from the push-down list. The pointer and counter words are incremented (decremented) two times during the execution of these instructions.

2.6.7 IO Instructions

Class	OP	mnem	Definition
IO	7		Input/Output instruction class. The R bits of an IO instruction (always long format) specify one of 8 types of operation. The D₁ field is interpreted depending upon the type of operation. Normally, 6 bits are used to select a particular device connected to the IO bus, two bits (optionally) select one of four busses (see diagram below). In an IOD class instruction, D₁ is used to augment the operation code and R fields of the instruction to specify a specific internal function.

IO Class Instruction Double-word

2.6.7 (Cont.)

Class	OP	mnem	Definition
-------	----	------	------------

If no device responds to the IO instruction, bit 6 of the Program Status double-Word is set indicating an IO error. If the Error Trap Enable bit (1) is on, a trap will occur. (See section 2.5.) This error may be caused by a programming error, a hardware malfunction, or the addressed device may be disconnected.

The only IO class instruction which may set condition codes is IOT:

condition code bit 0 remain unchanged
 1 set if bit 0 of the word resulting from the operation is a 1; cleared otherwise
 2 set if the word resulting from the operation is nonzero; cleared otherwise

The IO instructions may not be executed by a program operating in User mode on a system with the (optional) protection system installed. An attempt to do so will result in bit 7 of the PSW being set and a trap to the monitor will result. (See section 2.5, 2.8)

R Operation

IOR	0	<u>IO Read word.</u> The data buffer of the addressed input device (DA) replaces the right-hand byte of the effective word. The left-hand byte is cleared. If this instruction is used to address a device that can only do output, a zero byte is read.
-----	---	--

2.6.7 (Cont.)

Class	OP	mnem	Definition
			<u>R Operation</u>
			If the addressed input device responds with more than one byte of data, they are placed in memory starting with the byte located in the right half of the effective word. If an odd number of bytes are transferred, the last left-hand byte is left zero.
IORC	1		<u>IO Read Character.</u> The effective word of the IORC instruction is used as a byte pointer that locates a single byte; the data buffer of the addressed input device (DA) replaces the byte so located. The other byte in the 16-bit word is unchanged. If this instruction is used to address a device that can only do output, a zero byte is read. If the addressed input device responds with more than one byte of data, they are placed in memory starting at the located byte. The byte pointer referenced by this instruction (i.e., the effective word) has the format as indicated in section 2.6.4 and is left unchanged.
IOW	2		<u>IO Write Word.</u> The data buffer of the addressed output device (DA) is replaced by the byte located in the right half of the effective word. If this instruction is used to address a device that can only do input, it has no effect. If the addressed output device requests more than one byte of data, they are taken from memory starting with the byte located in the right half of the effective word.

2.6.7 (Cont.)

Class	OP	mnem	Definition
<u>R Operation</u>			
IOWC	3		<u>IO Write Character</u>. The effective word of the IOWC instruction is used as a byte pointer that locates a single byte; the byte so located replaces the data buffer of the selected output device (DA). If this instruction is used to address a device that can only do input, it has no effect. If the addressed output device requests more than one byte of data, they are taken from memory starting with the located byte. The byte pointer referenced by this instruction (i.e., the effective word) has the format as indicated in section 2.6.4 and is left unchanged.
IOS	4		<u>IO read Status</u>. The status register of the addressed device (DA) replaces the right-hand byte of the effective word. The left-hand byte is cleared. If the addressed device responds with two bytes of status, the second byte is placed in the left half of the effective word.
IOC	5		<u>IO Command</u>. The status register of the addressed device (DA) is replaced by the right-hand byte of the effective word. If the device requests a second byte of status, it is taken from the left-hand byte of the effective word.
IOT	6		<u>IO Test status</u>. The status register of the addressed device (DA) is

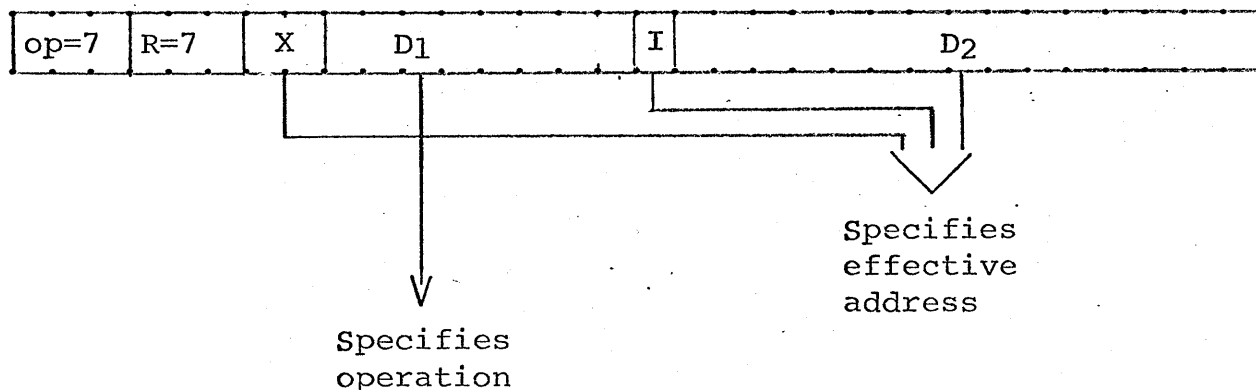
2.6.7 (Cont.)

Class	OP	mnem	Definition
-------	----	------	------------

			R .Operation
--	--	--	--------------

read into the central processor. If the addressed device responds with a second byte, it is placed into the left half; otherwise, the left half is cleared. This resultant 16-bit word is logically ANDed with the effective word. The resulting condition code bits (1,2) may be tested by a conditional branch instruction to determine the state of a selected status register bit. The effective word and addressed status register remain unchanged.

IOD	7	internal <u>IO</u> Device control. The D ₁ field of this class of IO instruction is used to select one of several functions. Instructions with undefined values of D ₁ are treated as no-operations. The IOD instructions are used to change the state of the IO system and the memory protection system. (See section 2.8.4)
-----	---	---

IOD Class Instruction Double-word

2.6.7 (Cont.)

IOD Class (OP = 7, R = 7)

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
0	RIO	<u>Reset IO</u> system; all devices are cleared; no interrupt may occur from any device unless that device is specifically re-initialized. The priority level internal request indicators, external request indicators, inhibit indicators and active indicators are cleared. If the system has the protection feature, active indicator 1 is set and RG is set to 1, otherwise RG is cleared. The instruction should only be executed at the level which will result, i.e., level 0 for machine without protection, level 1 for machine with protection. This instruction is accomplished by transmitting the reset code on the IO bus. All devices, both DEC and customer-designed must use this code to reset to a known, non-operating state, i.e., clear <u>ALL</u> flip-flops in the devices and in the controllers.
1	HLT	<u>HaLT</u> ; the central processor is halted at the completion of this instruction. The effective word is displayed in the console indicators. The program

2.6.7 (Cont..)

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
		counter is updated to point to the next sequential instruction.
2	RCS	<u>Read Console Switches</u> ; the contents of the console switches replace the effective word.
3	WCI	<u>Write Console Indicators</u> ; the effective word replaces the contents of the console indicators.
		(See section 2.7 for a description of the priority system.)
4	PSI	<u>Priority System Inhibit</u> ; the priority of the currently running process is raised by setting an inhibit indicator to the value specified by the effective word (i.e., the content of the effective address). The low order four bits of the effective word is compared with the highest priority level active indicator. If its value is higher than the indicator, then the priority level inhibit indicator corresponding to this value is turned on. If it is less than or equal to the active indicator, no operation results. In either case, the RG bits are not changed.

2.6.7 (Cont.)

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
		This instruction is normally used together with its return, PSC, in calling a subroutine which is not re-entrant. Such cases occur in changing queue parameters of certain interrupt service routines. A typical call is of the form: PSI [N] ; raise priority to N BAL SUBR ; call subroutine ARG ; argument(s) PSC ; restore priority ; N is chosen to be sufficiently high such that no routine entered at a priority level greater than N calls SUBR.
5	PSR	<u>P</u>riority <u>S</u>ystem <u>R</u>quest; the priority level request indicator corresponding to the value of the low order three bits of the effective word is set. An interrupt at that level is requested; this request is handled as if it were an external IO device. When the interrupt is granted, the interrupt address is determined according to the following table:

2.6.7 (Cont.)

D₁ mnem Definition

Priority Level Interrupt Address

0 No interrupt re-
 requested

1 401

2 402

• •

• •

• •

7 407

6 PSC Priority System Clear; the priority level of the currently running process is returned (by clearing an inhibit indicator) to the priority preceding the last PSS instruction issued.

The priority level inhibit indicators are compared with the priority level active indicators. If the highest inhibit level is greater than the highest active level, then that inhibit indicator is turned off.

Note that RG is not changed.

7 PSD Priority System Dismiss; the currently active process is terminated and control is returned to the interrupted process. The highest active level indicator is cleared and the RG bits are set to the value of the new highest active level indicator.

2.6.7 (Cont.)

<u>D₁</u>	<u>mnem</u>	<u>Definition</u>
----------------------	-------------	-------------------

This instruction is normally used at the completion of an interrupt service routine to return control to the interrupted process.

(See also section 2.8.4 for additional IOD class instructions.)

2.7 Priority (Interrupt) System (8 levels optional - Model II_b)

The Priority System (PS) controls the nine (maximum) levels of priority possible in the processor including the main program at the lowest level and the special multiplexor channel at the highest level. The interrupt due to an internal source (PSR instruction) or an external source (IO device) causes the new, appropriate set of general registers to be substituted for the previously operating set. Since the state of the processor is stored and reloaded almost instantaneously, the interrupt service routine can begin immediately. The priority levels are fully nested so that, for example, an interrupt at level 6 would occur into a process running at level 4, but an interrupt at level 2 could never occur into such a process.

Five elements of the hardware are important to the programmer; these include: The Register Group (RG) bits of the PSW, the priority level inhibit indicators, the priority level active indicators, the priority level internal request indicators, and the priority level external request indicators. The register set currently in use is specified by the RG bits. These priority levels are normally used as shown in the following table:

Level (RG)	Use	Traps
0	Main Program	Arithmetic
1	Monitor, Lowest Hardware Device	Monitor
2-6	Device	
7	Highest Hardware Device	Machine Check

The priority level active indicators are a set of seven flip-flops, one for each of levels 1-7, that determine which levels have active processes associated with them. These indicators are set by internal or external requests and are cleared only by the PSD instruction. If the main program were interrupted first by a level 2 device, and then by a level

2.7 (Cont.)

6 device before the former device was completely serviced, then the indicators for levels 2 and 6 would be on. No interrupt may occur at a level less than or equal to the highest active level as stored in these indicators.

The priority level inhibit indicators are a set of eight flip-flops, one for each of levels 1-8, that inhibit interrupts at all levels equal to or lower than the highest inhibited level. These indicators are set by the PSI instruction and cleared by the PSC instruction. If either an inhibit indicator or an active indicator is set, only interrupts with a priority above the highest indicator may occur.

The priority level internal request indicators are associated only with the PSR instruction. Execution of this instruction causes the specified (1-7) bit of the request indicators to be set. These indicators are treated as devices requesting interrupts at each of the seven external priority levels. Associated with each level is an interrupt address, contents of which are loaded into the program counter when the priority of the priority level request indicator is higher than any active process and is not inhibited. When the interrupt occurs, the request indicator is cleared and the corresponding active indicator is set.

<u>Level (RG)</u>	<u>Interrupt Address</u>
1	401
2	402
3	403
4	404
5	405
6	406
7	407

The priority level external request indicators (1-8) are set by external devices requesting interrupt service. When an interrupt occurs, the RG bits of the new PSW are set to the new priority level (unless it was a special multiplexor channel request at level 8)

2.7 (Cont.)

and the appropriate active indicator is set. Subsequent instructions will use the interrupt process PC and register set. The interrupt is cleared with a PSD instruction which clears the highest active level indicator and loads the RG register with the value of the new highest-active level indicator.

If the request was a multiplexor channel request at level 8, the input or output operation is performed and the interrupted program continued.

2.8 Protection Feature (optional - Model IIC)

The protection feature consists of three items to allow multi-user operation of PDP-X: (a) instruction protection, (b) memory protection, (c) monitor calls. The protection feature is only available on Model II processors with the Priority System option.

2.8.1 Instruction Protection

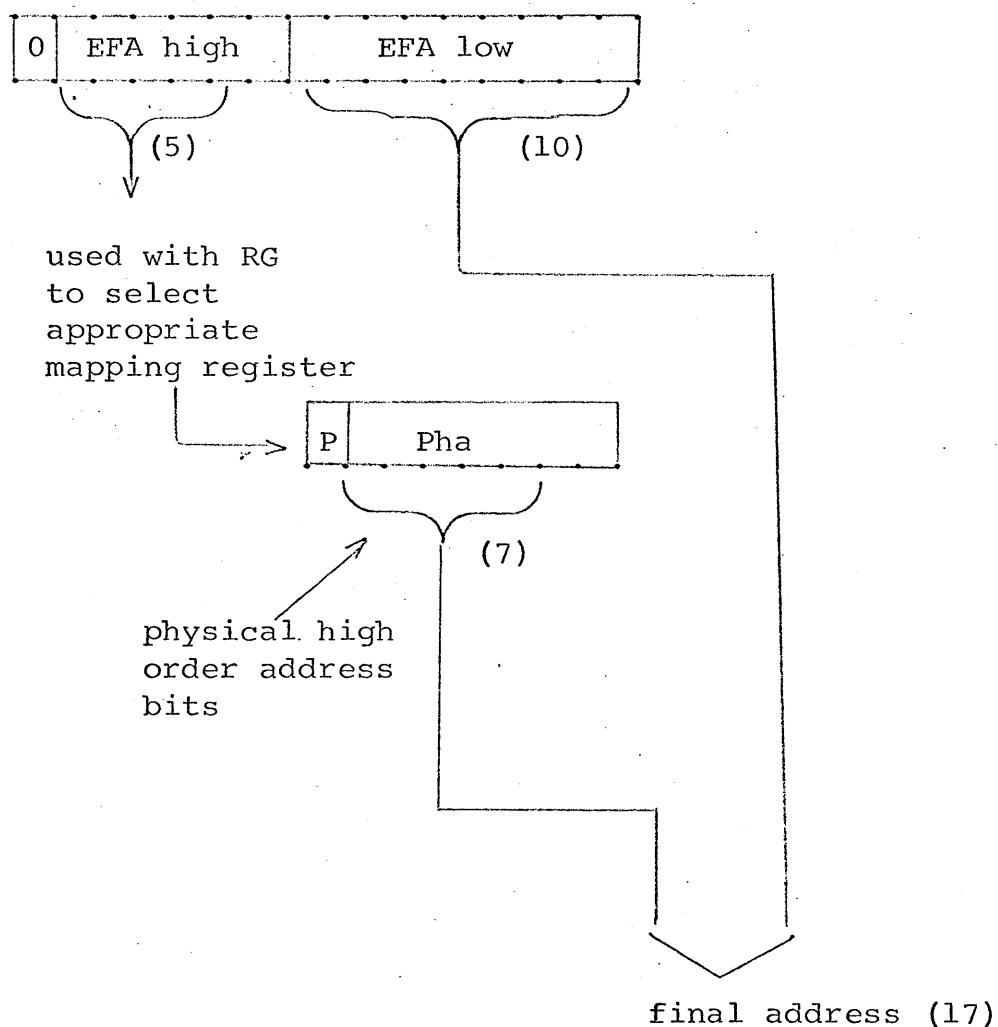
With the protection feature, PDP-X runs in two distinct modes: user mode and monitor mode. The PDP-X is in user mode when RG = 0. In user mode, the program may execute all instructions except the IO class instructions, thus the user program may in no way alter the state of (a) the IO system, (b) the Priority System, (c) the protection system. Monitor mode is entered whenever RG becomes nonzero, in other words, whenever priority is raised to levels 1-7. In monitor mode, all instruction classes may be executed.

If a program in user mode executes an IO instruction, the priority is raised to level 1 (RG set to 1), bit 7 of the PSW is set and a trap to the monitor occurs. Similarly, if the user mode program violates memory protection (see 2.8.2), a similar sequence will occur setting bit 8 of the PSW.

2.8.2 Memory Protection

Memory protection is accomplished through memory paging. The high order address bits formed by the processor are not sent directly to the memory system; instead, together with the RG bits of the PSW, they are used to select one of several mapping registers. The contents of the selected mapping register are sent to the memory system along with the unaltered low order bits.

Each mapping register contains one (read only) control bit and seven address bits which substitute for the processor-generated five high order bits. Page size is, therefore, 1K. The diagram on the following page outlines this mapping.



A maximum of 64_{10} protection/mapping registers, each one byte long, may be accommodated. One map (32 mapping registers) is provided for priority level 0 (user mode) which will be referred to as the user map. Another map is provided for the other seven levels (monitor mode) which will be referred to as the monitor map since the monitor program normally runs at priority level 1 and the monitor controlled IO routines run at levels 2-7.

The address of the mapping register is formed as follows:

0 if level 0
1 otherwise

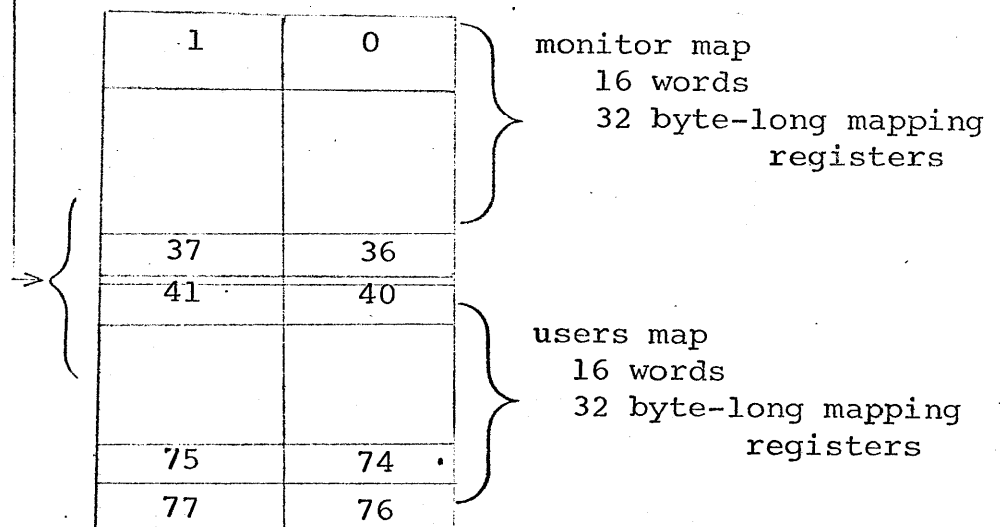
5 high order address bits

mapping word

register selection

0 = right byte

1 = left byte



If the P bit is set and PHA is nonzero, then the specified memory page is taken to be read only and no user mode program may write into that page. An instruction executed in user mode that attempts to write into a read only page causes a read only violation which: (a) raises priority to level 1, (b) sets bit 8 of the PSW, and (c) initiates a trap sequence (see section 2.5.1).

2.8.2 (Cont.)

If the P bit is set and PHA is zero, then no memory page has been assigned to the program. A program (user mode or monitor mode) that attempts to reference a word in such a page causes a non-existent memory violation which: (a) sets bit 4 of the PSW, and (b) initiates a trap sequence (see section 2.5.1). Normally, this would indicate a programming error. The user mode program, may be in a situation that requires more memory. In this case, the user program can call the monitor to request additional memory and then continue.

2.8.3 Monitor Calls

If a program running in user mode executes an EOP class instruction with D_1 in the range $0 \leq D_1 \leq 37_8$, the priority is raised to level 1 and the EOP is executed using the monitor's register group. If a user mode program executes an EOP class instruction with D_1 in the range $40_8 \leq D_1 \leq 77_8$, the priority is unaltered and the EOP is executed using the user program's register group (see section 2.6).

This mechanism allows a convenient means for the user program to communicate with the monitor.

2.8.4 Instructions for Memory Protection System

Eight IOD class instructions are added to PDP-X with the addition of the protection option. These instructions allow the program operating in monitor mode to load the user and monitor maps.

<u>D₁</u>	mnem	Definition
10	LML	<u>L</u> oad <u>M</u> onitor map <u>L</u> ow. The 8 words (16 bytes) starting at the effective address are loaded into monitor map registers 0 ₈ -17 ₈ . Monitor map register 0 is always zero and is not changed by this instruction. All other map registers are unaffected.
11	LMH	<u>L</u> oad <u>M</u> onitor map <u>H</u> igh. The 8 words (16 bytes) starting at the effective address are loaded into monitor map registers 20 ₈ -37 ₈ . All other map registers are unaffected.
12	LUL	<u>L</u> oad <u>U</u> ser map <u>L</u> ow. The 8 words (16 bytes) starting at the effective address are loaded into map locations 40 ₈ -57 ₈ (user map locations 0 ₈ -17 ₈). All other map registers are unaffected.
13	LUH	<u>L</u> oad <u>U</u> ser map <u>H</u> igh. The 8 words (16 bytes) starting at the effective address are loaded into map locations 60 ₈ -77 ₈ (user map locations 20 ₈ -37 ₈). All other map registers are unaffected.

2.8.5 Summary

The PDP-X memory protection system is designed to provide the following capability:

- a) efficient memory management and protection in a multi-user environment
- b) re-entrance for systems software packages such as the editor, assembler and compiler

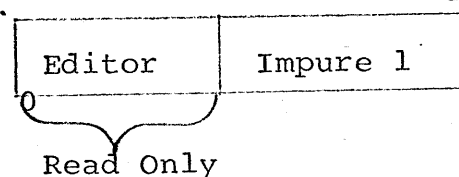
A paging system is employed for memory management and protection. Each user may have a virtual memory space of up to 32K words while the physical memory may contain 128K words. The individual user pages need not be contiguous in the physical memory allowing for the separate maintenance of IO buffers, temporary storage and data. Jobs may be swapped out and a new job swapped in without "shuffling" as is done in PDP-10. IO buffer areas need not be moved in physical space and, hence, they need not be swapped or shuffled.

The major systems programs may be run in a re-entrant environment if they do not modify themselves during the course of execution. Each program will have two parts termed the pure portion (that portion not modified, i.e., instructions and constants) and the impure portion (that portion that is modified, i.e., buffers, data and temporary storage). For several jobs to use the re-entrant program, it is only necessary to have one copy of the pure portion in core. Each job would have its own impure portion.

In this case, the physical space might appear as:

Monitor	Impure 1	Editor	Impure 2		Impure n
---------	----------	--------	----------	-----------	----------

When job 1 is to be run, the user map is arranged so that user virtual memory appears as:



Thus, only one copy of the editor need be resident in core for many users. The impure portions may be swapped in and out.

Paging is not sufficient, however, to accommodate the general case of arbitrarily-shared procedures. This case must be accommodated by well-defined conventions.

3.0 IO System

The PDP-X IO system is both byte and full-word oriented. For devices whose media is no more than a byte wide, the multiplexor channel packs (unpacks) bytes directly into (from) memory, eliminating the need for assembly (disassembly) registers in the devices.

For devices whose natural word is larger than a byte, both the IO instructions and the multiplexor channel transfer several bytes in a burst mode.

Interrupt sources are automatically identified by the basic hardware and the priority levels at which the devices interrupt are fully nested. In addition, no special device hardware is required in order to operate a device on the multiplexor channel; the multiplexor channel hardware generates the same control sequences as do the IOR and IOW instructions.

3.1 Devices and Controllers

The hardware involved in IO operation is logically divided into four parts: IO section (in the Central Processor), IO bus, controller, and device. The IO bus is described in detail below. Controllers and devices are generally different for each type of IO media; from the programming point of view, most controller functions merge with IO device functions.

In all cases, the controller function is to provide the logic and buffering capabilities necessary to operate the associated IO device. Each controller functions only with the IO device for which it is designed, but each controller has standard signal connections with regard to the IO bus. The teletype device (keyboard, printer), for example, connects to the IO bus through teletype controller logic (single character data buffering and interrupt logic). The detailed meaning of the command/status bits read under program control through the IO section from controller type to type, but the general format remains unchanged.

3.2 Modes Of Data Transfer

There are three basically different modes of data transfer available in the IO system: program-controlled, multiplexor channel, and selector channel. All three use the standard IO bus interface; the third provides an additional physical bus interface and additional control logic at the processor end. Maximum data transfer rate for each mode varies with processor model, but the program-controlled rate is always lowest and the selector channel rate highest. In all cases, transfer sequences are initiated by IO instructions issued to the appropriate controller, rather than to the channel.

Program-controlled transfer, while slowest, provides the greatest flexibility. Data may be modified, limit checked, or otherwise monitored as it is transmitted; control sequences required by special purpose or custom-designed IO equipment may be generated. For the slower devices, especially paper tape or teletype, direct program control of IO may lead to simpler programming. Programmed controlled IO transfers are effected by the execution of IOR, IORC, IOW or IOWC instructions (see Section 2.6.7).

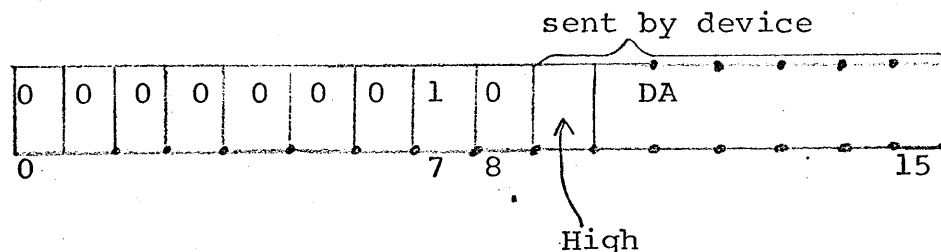
Multiplexor channels are provided in the basic Model II processors. When a device requires channel servicing, the device is serviced, and the program is simply subjected to a short delay. The multiplexor channel is capable of sustaining concurrent IO operations with several devices. Bytes of data are interleaved together and routed to or from the selected IO devices and to or from the desired locations in main storage. The channel's single data path is time-shared by the concurrently operating devices.

Selector channels are capable of operating only one device at a time; however, they permit data rates over a million bytes per second. Devices such as disc files and magnetic tapes operate only with selector channels, other devices operate in all data transfer modes. As with the multiplexor channel, the selector channel is invisible to the program; all instructions are directed at the device rather than the channel.

3.3 Operation of the Multiplexor Channel and Interrupt

Most devices require attention at the completion of every task. When operating in the multiplexor channel mode, this attention is automatically provided by the hardware without the need for program intervention until the transfer of the block of data is completed. Data transfers under program control require program intervention for every byte (or burst of bytes).

A device signals that it requires attention by requesting service at the priority level that has been assigned. (See Section 3.4 for use of REQ, ENABLE, and HIGH status bits.) When the priority of the active process drops below the priority of the request, the interruption occurs. The state of the old process is stored as part of its general register set (R_0 and R_1 contain the PSW) and a new general register set is switched in. Processor hardware then requests the device to transmit its address and its HIGH bit; this 7-bit number is ORed into bit positions 9-15 of a word with bit position 7 set to 1 and all other positions 0. This address, called the interrupt address, lies somewhere in field 1:



or $400 + DA$ if High = 0
 $500 + DA$ if High = 1

If the device is requesting service on priority level 8 (exclusively multiplexor channel transfers), the address selected by the processor will be:

$500 + DA$

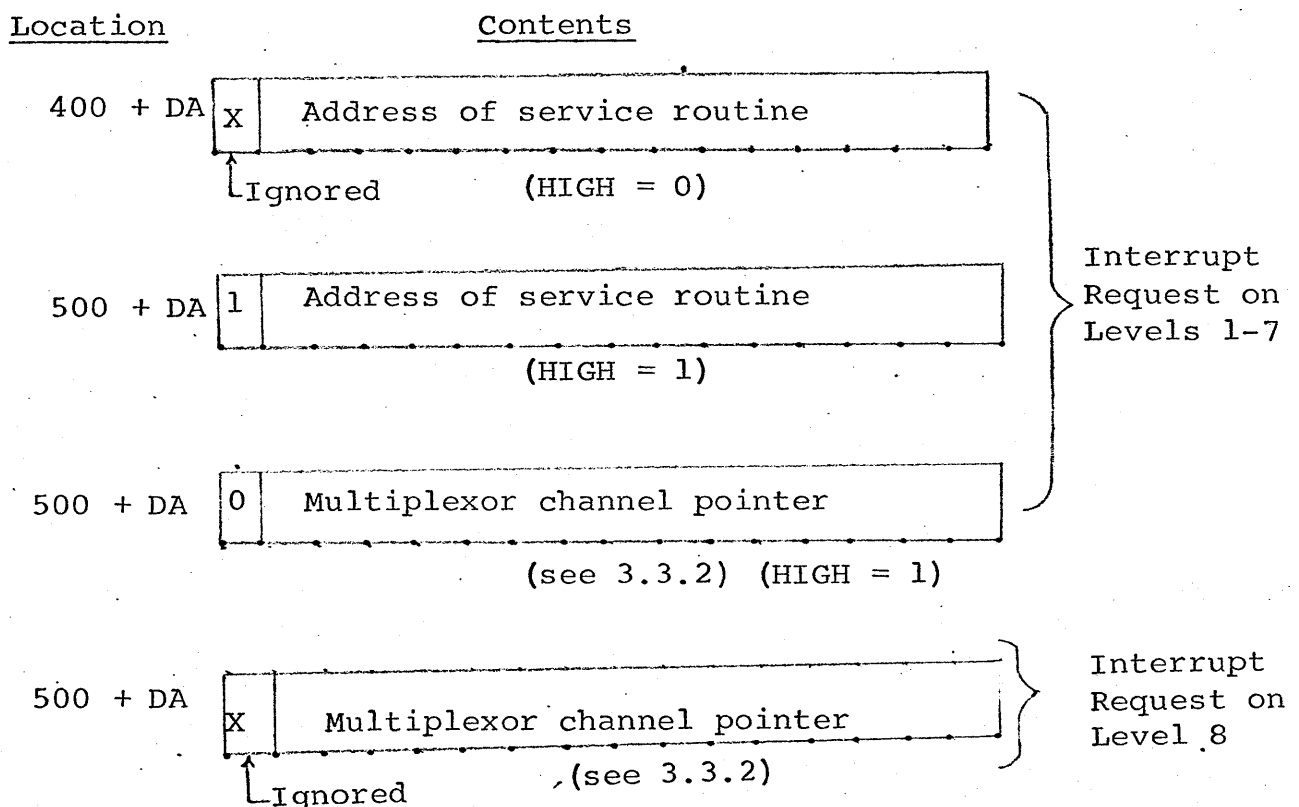
3.3 (Continued)

Subsequent operations will be one of two possible types:

- a. An IO service routine will be entered for program controlled data transfer or for the handling of end or unusual conditions.
- b. A multiplexor channel operation will be initiated and a data transfer will occur between main memory and the device data buffer. The interrupt will be automatically dismissed.

3.3.1 Program Controlled Interrupt Service

After the processor has received seven bits from the device (HIGH, DA), it forms a memory address as indicated above and then reads the addressed memory location. If HIGH was a zero or if bit 0 of the word read from memory was a one, bits 1-15 of the word read from memory are loaded into the program counter and control proceeds to a service routine. Thus, the service routine may or may not be entered from the HIGH interrupt location ($500 + DA$) depending upon the state of bit 0 of the word found at that location; however, the service routine is always entered if $HIGH = 0$. This is illustrated below:

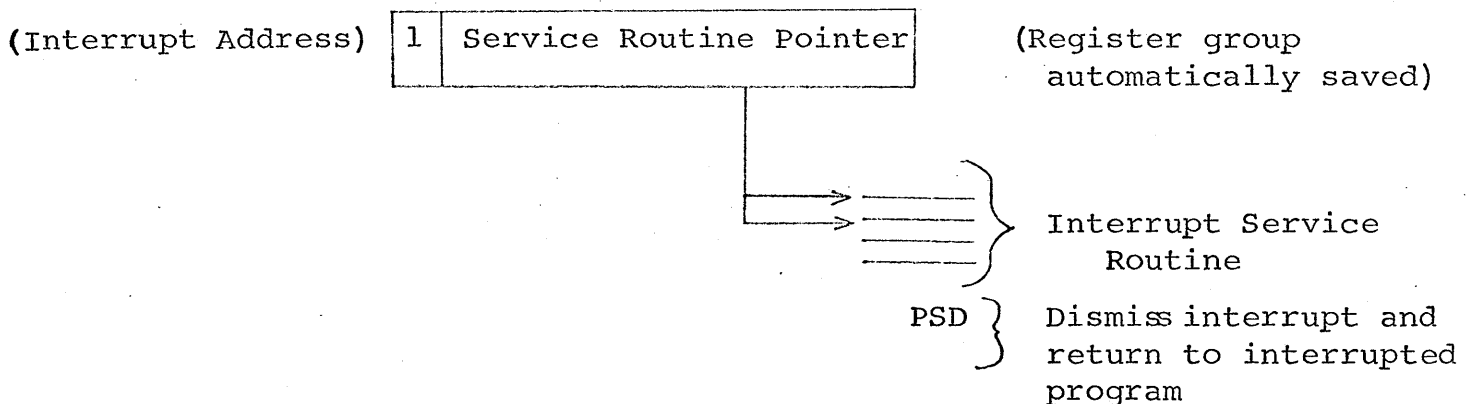


3.3.1 (Continued)

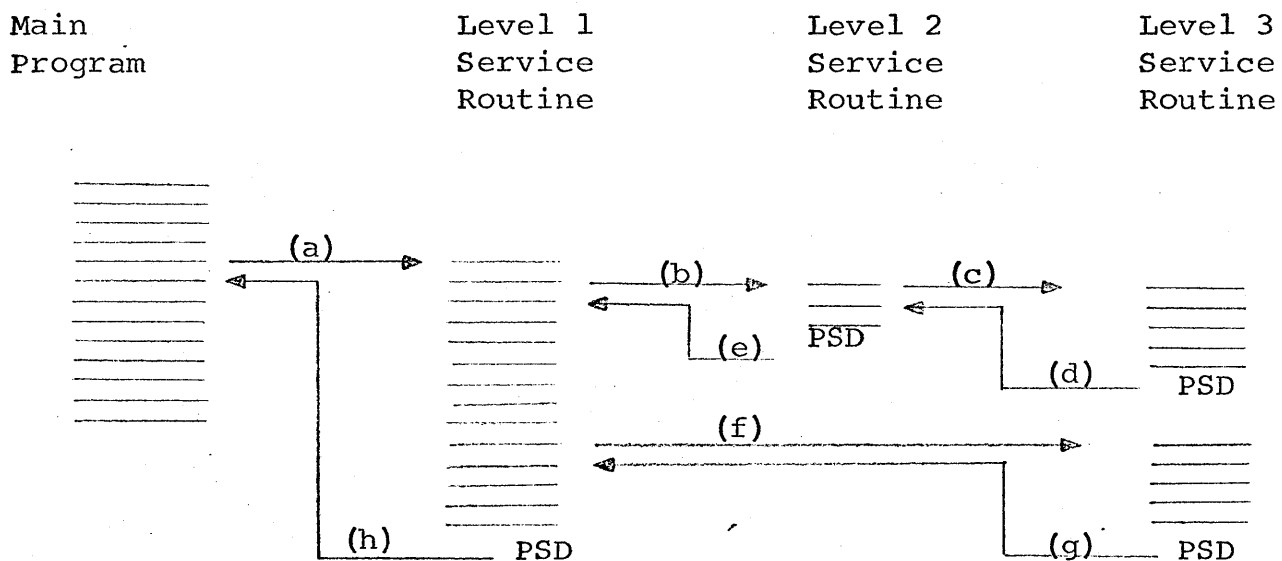
Note that a service routine may not be entered as a result of an interrupt request on Level 8. This level is reserved only for multiplexor channel operations.

When the interrupt service routine is entered, a bit in the priority level active indicators (see Section 2.7) is set corresponding to the priority level of the interrupt request. The interrupt service routine may be interrupted by a higher priority device but it may not be interrupted by a device of equal or lower priority.

Control is returned to the interrupted program by execution of the PSD instruction which clears the highest active level indicator and loads the RG register with the value of the new highest-active level indicator:



The interrupt system is fully nestable as illustrated below:

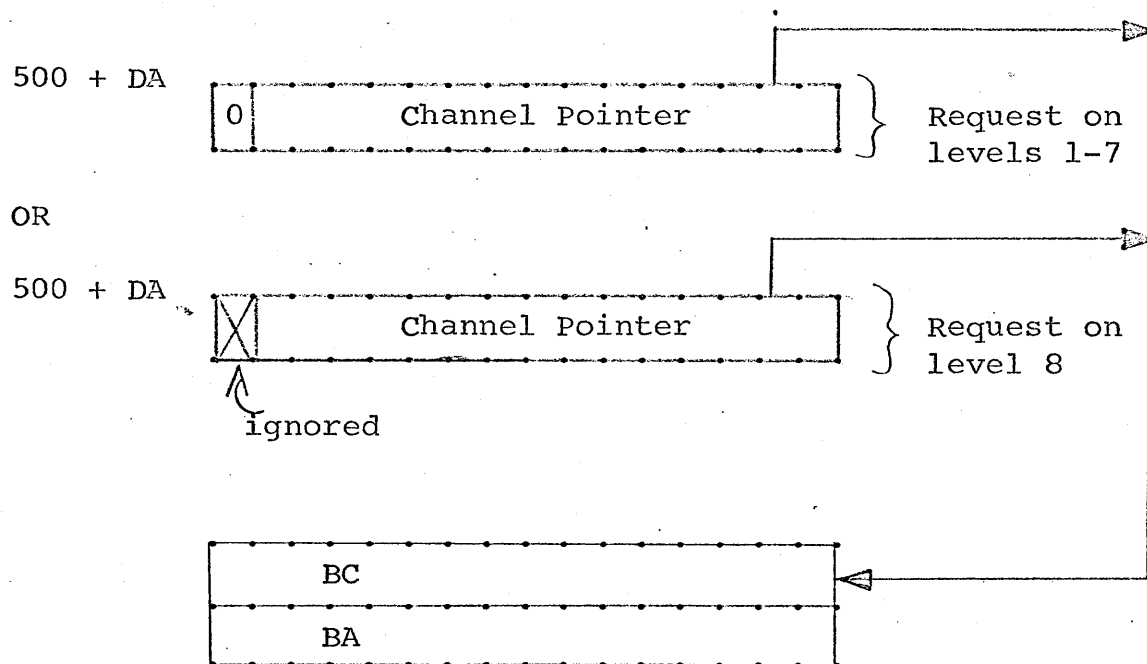


3.3.1 (Continued)

- (a) A device on Level 1 interrupts the main program; the RG is changed and the Level 1 service routine entered.
- (b) A Level 2 device interrupts the Level 1 program.
- (c) A Level 3 device interrupts the Level 2 program.
- (d) A Level 3 program completes and control returns to the interrupted Level 2 routine.
- (e) The Level 2 program completes and control returns to the interrupted Level 1 routine.
- (f) A Level 3 device interrupts the Level 1 program.
- (g) The Level 3 program completes and control returns to the interrupted Level 1 routine.
- (h) The Level 1 program completes and control returns to the interrupted main program.

3.3.2 Multiplexor Channel

The seven bit number returned by the device during the interrupt request sequence (HIGH, DA) is used to form a memory address as indicated in section 3.3. The processor then reads the word located at that address. If the request was on priority level 8 or if HIGH was a one and bit zero of the word read from memory is a zero, a multiplexor channel operation is indicated (see section 3.3.1). Bits 1-15 of the word read from memory are used to locate the first word of a channel double-word. This double-word is indicated below:



BC stands for byte counter and maintains a count of data bytes as they are transferred to and from the device. After each transfer, BC is incremented to determine whether or not this is the last byte. When initializing a device for multiplexor channel operation, the programmer must load BC with the two's complement of the number of bytes to be transferred. At the end of channel operation, the entire word will be set to zero. Exceptional conditions which cause termination before the specified number of bytes is read (written) leave the word nonzero.

3.3.2 (Cont.)

BA stands for byte address and maintains the address of the data byte next to be transferred to and from memory. After each transfer, BA is incremented. This byte address is shifted right before use to form a word address, the end bit determines which half of the word the data byte will be loaded into. A 1 indicates the left byte, a zero the right byte. (See section 2.6.4.) When the program initializes the channel, it must load BA with the byte address of the first byte to be transferred.

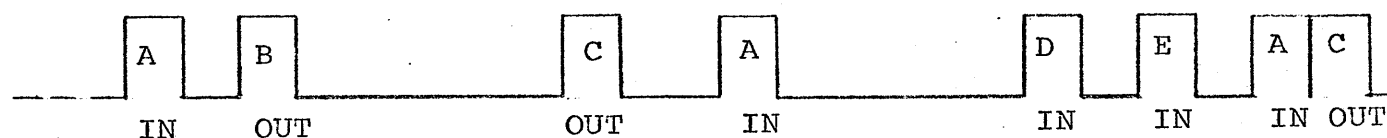
A byte of data is read from (sent to) the device and packed into (unpacked from) memory. The byte counter and address pointer are updated after each byte transfer. If the byte counter went to zero indicating that the last byte has been transferred, the multiplexor channel informs the device which clears its HIGH bit and does not clear its REQ bit. The device will then interrupt (possibly at a lower priority level) and a device interrupt service routine will be entered.

If the byte transferred was not the last byte, the device will normally clear its REQ bit and set its BUSY bit. (See section 3.4.1.)

If a device detects an unusual condition, it will clear HIGH and set UNUSUAL. If ENABLE is set, the device will interrupt and a device service routine will be entered.

The multiplexor channel (as well as IO operations initiated by the execution of an IO instruction) is capable of operating in one of two modes; the multiplexed mode and the burst mode.

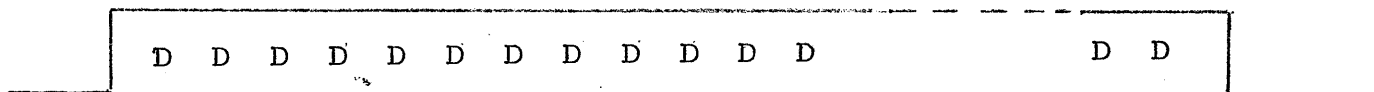
Capability to sustain several IO operations on a time-shared basis is the most important feature of the multiplexor channel. The multiplexed operation is illustrated below:



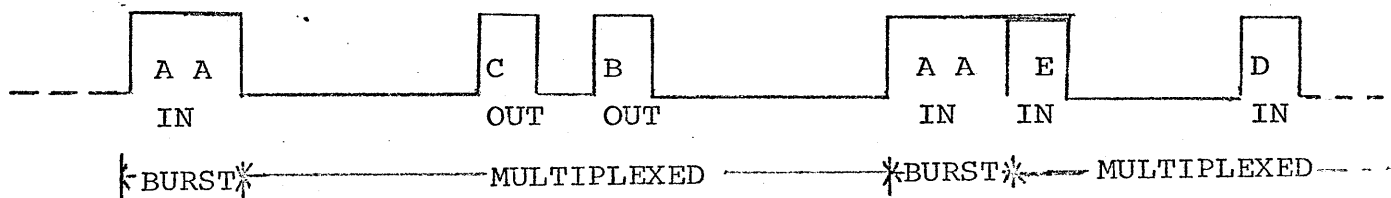
3.3.2 (Cont.)

In this mode, the channel services the IO devices asynchronously as the input data becomes available or when the output devices can accept more data (REQ is raised). Each data transfer consists of one byte of data (8 bits).

Burst mode is an alternate multiplexor channel operation. In this mode, shown below, the channel remains connected to a particular device until the transfer of a block of data is completed. All other devices are normally locked out of the channel until the end of the burst transmissions. The lengthy burst mode shown is usually reserved for high-speed, long-record transfers. As in all standard data transfers, the transmission is a byte at a time.



Most frequently, the channel will be alternating between multiplexed and burst modes. The burst mode is controlled by the device requesting service and can be as short as two bytes long. For instance, consider an input device, A, which provides 16 bits of information at a time. It places the channel in a burst mode long enough to transmit 2 bytes, then disconnects from the channel. The channel is then returned to normal multiplex operations. Use of the short burst mode increases the channel time efficiency since only one address setup cycle is needed, where two such cycles would be needed if the bytes from device A were transmitted in the multiplex mode.



After the data transfer, either single byte or burst, control is returned to the interrupted program.

3.4 Basic Peripheral Structure

Status Register

All PDP-X peripherals are normally controlled by loading their status register using an IOC command; however, certain simple devices (paper tape or teletype) may respond directly to IOW and IOR; the status register may be sensed using the IOS or IOT commands. All PDP-X peripheral unit data transfers are accomplished using IOR and IOW instructions (or by means of the multiplexor channel) which may also modify the status register. The six low order bits comprise the basic section of the status register which is common to all peripherals.

UNUSUAL	DIR	REQ	BUSY	HIGH	ENABLE
10	11	12	13	14	15

ENABLE - The ENABLE bit connects the peripheral to the interrupt system; with ENABLE set, either REQ or UNUSUAL cause an interrupt. With this bit cleared, the peripheral may in no way affect the operation of the rest of the system. ENABLE is cleared through the command line RESET on the IO Bus during the power up/down sequence, by instruction, and from the console.

HIGH - The HIGH bit indicates which of two possible interrupt addresses will be used by the processor when the peripheral interrupts. In addition, HIGH normally specifies which of two priority levels will be used to request the interrupt, HIGH = 1 indicating a higher priority.

HIGH is cleared whenever UNUSUAL is set. In normal use, HIGH = 1 indicates normal data transfer on the multiplexor channel, a byte is transferred each time REQ rises. When the block of bytes is completely transferred (overflow), HIGH is cleared and REQ is set, causing an interrupt at the lower address (400+DA).

3.4 (Cont.)

- BUSY - The BUSY bit indicates that the device is performing the requested function. It may be explicitly set by an IOC instruction or implicitly by an IOW instruction. BUSY is cleared when the device has completed its operation, REQ is set by the device when BUSY is cleared.
- REQ - The REQ bit is set whenever the device requires attention under normal conditions. If ENABLE is also set, an interrupt will occur. REQ may be cleared by an explicit IOC instruction and is implicitly cleared during byte transfers on the multiplexor channel.
- DIR - The DIR bit indicates the transfer direction of the device. Since it is a function either of the device type or device function, it may not be explicitly changed, although it may be sensed. DIR is sensed by the multiplexor channel to determine the direction of data transfer.
- 1 = out of processor into device
- 0 = out of device into processor
- UNUSUAL - The UNUSUAL bit indicates either error status or that some non-normal event has occurred. Some unusual conditions may only be cleared by operator intervention. Others may be cleared by clearing an error bit elsewhere in the status register. When UNUSUAL is raised, HIGH is cleared. If ENABLE is also set, an interrupt will occur.

(See also section 4.3.)

3.5 Paper Tape Peripherals

(Note that all are independent and may be simultaneously operating at their maximum rates.)

3.5.1 Paper Tape Reader/Punch

The paper Tape Reader/Punch (PTR/PTP) is built around the PC01 Reader/ Punch mechanism. It is intended for use where high-speed paper tape operations are required. Fan fold tape is normally used.

The tape reader operates at a maximum rate of 300 bytes (tape lines) per second; each byte may be program interpreted as either binary or alphanumeric (ASCII) data. A byte is read every 3.3 milliseconds, the reader stops if not reselected (set BUSY bit to 1 within 1.6 milliseconds after REQ comes up) between characters. Maximum reading rate is only obtained after reading approximately 10 characters consecutively.

The tape punch operates at a maximum rate of 50 bytes (tape lines) per second. Power to the punch motor is program controlled, "warm up" before punching the first character is 1 second; power remains up for 5 seconds after the last character is punched. Operation of the punch is synchronized with the motor, thus, the full rate may be obtained only if BUSY is set within 5 milliseconds of the previous REQ rise.

The status register of both the reader and the punch follow the general format. The UNUSUAL bit is raised to indicate that the device is out of tape; all 1's will be read by the reader, the punch may still punch approximately 10 lines. The out-of-tape condition, if ignored, does not stop the punch.

3.5.2 Keyboard/Printer

The keyboard/printer (TTI, TTO) is built around Teletype Corporation's Models 33, 35, and 37 ASR or KSR mechanism. The connection between the mechanism and logic is a simple four-wire cable so that the mechanism is easily remoted from the

processor. The ASR models have an attached paper tape reader/punch. Both the keyboard and printer operate at a maximum rate of 10 bytes per second; since operation is full duplex (completely independent), the program must echo any characters from the keyboard it wants printed. The status byte for both follow the standard format; there is no UNUSUAL bit.

BUSY is set in the keyboard when a key is struck; it is lowered and REQ set when the byte is available for input. If busy is set by the program, the (optional) reader mechanism is started (if enabled by the switch on the mechanism). The (optional) punch mechanism punches whatever is printed (when enabled by the switch on the mechanism).

3.5.3 Priority Assignments

The HIGH bit in the device status registers normally accomplishes two purposes: (a) it allows the processor to distinguish between two interrupt locations and, (b) it allows the device to select one of two priorities on which it may request service. Since the paper tape peripherals are relatively slow and have no data over-run problem, the latter is unnecessary. Thus, the paper tape peripherals request service on only one priority level regardless of the setting of the HIGH bit.

3.5.4 Paper Tape Peripherals Status Bytes

Paper Tape Reader	PTR	X	X	X	UNUSUAL (Out of Tape)	DIR =0	REQ	BUSY	HIGH	ENABLE
Paper Tape Punch	PTP	X	X	X	UNUSUAL (Out of Tape)	DIR =1	REQ	BUSY	HIGH	ENABLE
Keyboard	TTI	X	X	X	X	DIR =0	REQ	BUSY	HIGH	ENABLE
Printer	TTO	X	X	X	X	DIR =1	REQ	BUSY	HIGH	ENABLE

X indicates permanently 0

3.6 Device Assignment Table

Number (DA)	Type	Interrupt Location	Multiplexor Channel Pointer Location
00	power fail/parity	400	---
01-07	initiated priority interrupts	401-407	---
10	TTO	410	510
11	TTI	411	511
12	PTP	412	512
13	PTR	413	513
14	Small Display with Light Pen	414	514
15	Card Readers	415	515
16	Incremental Plotter	416	516
17	Relay Buffer	417	517
20-27	A/D/A; multiplexor, etc.	420-427	520-527
30-37	four extra full duplex TTY's	430-437	530-537
40	Magnetic Tape	440	540*

3.6 (Cont.)

Number (DA)	Type	Interrupt Location	Multiplexor Channel Pointer Location
41	Dectape	441	541*
42	Disk	442	542*
43	Line Printer	443	543*

Note: To allow for full expansion up to the maximum number of devices (64), memory locations 400_8 - 477_8 should be reserved in field 1 (locations 500_8 - 577_8 for Model II with Multiplexor Channel).

*Selector Channel Pointer Location

3.7 IO Bus

3.7.1 General characteristics:

1. Electrical - A twisted pair signaling system using push and pull current.
2. Interlock - DC interlocked control signals are used to insure reliable operation over extremely long distances and permit arbitrarily fast devices physically close to the processor.
3. Bi-directional - The signaling systems permit bi-directional signal flow; this is utilized on the data lines.
4. General line format -
 - 8 Bi-directional address/data lines
 - 8 Outbound control lines
 - 8 Inbound control lines
 - 8 Inbound priority interrupt request lines
 - 8 Outbound priority interrupt grant lines
5. Power control and ground return

3.7.2 Operation

The connection between the processor and the IO device control units is called the IO Bus. The interface consists of signal lines that connect the control units to the processor; except for the signal used to establish priority, all communication lines to and from the processor are common to all control units. At any one instant, however, only one control unit may be logically connected to the processor. The logical connection is maintained from the time it is first established by the processor until it is broken by the processor. The rise and fall of all signals transmitted over the interface are controlled by interlocked

3.7.2 (Cont.)

responses. This interlocking removes the dependence of the interface on circuit speed and bus length, making it applicable to a wide variety of circuits, data rates, and system configurations.

Forty signals comprise the bus including 24 control signals, data signals, power, ground, and spares. The out-bound priority interrupt grant signals are retransmitted by each device, all others are common to every device. The data lines are time-shared between data and address information. At the beginning of a processor initiated IO operation, these lines are used by the processor to transmit a device address, later in the operation, they are used to transmit the data bytes. The data/address bus is bi-directional.

The priority lines are used to synchronize requests from devices at the various priority levels. A device requesting attention at a particular priority level does so by transmitting its request on the appropriate line of the eight available. The processor IO section grants such a request by transmitting on the associated out-bound priority interrupt grant line from the processor. Devices not requesting retransmit the grant signal; the first one requesting (nearest to the processor on that level) blocks it and transmits its address and the HIGH bit to the processor on the data lines and a signal on the Address In line (Inbound Control).

The control signals are used to establish the mode of operation on the bus. One control line out (SYNC) is the synchronizing signal; its rise indicates that all other control and address information is correct and may be strobed by the devices. One control line in (RTN) is similarly used by the processor. The other seven lines are decoded to indicate data/command, in/out, multiple/single byte, etc. Upon (a) receipt of SYNC, (b) control lines decode to address, (c) and its address is on the data lines, then a device becomes selected and will respond to subsequent commands. The device indicates that it has become selected by transmitting the RTN signal.

3.7.2 (Cont.)

With the rise of RTN which indicates that a device has become selected and is ready to accept a command, the processor drops SYNC, resets the control lines to the appropriate command code, and prepares to raise SYNC again as soon as RTN drops. The subsequent RTN rise, SYNC fall, RTN fall triplet completes the byte transfer. A word transfer requires raising SYNC followed by another such triplet.

If during a selection sequence a device fails to respond (e.g., addressing a non-existent or disconnected device) automatic time-out circuitry causes the processor to continue, indicating this failure to the program by setting bit 6 of the Program Status double-Word. This will cause a trap if the Error Trap Enable bit (1) of the PSW is set.

See PDP-X Technical Memorandum #26 for a detailed description of sequences on the IO Bus.

3.7.3 Line Definitions

Line (direction)	X	State	Function
Command X Out			Processor device control information
	0		SYNC - assertion indicates validity of other command lines.
	1		Parity out (optionally used)
	2,3		Reset conditions
		0	normal operating state
		1	processor has stopped
		2	load (reset to state for automatic read-in of bootstrap program)
		3	reset, clear all device status registers
	4		spare
	5,6,7		command mode, information content of data lines
		0	DATO - data to device
		1	DATI - data to processor, channel operating
		2	ADDRESS - address to device
		3	DATI END - data to processor, no channel

Line (direction)	X	State	Function
		4	DATO LAST - data to device, channel overflow
		5	DATI LAST - data to processor, channel overflow
		6	CONO - command to device status
		7	CONI - device status to processor
Response X In			Device→processor response to command
	0		RTN - assertion in response to SYNC to interlock bus, assertion indicates validity of other response lines
	1		Parity In (optionally used)
	2		Address In (Response to Priority Out)
	3		More bytes (data/status) required
	4,5		special interrupt controls for channel
		0	normal
		1	inhibit BA count in channel
		2	force last cycle of channel operation after transmitting 16-bit address
		3	permit only BC cycle of channel operation

Line (direction	X	State	Function
	6,7		Direction control
	.	0	OUT - processor to device
		1	
		2	IN - device to processor
		3	ADD IN - add device data to processor
Request X In	1-8		Device→processor interrupt request lines; eight is highest priority*
Data X In/Out	0-7		Device↔processor data/address lines
Priority X Out	1-8		Processor→device interrupt request grant lines; instructs device to transmit its address; these lines are retransmitted if used; eight is highest priority*, one lowest
Gnd	1		Heavy ground wire interconnection
Pwr	1		Power supply for remote turn on/off and priority line receiver/transmitter

*Priority level eight cannot support a program process (i.e., it has no associated register group), but it can only handle multiplexor requests.

3.8 IO Configurations

PDP-X Model II has two priority levels (register group 0, 1) and the special multiplexor channel (priority level 8). A typical system might be configured as follows:

Device	Priority		
	0*	1	8
Teletype In/Out		L,H	1
Paper Reader/Punch		L,H	
Display and Light Pen		L	H
A/D Converters		L	H

*Main Program Level

3.8 (Cont.)

A larger PDP-X System with 4 levels of priority interrupt (0-3) in addition to the special multiplexor channel at level 8 might be configured as follows:

Device	Priority				
	0*	1	2	3	8
Teletype In/Out		L,H			
Paper Tape Reader/Punch			L,H		
Display and Light Pen		L		H	
Card Reader			L	H	
Incremental Plotter			L,H		
A/D Converters				L	H
Real Time Clock**		L			
Interprocessor Buffer		L		H	
Dectape Selector Channel***		L			
Magtape Selector Channel***		L			
Line Printer Selector Channel***		L			
Power fail/parity detection**				L	

*Main Program Level

**Has no Channel Provision (HIGH is permanently zero)

***Data Transfer occurs through selector channel, HIGH is permanently zero

H indicates HIGH = 1

L indicates HIGH = 0

3.8 (Cont.)

For configurations with special IO devices or for systems with special IO requirements, the priority interrupt system may be expanded to 8 levels (0-7) in addition to the special multiplexor channel at level 8.

4.0 Appendices

4.1 Assembly Language Conventions

To facilitate the interchange of code for the PDP-X, some assembly language conventions will be briefly stated below. It is to be noted that these are in no way complete nor are they necessarily indicative of the final form of the assembly language for the PDP-X.

The general syntax is derived from MACRO-10. The generalized instruction statement will have the form:

TAG: OPCODE R, @ ADDRESS (INDEX) ; COMMENT

This may be summarized by

1. Symbols may be six characters long, must begin with a letter and may contain any of the characters A-Z, 0-9, %, . .
2. : Delimits the tag field. The symbol to the left of the colon is assigned a value equal to the current location counter.
3. , Separates the R-field and the address field.
4. @ Indicates indirect addressing and, in an instruction, forces long form. In an address constant, it sets bit 0 to a 1. For example, @A would generate a 16-bit constant with bits 1-15 containing the value of A and with bit 0 a 1.
5. () Surround the index field. If no index field is specified, the assembler will pick the appropriate addressing mode. For example, in location 1000 (octal), the instruction LDA 2, 100 would generate an index field of 0 and D₁ would contain 100.

LDA 2, 1010 would generate an index field of 1 and D₁ would contain 010.

LDA 2, 100 (3) would generate a short form indexed instruction. D₁ would contain 100.

LDA 2, 1010 (3) would generate a long form indexed instruction. D_1 would contain 200, D_2 would contain 001010.

LDA 2, 200 would force long form. D_1 would contain 200, D_2 would contain 000200.

6. ; Delimit comments. Comments extend to the next CR-LF pair.
7. • Has the value of the current location counter.
8. = Indicates assignment. The symbol to the left of the = is assigned the value of the expression to the right of the =.
9. In I/O instructions, the device number replaces the R-field. For example, if TTY has the value of the teletype device number, IOW TTY, MEM will generate a doubleword with the device number in D_1 , $X = 0$, $D_2 = \text{MEM}$.
10. In instructions where the R-field is interpreted as part of the operation code, the comma may be omitted. For example, BZ , Y and BZ Y are both legal and equivalent, whereas BZ 3, Y is illegal. The R-field is ignored and the error flagged.
11. [] Are used to indicate literals.
12. ↑ used to indicate a local radix change (see MACRO-10 User's Manual). Valid forms are:
 - ↑B binary (0-1)
 - ↑O octal (0-7)
 - ↑D decimal (0-9)
 - ↑X hexadecimal (0-9, A-F)
13. PSEUDO instructions
 - a. RADIX EXPRESSION - Global radix is changed.
 - b. BOUND EXPRESSION - Where expression has a value equal to some power of

two. This will move the location counter to the appropriate word boundary. For example, BOUND 4 will move the location counter to the next multiple of 4.

- c. LOC EXPRESSION - Will set the location counter.
- d. PHASE EXPRESSION
DEPHASE - See MACRO-10 User's Manual.
- e. BLOCK EXPRESSION - Generates a block of zeros.
- f. LPOOL EXPRESSION - Generates a literal pool of size equal to the value of the expression. (See SDS Sigma 2 Symbol Reference Manual.)

14. Expressions - Any atom (symbol or number) can be combined with the operator's

- + Add
- Subtract
- / Divide
- * Multiply
- & Logical and
- ! Logical or
- () for evaluation procedure

(See MACRO-10 User's Manual for effect upon relocation.)

15. In general, the assembler will optimize short program segments and produce relocatable binary output. The size of the largest optimizable segment will be determined by the available memory size. Programs will consist of several of these segments loaded by a linking-loader. Core images can then be saved.

4.2 Instructions (alphabetic)

mnem	Definition	OP	D ₁ (Octal)	R	CC	Models
ADD	<u>A</u> DD	3	--	--	0,1,2	I,II
AND	<u>A</u> ND	2	--	--	1,2	I,II
B	<u>B</u> Branch	4	--	3	--	I,II
BAL	<u>B</u> Branch <u>A</u> nd <u>L</u> ink	4	--	7	--	I,II
BCN	<u>B</u> Branch if <u>C</u> arry <u>N</u> onzero	4	--	0	--	I,II
BCZ	<u>B</u> Branch if <u>C</u> arry <u>Z</u> ero	4	--	4	--	I,II
BM	<u>B</u> Branch if result <u>M</u> inus	4	--	1	--	I,II
BN	<u>B</u> Branch if result <u>N</u> onzero	4	--	2	--	I,II
BP	<u>B</u> Branch if result <u>P</u> ositive	4	--	5	--	I,II
BZ	<u>B</u> Branch if result <u>Z</u> ero	4	--	6	--	I,II
CLR	<u>C</u> Lea <u>R</u>	5	--	7	1,2	I,II
CMP	<u>C</u> o <u>M</u> Pare	6	111	--	1,2	II

mnem	Definition	OP	D ₁ (octal)	R	CC	Models
COM	<u>COM</u> plement	5	--	1	1,2	I, II
DIV	<u>DIV</u> ide	6	103	--	--	II
HLT	<u>HALT</u>	7	001	7	--	I, II
INC	<u>INC</u> rement	5	--	2	1,2	I, II
IOC	<u>IO</u> <u>Com</u> mand	7	--	5	--	I, II
IOR	<u>IO</u> <u>Rea</u> d	7	--	0	--	I, II
IORC	<u>IO</u> <u>Rea</u> d <u>Ch</u> aracter	7	--	1	--	I, II
IOS	<u>IO</u> <u>Sta</u> tus	7	--	4	--	I, II
IOT	<u>IO</u> <u>Test</u> status	7	--	6	1,2	I, II
IOW	<u>IO</u> <u>Write</u>	7	--	2	--	I, II
IOWC	<u>IO</u> <u>Write</u> <u>Ch</u> aracter	7	--	3	--	I, II
LCMP	<u>Log</u> ical <u>Co</u> <u>Mp</u> are	6	110	--	1,2	II
LDA	<u>Loa</u> d <u>Acc</u> umulator	0	--	--	--	I, II
LDC	<u>Loa</u> d <u>Ch</u> aracter	6	114	--	--	II
LDIV	<u>Log</u> ical <u>DIV</u> ide	6	102	--	--	II

mnem	Definition	OP	D ₁ (Octal)	R	CC	Models
LMH	<u>L</u> oad <u>M</u> onitor map <u>H</u> igh	7	011	7	--	II _c
LML	<u>L</u> oad <u>M</u> onitor map <u>L</u> ow	7	010	7	--	II _c
LMUL	<u>L</u> ogical <u>M</u> Ultiply	6	100	--	--	II
LUH	<u>L</u> oad <u>U</u> ser map <u>H</u> igh	7	013	7	--	II _c
LUL	<u>L</u> oad <u>U</u> ser map <u>L</u> ow	7	012	7	--	II _c
MUL	<u>M</u> Ultiply	6	101	--	--	II
NEG	<u>N</u> EGate	5	--	3	1,2	I,II
POB	<u>P</u> Op and <u>B</u> ranch	6	116	6	--	II
POC	<u>P</u> Op and <u>C</u> ount	6	116	4	--	II
POL	<u>P</u> Op, branch and <u>L</u> ink	6	116	7	--	II
POP	<u>P</u> OP	6	116	5	--	II
PSC	<u>P</u> riority <u>S</u> ystem <u>C</u> lear	7	006	7	--	I,II
PSD	<u>P</u> riority <u>S</u> ystem <u>D</u> ismis	7	007	7	--	I,II
PSI	<u>P</u> riority <u>S</u> ystem <u>I</u> nhibit	7	004	7	--	I,II
PSR	<u>P</u> riority <u>S</u> ystem <u>R</u> equest	7	005	7	--	II _b

mnem	Definition	OP	D ₁ (Octal)	R	CC	Models
PUSH	<u>P</u> USH	6	116	1	--	II
PUB	<u>P</u> ush and <u>B</u> ranch	6	116	2	--	II
PUC	<u>P</u> ush <u>C</u> ount	6	116	0	--	II
PUL	<u>P</u> ush, branch and <u>L</u> ink	6	116	3	--	II
RCS	<u>R</u> ead <u>C</u> onsole <u>S</u> witches	7	002	7	--	I, II
RIO	<u>R</u> eset <u>I</u> O system	7	000	7	--	I, II
RL	<u>R</u> otate <u>L</u> eft	5	--	5	0,1,2	I, II
RR	<u>R</u> otate <u>R</u> ight	5	--	4	0,1,2	I, II
SHFT	<u>S</u> HIFT	6	113	--	0,1,2	II
STA	<u>S</u> Tore <u>A</u> ccumulator	1	--	--	--	I, II
STC	<u>S</u> Tore <u>C</u> haracter	6	115	--	--	II
SUB	<u>S</u> UBtract	6	112	--	0,1,2	II
SWP	<u>S</u> WAP bytes	5	--	6	1,2	I, II
TST	<u>T</u> eST	5	--	0	1,2	I, II
TSTC	<u>T</u> eST and <u>C</u> omplement	6	107	--	1,2	II

mnem	Definition	OP	D ₁ (octal)	R	CC	Models
TSTN	<u>Te</u> <u>ST</u> but change <u>N</u> othing	6	104	--	1,2	II
TSTO	<u>Te</u> <u>ST</u> and set to <u>O</u> nes	6	106	--	1,2	II
TSTZ	<u>Te</u> <u>ST</u> and <u>Z</u> ero	6	105	--	1,2	II
WCI	<u>W</u> rite <u>C</u> onsole <u>I</u> ndicators	7	003	7	--	I,II

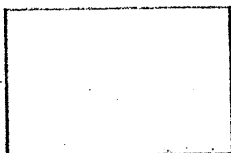
4.3 Peripheral Structure (Continued)

4.3.1 Flow Chart Conventions

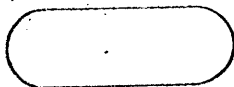
.....

Implicit transfer of control.
Since the device proceeds asynchronously with the processor.

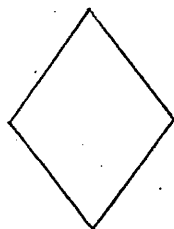
Explicit transfer of control.



Operation performed by the processor either under control of the program or under control of the multiplexor channel.

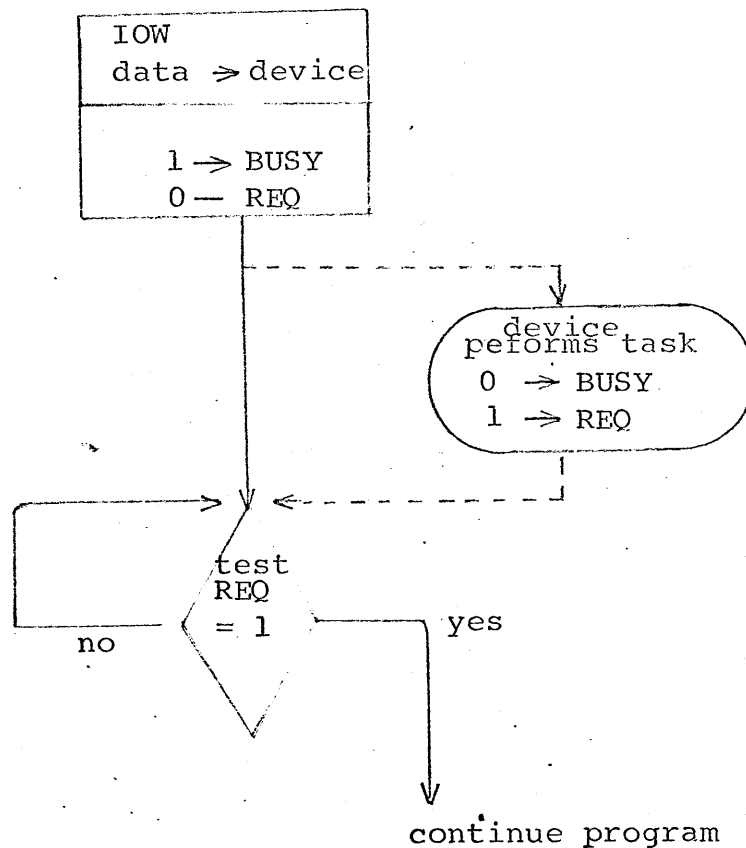


Operation performed by the device.



A test made either by the device (implicitly) or by the processor under program control.

4.3.2 Output Device - no interrupt, output one character

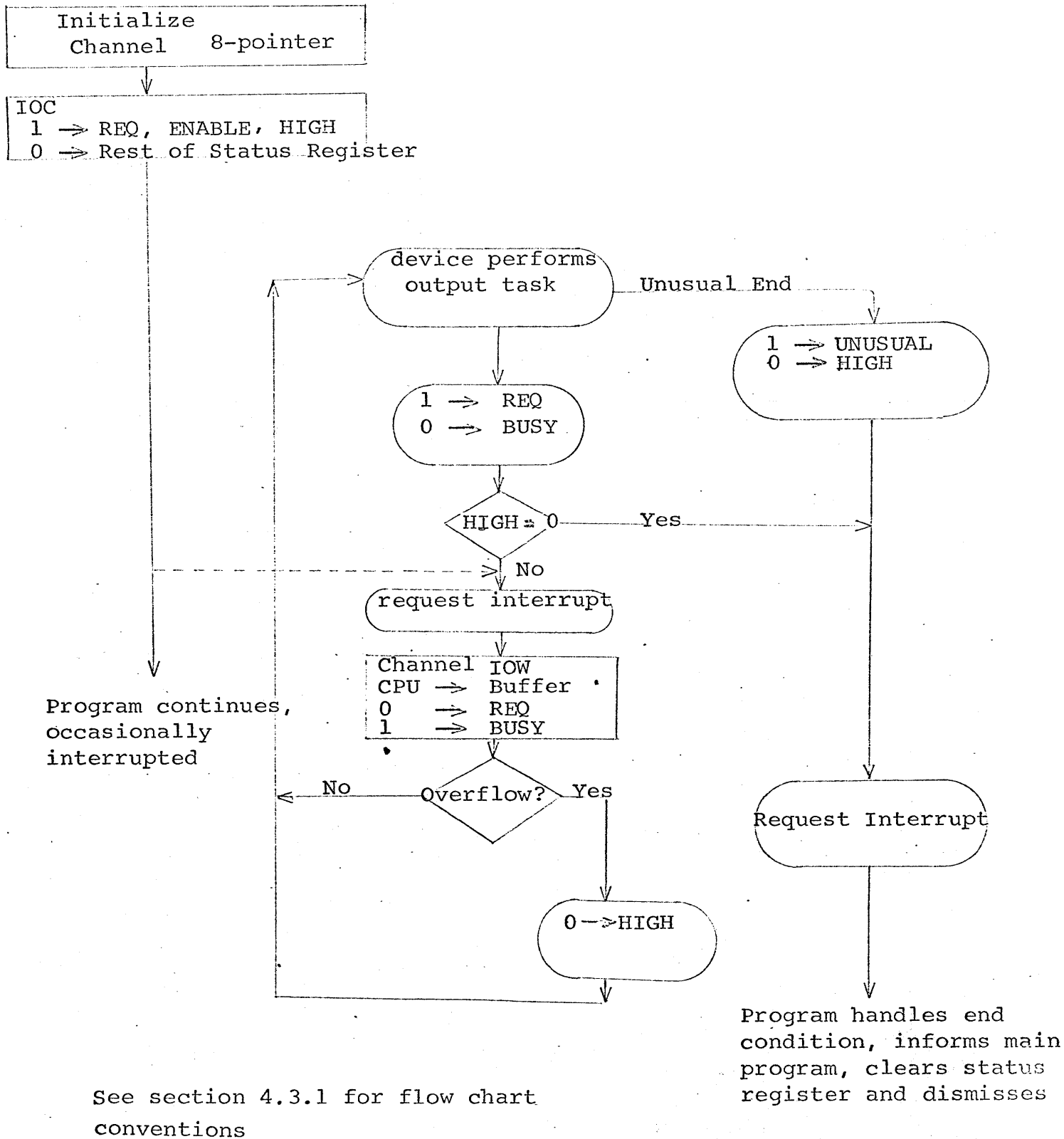


; example for paper tape punch

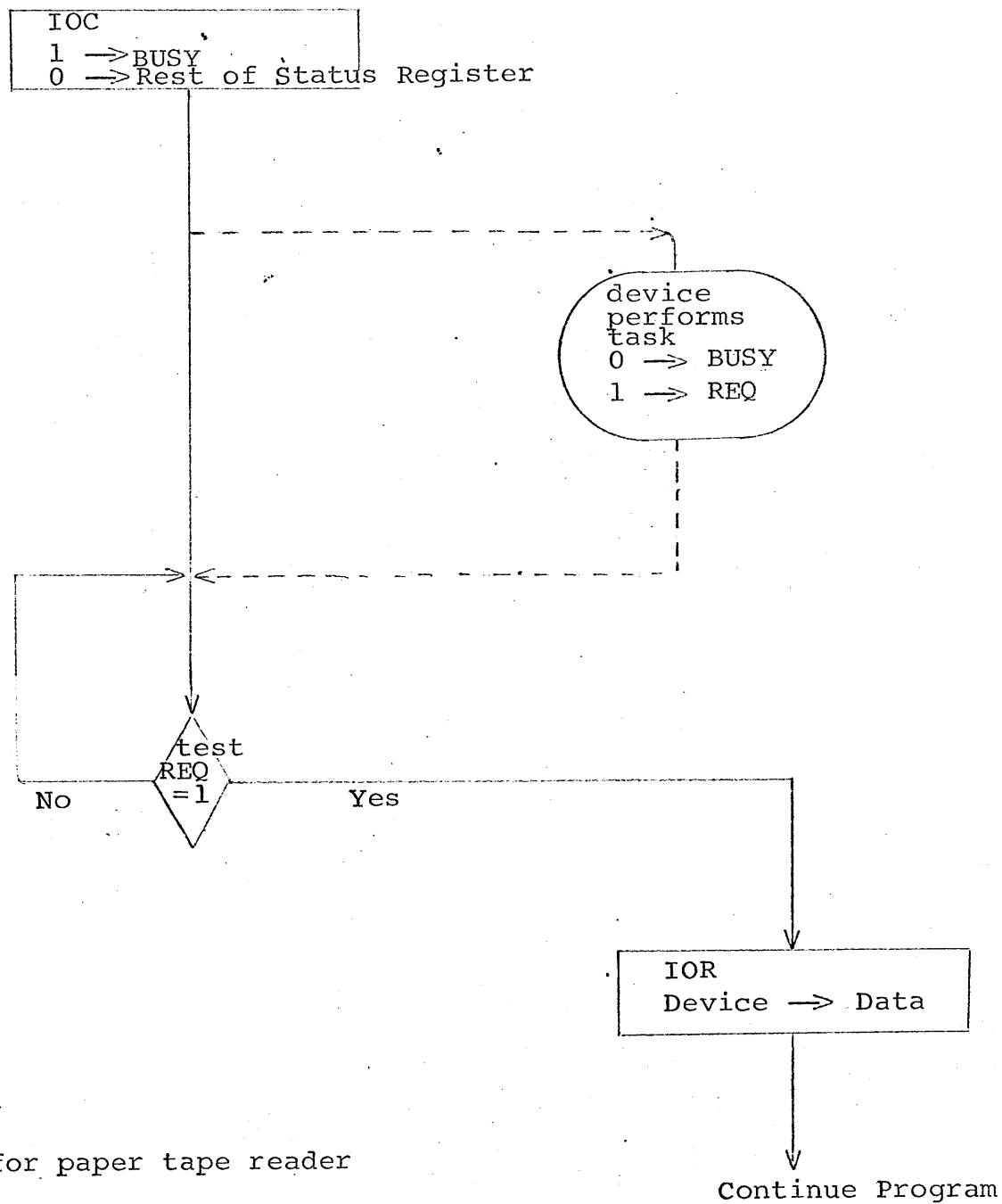
IOW PTP, data	; load byte in punch
IOT PTP, [10]	; set BUSY clear REQ
	; test REQ (done)
BZ .-1	; not set, go back to test

(See Section 4.3.1 for flow chart conventions.)

4.3.3 Output Device - interrupt mode (multiplexor channel)



4.3.4 Input Device - no interrupt, input one character



; example for paper tape reader

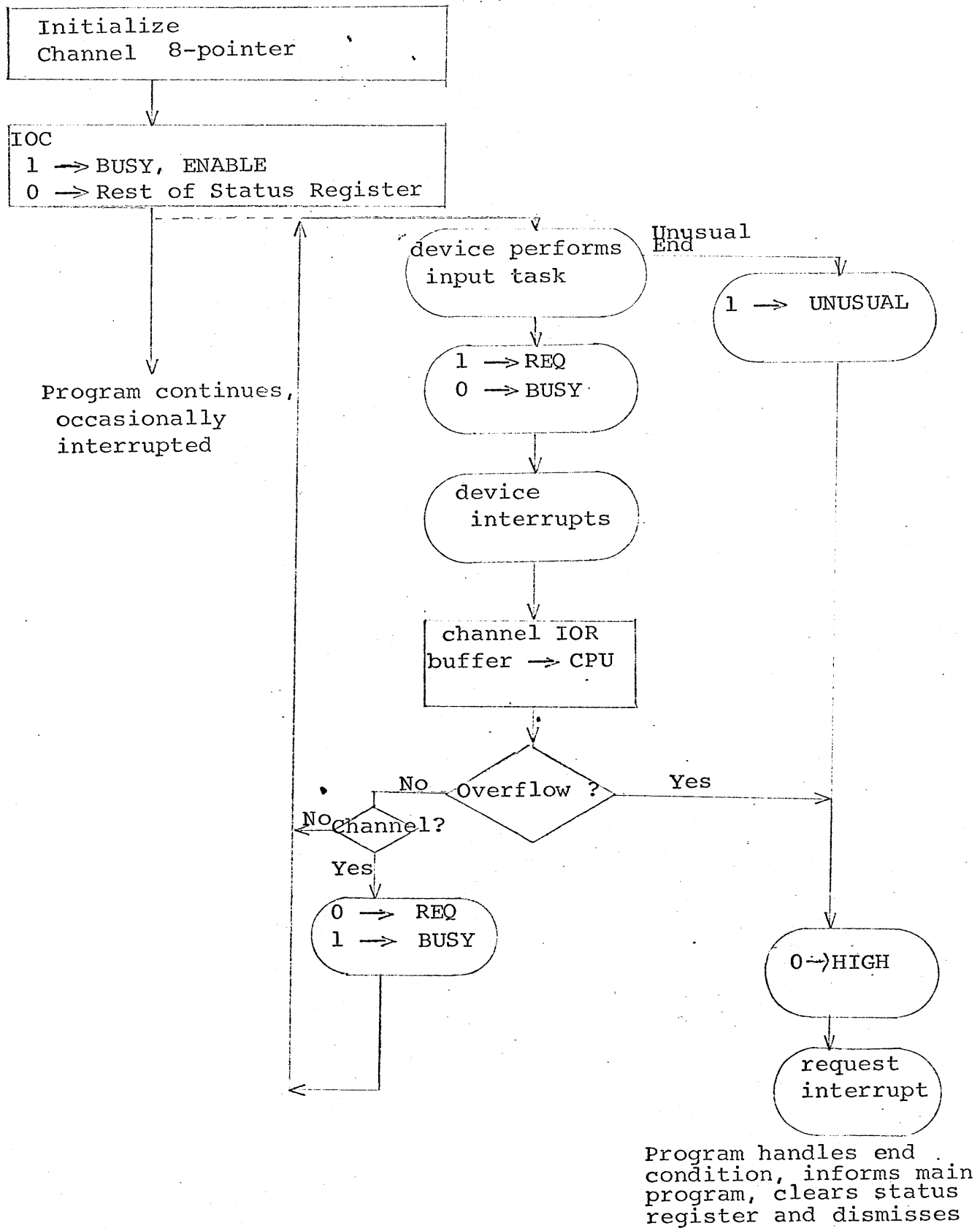
IOC PTR, [4] ; start reader

IOT PTP, [10] ; test REQ

BZ .-1 ; not set, go back to test

IOR PTR, data ; read in data byte

4.3.5 Input Device - interrupt mode (multiplexor channel)



4.3.6 Programming Example - Punch Routine

```

; General Subroutine to Punch a Block of Data
; Called by BAL PUNCH
;
;      Number of Data Bytes
;
;      Byte Pointer to Data
;
;      Return when Punch has Started
PUNCH:  TST PUNLOK          ; IS PUNCH ROUTINE BUSY?
        BZ  PUNCH          ; IF ZERO, IT IS STILL BUSY, WAIT
        STA 3, PUNLOK      ; IT IS DONE - SAVE AC3
        LDA 3, (2)         ; GET COUNTER
        NEG 3              ; FORM TWO's COMPLEMENT
                           ; AS REQUIRED BY CHANNEL
        STA 3, PUNBLK      ; STORE COUNTER FOR CHANNEL
        LDA 3, 1 (2)       ; GET POINTER
        STA 3, PUNBLK+1    ; STORE POINTER FOR CHANNEL
        LDA 3, PUNLOK      ; RESTORE AC3
        CLR  PUNLOK        ; SET LOCK TO ZERO
                           ; START UP PUNCH BY ENABLING
        IOC, PTP, [13]     ; TO INTERRUPT AND SETTING
                           ; REQ, HIGH, PUNCH WILL START
                           ; IMMEDIATELY
        B      2(2)        ; EXIT ROUTINE
PUNLOC:  -1                ; 0 MEANS BUSY, -1 MEANS

```

4.3.6 (Cont.)

```

; NOT BUSY

; Unusual or usual end interrupt handler

PUNDON:  IOT  PTP,  [40]      ; TEST UNUSUAL BIT
        BN   PUNERR          ; OUT OF TAPE IN PUNCH, GO TO ERROR
        COM  PUNLOK          ; MAKE LOCK NONZERO (NON-BUSY)
        IOC  PTP,  [0]      ; CLEAR OUT PUNCH
        PSD                      ; DEBREAK AND DISMISS

PUNBLK:  BLOCK 2              ; 2 WORDS FOR BC, BA

; INTERRUPT ADDRESS

LOC      400 + PUNDA

        @ PUNDON              ; ADDRESS OF SERVICE ROUTINE
                                ; WITH BIT 0 SET TO A 1

; MULTIPLEXOR CHANNEL POINTER

LOC      500 + PUNDA

        PUNBLK                ; POINTER TO BYTE COUNTER AND
                                ; BYTE ADDRESS SET UP BY PUNCH
                                ; ROUTINE FOR USE BY MULTIPLEXOR
                                ; CHANNEL

; DEFINE DA CODE FOR PUNCH

PUNDA = 12

```

4.4 Reserved Memory

Certain memory locations are reserved in PDP-X for general registers, IO service routine pointers and multiplexor/selector channel pointers. This is illustrated below:

