Processor
Organization

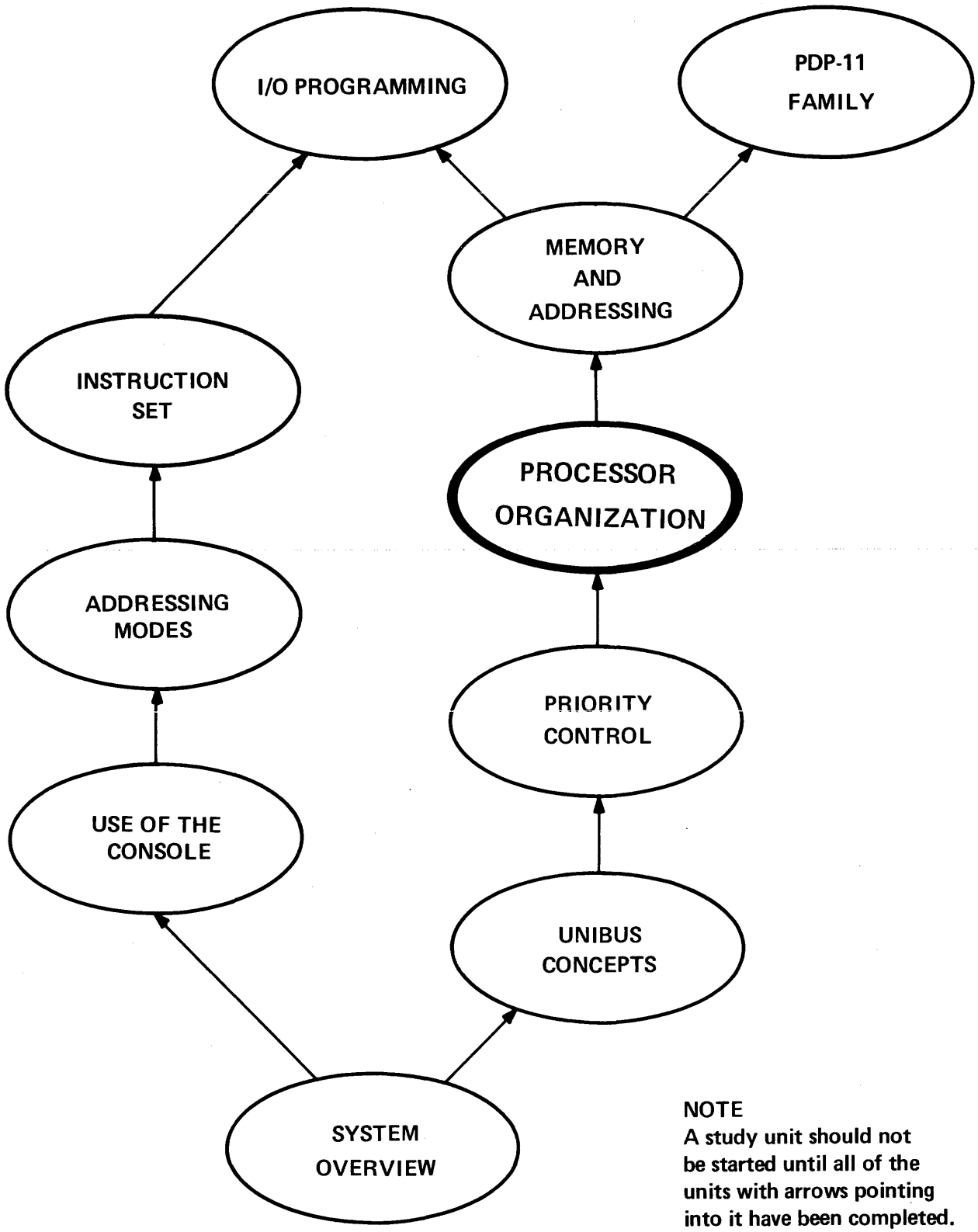
student workbook introduction to the pdp11

digital equipment corporation · maynard. massachusetts

1st Printing, March 1974
2nd Printing (Rev), November 1974
3rd Printing (Rev), April 1977

Printed in U.S.A.

course map



NOTE
A study unit should not be started until all of the units with arrows pointing into it have been completed.

read on

READ
LEARNING
OBJECTIVES
(page 1)

NOW RUN FILM
CARTRIDGES A & B

REVIEW
MATERIAL
(pages 3 – 16)

TAKE TEST &
CHECK RESULTS
(pages 17 – 24)

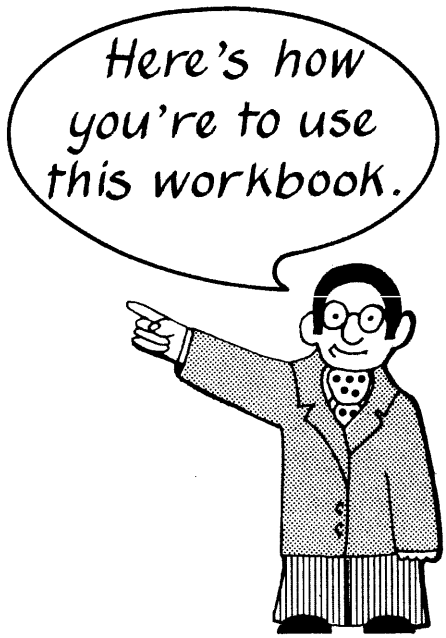
ANY
INCORRECT
ANSWERS

YES

PLEASE REVIEW THE
MATERIAL YOU'RE
HAVING DIFFICULTY WITH.
(ADDITIONAL RESOURCES ARE
LISTED ON PAGE 2.)

NO

GOOD WORK!
NOW GO ON TO THE
NEXT STUDY UNIT.



read on

objectives

After completing this study unit you should be able to . . . .

★ Describe the role of the processor in a PDP-11 system and describe the purpose of the three main elements in the processor:

- Unibus control
- General-purpose registers
- Data manipulation logic

★ Describe how information flows through the various processor elements.

★ Explain the steps the processor performs in order to execute an instruction and explain the various functions and bus cycles performed during each of these steps:

- Fetch instruction
- Obtain the operand(s)
- Execute the instruction
- Service pending interrupts or traps

★ Explain how the Unibus Control handles all address, control, and data flow between the processor and all other devices.

★ Describe in detail the role of the Unibus Control during data transactions, interrupt handling, and trap handling.

★ Describe the three prime functions of the data manipulation logic.

★ Explain the general functions of the eight general-purpose registers and describe the specific functions of these registers when used as:

- Program counter register (PC)
- Stack pointer register (SP)

★ Explain the use of the stack during interrupts; including the roles played by the program counter and stack pointer registers.

★ Briefly describe the purpose of the eight additional general-purpose registers in the large PDP-11 processors, such as the 11/45 and 11/70.

★ Explain the functions of the various bits that make up the processor status word.

read on ➧

additional resources

- **PDP-11/04/05/10/35/40/45 Processor Handbook**

  Read Chapter 2, Paragraph 2.2 (Central Processor), and Chapter 5, Paragraph 5.7 (Processor Traps).

The following material is covered in this study unit:

Topic	*Key Points*	*Visual Ref.*
basic concepts	★ The PDP-11 processor may be divided into three functional elements: • Unibus control – handles communication between processor and external devices. • General-purpose registers – provide internal storage of data. • Data manipulation logic – performs arithmetic and logic operations, decodes instructions, and controls information flow to and from the general-purpose registers.	10–13
information flow	★ Information flow through the three processor elements is in the form of a "figure eight".	14
data manipulation logic	★ The data manipulation logic may be divided into four functional blocks: • Bus address register (BAR) – provides temporary storage for bus addresses used by the processor. • Buffer register (BR) – serves as a holding area for data. • Instruction register (IR) – holds all instructions fetched from memory so they are available for decoding. • Arithmetic logic unit (ALU) – manipulates operands based on the instruction decoded in the IR.	15–18

read on ➧

Topic	*Key Points*	*Visual Ref.*
typical instruction cycle	★ In order to perform the job required by an instruction, the processor steps through a specific sequence of operations: • fetch – obtain instruction from the memory. • obtain data – get operand or operands. • execute – performs job specified by the instruction. ★ That completes execution of an instruction. However, at this point the processor may *service* any pending interrupts or traps because servicing occurs *between* instruction cycles.	19–24
fetch operation	★ This step is used to obtain a specific instruction from memory. • The CPU obtains the address of the instruction from the program counter (PC). • The Unibus control performs a DATI bus cycle to retrieve the instruction. • The data manipulation logic decodes the instruction to determine what task is to be performed. • The PC is incremented by two so that it points to the next instruction word in the program.	25–30

read on ➧

Topic	*Key Points*	*Visual Ref.*
obtain operand(s)	★ This step is used to obtain the operand, or operands, specified by the instruction. • If the operand is stored in a GPR, no further operation is required. • If the operand's *address* is stored in the GPR, then the processor must do a DATI bus cycle to obtain the operand. • If an address *pointer* is stored in the GPR, then the processor must perform two DATI bus cycles; one to get the address, the second to get the operand. • If the GPR contains a *base address,* then the processor must first calculate the *actual* address before performing the necessary bus cycles to obtain the operand.	31–37
execute operation	★ This step is used to perform the task specified by the decoded instruction. • The task is executed by the data manipulation logic in the processor. • The data manipulation logic may perform arithmetic, logical, or control operations. • The result of the operation is then stored either internally (GPR) or externally (some bus device such as memory or I/O register).	38–41

read on ➧

Topic	*Key Points*	*Visual Ref.*
service operation	★ This step is performed after the instruction is executed and is used to service pending interrupts or traps. • If no interrupts or traps are pending, this step is skipped. • If this step is needed, the CPU must get a service routine from memory. • If an *interrupt* is pending, the device supplies the CPU with the address of an *interrupt vector* that directs the CPU to the starting address of the service routine. On the other hand, a *trap* causes the CPU to jump to a fixed memory location containing a trap vector. The trap vector contains the starting address of an error handling routine. • The CPU steps through the routine. The last instruction returns the CPU to the main program.	42–46
ROM states	★ The operations just described are broken down into 4 ROM states: • fetch – get the instruction • source – get one external operand (if needed) • destination – get second external operand (if needed) • execute – perform the instruction ★ Note that there are *two* ROM states for getting operands . . . source and destination. One of these states is used if the instruction calls for one *external* operand. Both are used if two external operands are required.	48

read on

Topic	*Key Points*	*Visual Ref.*
microprogram ROM	★ Fetch, source, destination, and execute are called ROM states because CPU operation is controlled by a *Read-Only Memory*, or *ROM.* • This ROM contains a hardwired micro-program that supplies timing signals for each instruction. • The ROM is not an external unit – it's part of the CPU. • All CPUs except the 11/15 & 11/20 use this ROM; the 11/15 & 11/20 use flip-flops in place of the ROM.	49
instructions versus ROM states	★ An instruction may use *two, three,* or *four* ROM states depending on the number of operands and where they are located. ★ The number of bus cycles required by any ROM state varies. • fetch – uses DATI to get instruction from memory. • source & destination – if operand is stored internally, no bus cycle is used; if stored externally, it uses one or more DATI cycles. • execute – uses DATO or DATOB if result is stored externally; otherwise, no bus cycle is required.	50–57
unibus control	★ Handles all address, control, and data flow between processor and the devices that are connected to the Unibus. ★ Priority arbitration logic is also part of the Unibus control.	59

read on ➧

Topic	*Key Points*	*Visual Ref.*
data transactions	★ Unibus control selects slave by placing slave address on the bus.	60
	★ Specifies type of transfer by asserting or clearing C0 and C1 lines.	
	★ Generates MSYN to inform slave it is about to engage in a data transfer.	
	★ Slave responds with SSYN signal.	
	★ Data is either strobed into Unibus control or sent out to the slave.	
interrupt handling	★ Unibus control receives an INTR signal, a vector address, and a MSYN signal from the device that is initiating the interrupt.	61
	★ Unibus control responds to the interrupt by issuing a SSYN signal to notify the device that its interrupt will be serviced.	

read on

Topic	*Key Points*	*Visual Ref.*
trap handling	★ The Unibus control is responsible for trap handling when a processor error occurs.	62–70
	• Traps are similar to interrupts. They cause the CPU to exit from a main program and jump to a predetermined memory location.	
	• Traps are initiated internally (not externally) by hardware that detects certain conditions such as power fail or illegal instructions.	
	★ The Unibus control responds to these conditions by placing the address of a trap vector on the bus. The trap vector resides in memory and specifies the starting address of the appropriate error handling routine.	
	★ Unibus control performs a DATI to retrieve the starting address of the error handling routine from memory.	
	★ The starting address is routed to the PC. The processor can now retrieve and execute the first instruction in the error routine.	
	★ The error routine is then handled in the same manner as any other program.	
trap handling signals	★ The Unibus control places the address of the trap vector on the bus address lines.	71
	★ Clears C0 and C1 lines to specify a DATI bus cycle.	
	★ Asserts MSYN indicating it wants to retrieve data from memory.	
	★ Memory places starting address of trap handling routine on the bus data lines and asserts SSYN so that Unibus control can strobe in the address.	

read on ➧

Topic	*Key Points*	*Visual Ref.*
data manipulation logic	★ The data manipulation logic performs three prime functions: • Decodes all instructions to determine the operation to be performed, the location of operands, and the storage location for the result of the task. • Performs required arithmetic operations such as add or increment, and required logical operations such as complement or logical ANDing and ORing. • Coordinates all activities within the processor by routing information to and from the other elements (Unibus control and GPRs).	72–75
general-purpose registers (GPRs)	★ Every PDP-11 processor has a number of internal storage elements called general-purpose registers, or GPRs. ★ Large processors (i.e., PDP-11/45 and PDP-11/70) have 16 of these registers; the other PDP-11 processors have 8 GPRs. ★ The eight GPRs are numbered R0 through R7. ★ Each register has a unique address so that it can be both read and loaded. ★ All GPRs are 16-bit registers.	77–79

read on ➧

Topic	*Key Points*	*Visual Ref.*
multi-purpose registers	★ Registers R0 through R5 are multi-purpose registers.	80–87
	★ These registers are not dedicated to any specific function; their use is determined by the instruction that is decoded.	
	• They can be used for *operand storage.* For example, contents of two registers can be added and stored in another register.	
	• They can contain the *address* of an operand or serve as *pointers* to the address of an operand.	
	• They can be used for *automatic incrementation,* or *decrementation.*	
	• They can be used as *index registers* for program relocation.	
program counter register (PC)	★ Register R7 is the program counter (PC) which points to the next instruction word.	88, 89
	★ Whenever an instruction is fetched from memory, the program counter is automatically incremented by *two* to point to the *next* instruction *word.*	
stack pointer register (SP)	★ Register R6 is the hardware stack pointer (SP) that keeps track of the latest entry on the stack.	90–95
	★ The stack pointer moves up (the memory address is decremented by 2) as items are added to the stack and moves down (the memory address is incremented by 2) as items are removed. Therefore, it always points to the top of the stack.	
	★ The hardware stack is used during trap or interrupt handling to store breakpoint information allowing the processor to return to the main program.	

read on ➧

Topic	*Key Points*	*Visual Ref.*
stack example	★ When a device interrupts the CPU, the stack pointer (R6) is first decremented by 2. A status word is then placed on the stack at the location specified by R6. The CPU then decrements the stack pointer again and places the current PC value on the stack. ★ If a second interrupt occurs, the CPU again stores the current status word and PC on the stack. ★ If a third interrupt occurs, the process is repeated, causing the stack pointer to move up so it points to the top of the stack. ★ After the third interrupt is serviced, the PC and status word at the top of the stack are removed to allow return to the proper point in the routine for the second interrupt. ★ Once the second interrupt is serviced, the CPU again removes the status word and PC to permit servicing of the first interrupt. ★ When the first interrupt is serviced, the CPU can remove the last two entries on the stack and return to the proper spot in the main program. ★ In the above cases, the SP is autodecremented when information is being added to the stack; and is autoincremented as information is being removed from the stack.	96–100

read on

Topic	*Key Points*	*Visual Ref.*
general-purpose registers: large PDP-11 CPUs	★ Sixteen GPRs are implemented in large PDP-11 processors, such as the 11/45 and 11/70. ★ There are 12 multi-purpose registers instead of six. • Registers divided into two identical sets of 6 each. • Each set labeled R0 through R5. • Only one set can be used at a time. • They perform the same functions as R0 through R5 in other PDP-11 processors.	101–102
three stack pointer registers	★ The large PDP-11 processors (e.g., the 11/45 and 11/70) have three operating modes (Kernel, Supervisor, and User), three hardware stacks, and three stack pointer (SP) registers. ★ Each stack pointer register is called R6. ★ Only one stack pointer register can be used at any given time. In effect, there is one stack for each of the three operating modes.	103
program counter (PC)	★ Large processors, such as the 11/45 and 11/70, have one program counter (PC) register called R7. ★ The function of this PC register is identical to the function of the PC that is implemented in the small and medium-size PDP-11 processors.	104

read on

Topic	*Key Points*	*Visual Ref.*
processor status word	★ The processor status word (PSW) is stored in a 16-bit register in the CPU. • The PSW register has its own address so that it can be read and loaded. • The PSW address is part of the I/O device address structure; therefore, the address is in the top 4K of the total address space. • Only the CPU can address the PSW. The PSW is available to the programmer but *is not* available to any other bus device.	107
PSW make-up	★ The first 4 bits (0–3) indicate conditions resulting from the instruction and are called "condition codes." • *C bit (bit 0)* – a "carry" indicator. When set, indicates the previous operation ended with a carry out of the most significant bit. • *V bit (bit 1)* – an "overflow" indicator. When set, indicates the previous operation resulted in an arithmetic overflow. • *Z bit (bit 2)* – a "zero" indicator. When set, indicates the operation resulted in zero. • *N bit (bit 3)* – a "negative" indicator. When set, indicates the result of the operation was a negative value.	108–113

read on

Topic	*Key Points*	*Visual Ref.*
CPU priority level	★ Bits 5, 6, and 7 in the PSW establish a specific priority level for the CPU. • Level 7 is the highest; level 0 is the lowest. • The level can be raised or lowered by way of the PSW. ★ The prime purpose of the CPU priority level is to *"mask"* or *"block out"* device interrupts occurring at one or more of the BR levels. • For example, if the CPU's level is set at 5, the CPU *cannot* be interrupted by any BR5 or BR4 device although it can service interrupts at the BR6 or BR7 level. • The CPU's priority level can be continually changed by the program to mask out lower level interrupts while servicing higher priority devices.	114–121
trap indicator	★ Bit 4 of the PSW is the trap (T) bit that, when set, causes the CPU to automatically trap to a debugging routine. ★ When the T-bit is set, the CPU completes the current instruction and then traps to vector location 14 which contains the starting address of the debugging routine.	122, 123
remaining PSW bits	★ *Bits 8–10* – these bits are not used but are available for system expansion. ★ *Bit 11* – used to select one of the two register sets. (This bit is used in the large PDP-11 processors, such as the 11/45 and 11/70.) ★ *Bits 12–15* – these bits are only used in PDP-11 systems that contain memory management hardware.	125

read on ➧

Topic	*Key Points*	*Visual Ref.*
use of the PSW	★ When the CPU is interrupted for service, it stops its current job and stores the PC and PSW in the hardware stack. ★ The CPU then takes a *new* PC and PSW from memory. The new PSW provides the initial status to be used with the service routine; for instance, the appropriate priority level for the CPU. ★ Once the service routine is finished, the CPU re-loads the old PC and PSW so it can resume the interrupted program. The PSW indicates the results of the last instruction executed *prior to the interrupt.* The PSW also re-establishes the CPU's priority level which existed prior to the interrupt. The CPU can now continue with the original program.	126–128

read on

When you have completed the study unit, please take this self-scoring test. Then compare your answers against the "answer sheet" which can be obtained from your supervisor. Based on your test results, either review the appropriate material in this study unit, or proceed to the next unit in the series.

1 The processor consists of three main functional units. Match each of these units with its corresponding functions.

a. Unibus control

b. General-purpose registers

c. Data manipulation logic

( ) performs all arithmetic and logic operations

( ) arbitrates which device gets the bus next

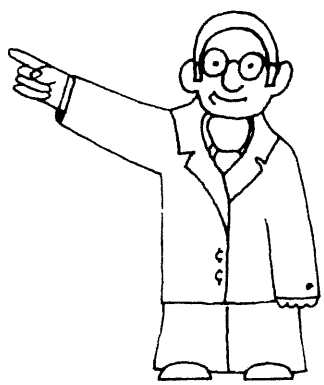
( ) handles communication between the processor and memory

( ) internal storage element

( ) decodes incoming instructions

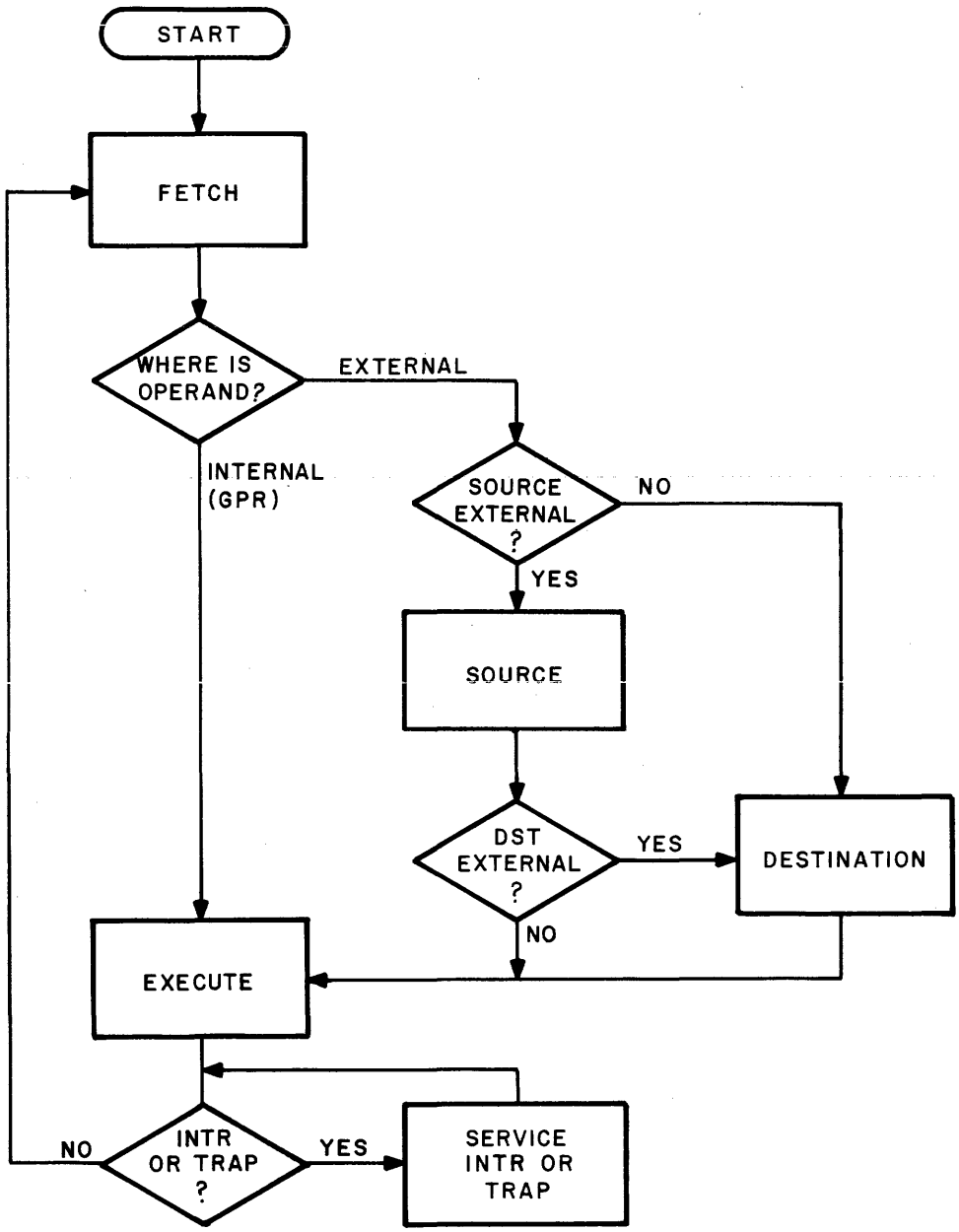
( ) handles communication between processor and I/O devices

( ) handles information flow to and from the GPRs



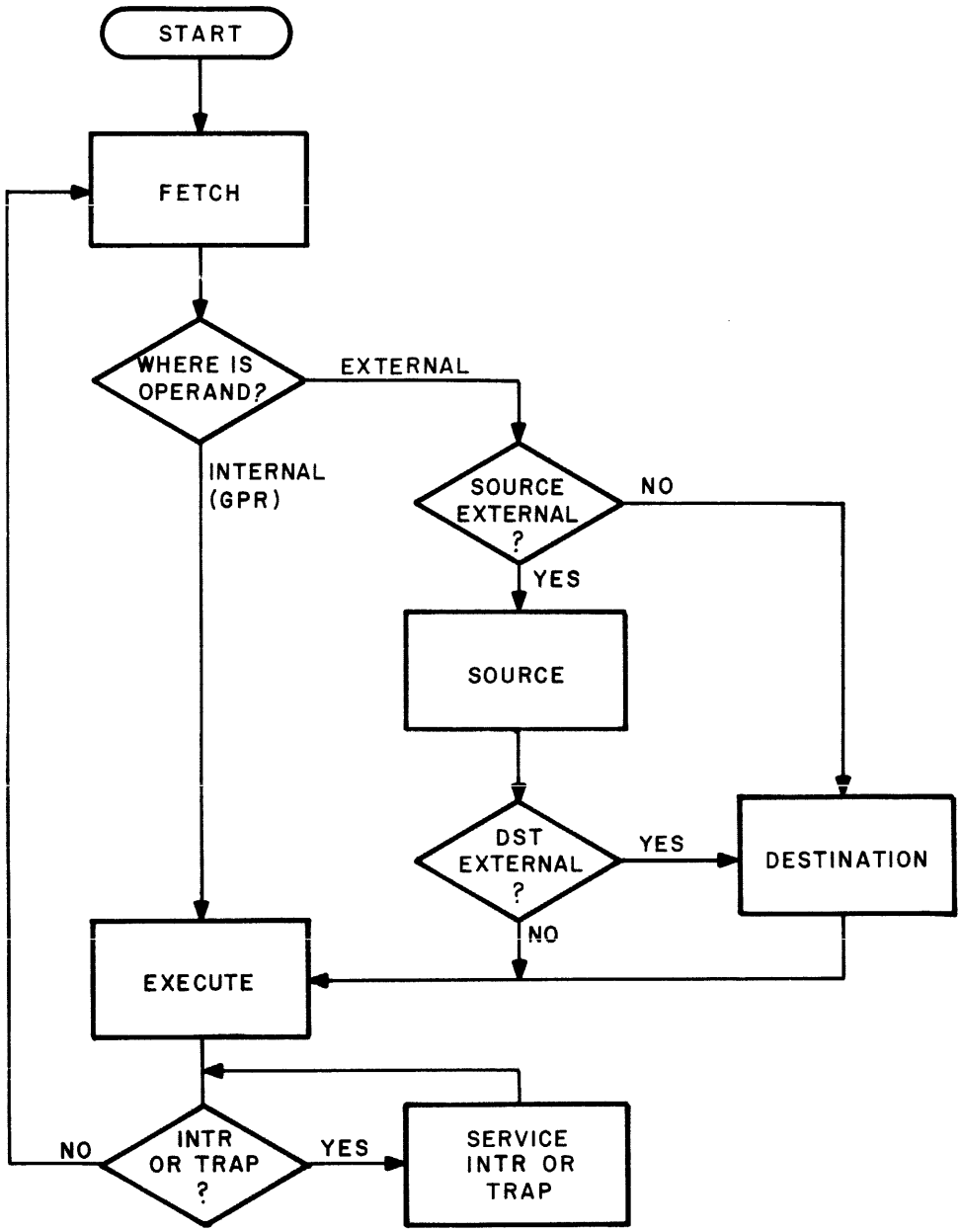
read on

2 An instruction states that the contents of GPR three is to be cleared. Draw a line on this diagram to indicate the path that will be followed when executing this instruction. No traps or interrupts are pending.



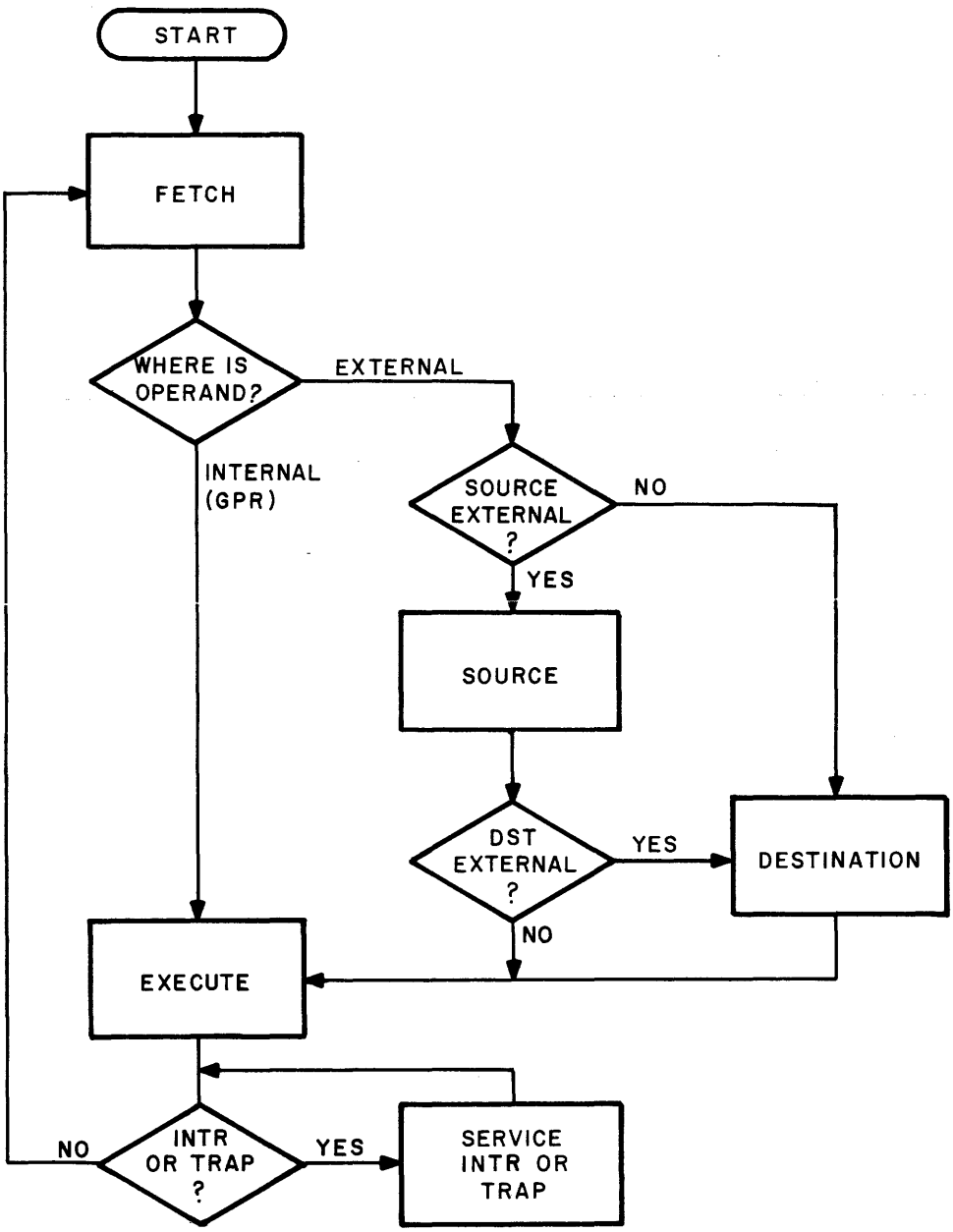
read on

3 In this case, what flow will take place to execute an instruction that will increment an I/O register? The *address* of this I/O register is stored externally in memory and there are no traps or interrupts pending.



read on

4 In this case, what flow will take place to execute an instruction that adds the value in an I/O register to the value in a memory location and stores the result back into the original memory location? Assume that a device makes an interrupt request during execution of this instruction. Also, assume the addresses of the I/O register and the memory location are stored in a pair of GPRs.



read on

5 List the bus cycles that are required in problems 2, 3, and 4. If no bus cycles are required for a particular operation, write "none". Assume that each instruction is operating on full words, rather than bytes.

	Problem 2	Problem 3	Problem 4
Fetch Instruction			
Get Source Operand			
Get Destination Operand			
Store Result			

6 If an MSYN signal is issued and the addressed slave device fails to respond, a "time-out" condition occurs. This condition causes the CPU to:

(a) Abort the data transaction and proceed to the next instruction.

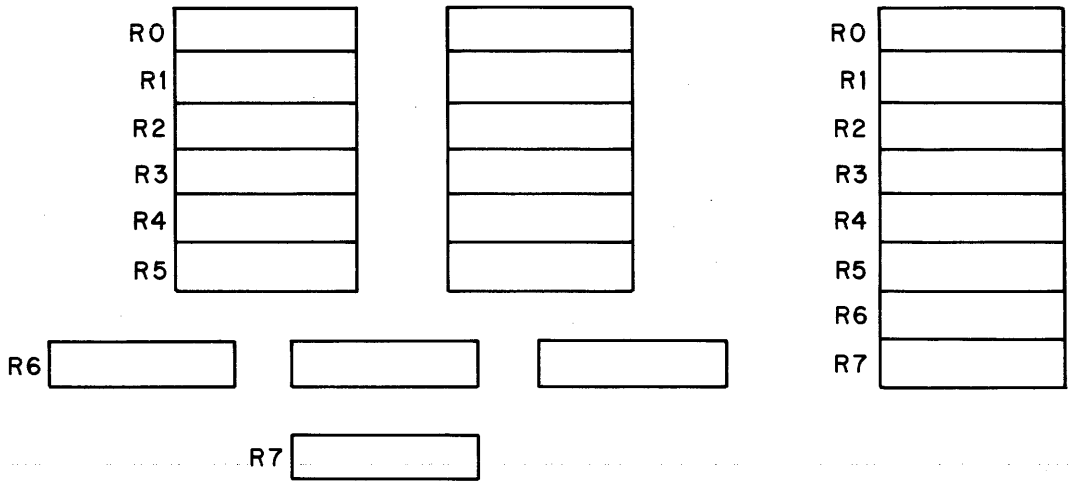
(b) Try again, by sending the MSYN signal a second time.

(c) Jump to a fixed memory location that contains an interrupt vector.

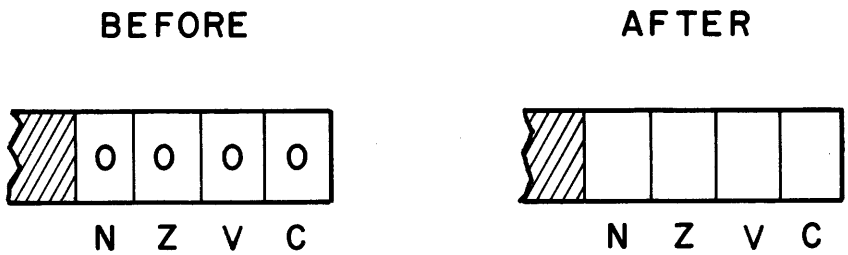
(d) Jump to a fixed memory location that contains a trap vector.

read on

7 The following diagrams indicate the 16 GPRs used in the large PDP-11 processors and the 8 GPRs used in all other PDP-11 systems. In each case, indicate which registers serve as stack pointers (SP) and which registers serve as program counters (PC).

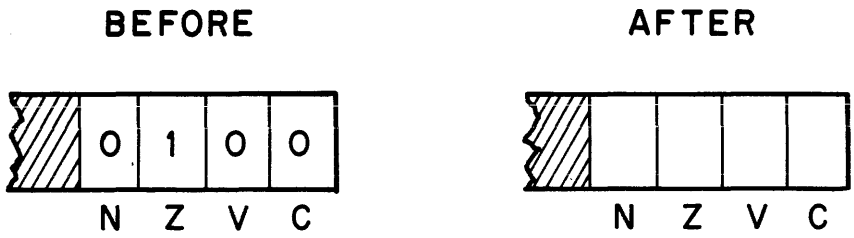


8 The following is a diagram of the condition code bits in the processor status word. Assume the processor adds the numbers 000 002 and 177 776. What will be the new values of the condition codes?



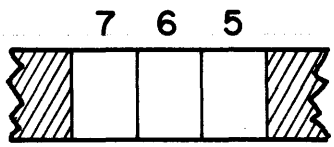
read on

9 Now assume that our next instruction adds two positive values and produces a negative result. What are the new condition code values?

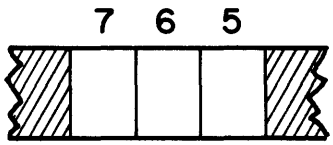


10 Certain bits in the PSW determine the priority of the processor. In the following example, we would like you to fill in the appropriate priority bits for each step.

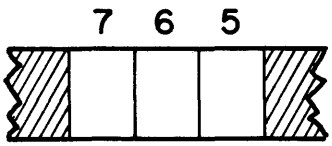
a. The processor is executing the main program and is operating at level three.



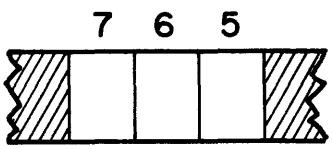
b. A BR7 device interrupts the processor, so the processor begins a service routine.



c. After completion of service, an interrupt at BR6 is pending, so the processor services it.

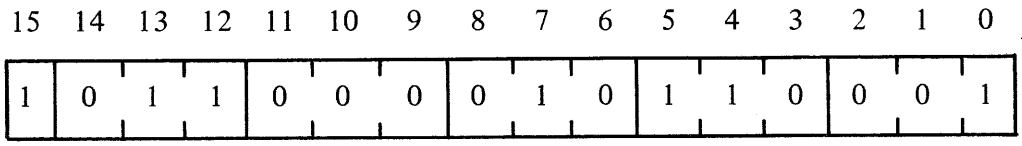


d. The processor resumes the main program it was executing prior to being interrupted.



read on

11 The following is a PSW word from a PDP-11/45 or PDP-11/70 processor:



Indicate what each of the following mean:

a. Bit 11:

b. Bits 5, 6, 7:

c. Bit 4:

d. Bit 0: