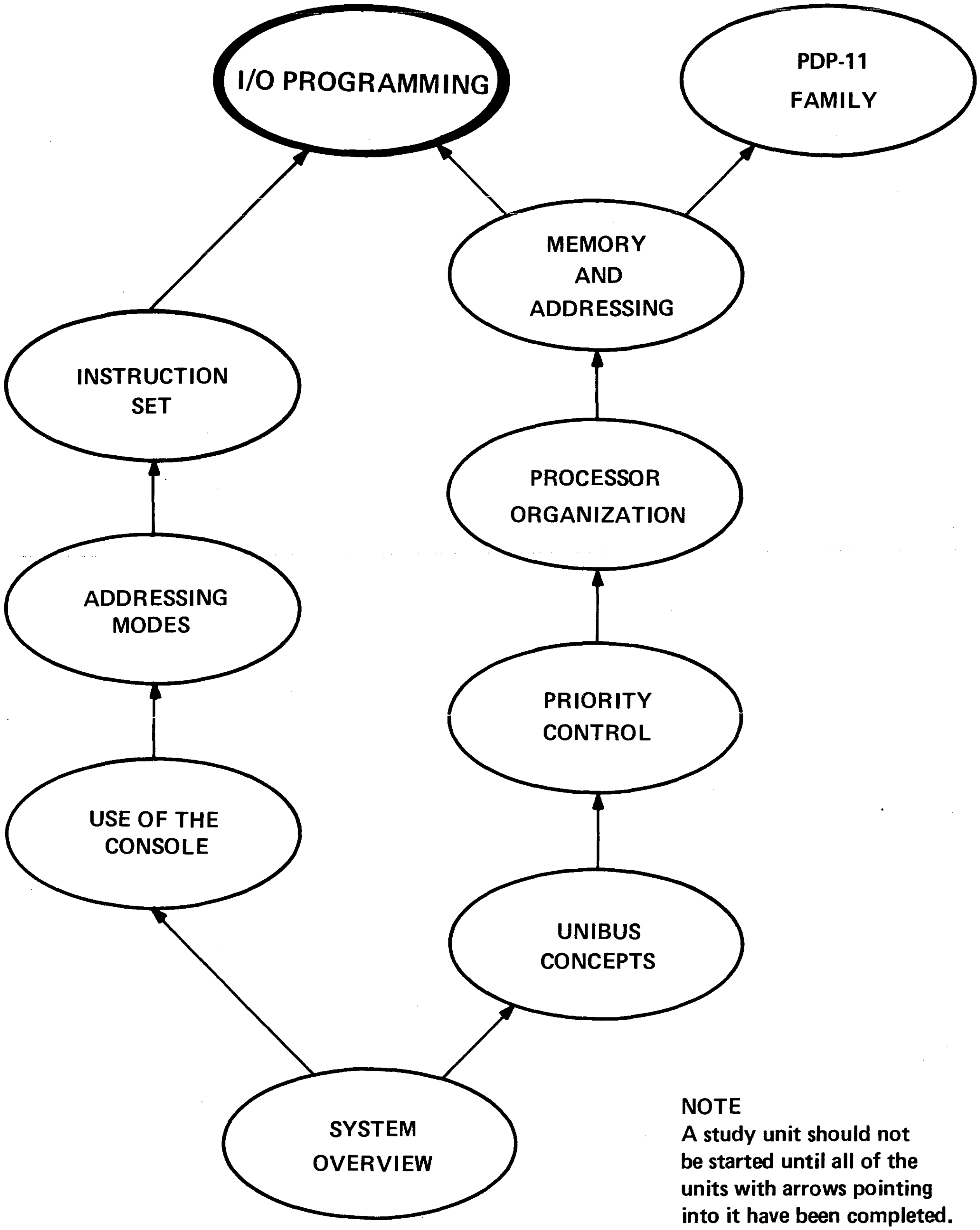
I/O Programming

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

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 3–9) AND COMPLETE EXERCISES (Pages 15–39)

↓

NOW RUN FILM CARTRIDGES C & D

↓

REVIEW MATERIAL (pages 10–14)

↓

TAKE TEST & CHECK RESULTS (pages 41–46)

↓

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 . . . .

★ Analyze and write small I/O programs that are used by the CPU to communicate with peripheral devices.

★ Explain how the device interface registers are used to transfer data and control information between the CPU and a peripheral.

★ Describe the typical bit assignments of a Control and Status register.

★ Describe the advantages of the two I/O programming methods – programmed data transfers and program interrupts – and give examples of each.

★ Explain how program subroutines and interrupt service routines are used in I/O programming.

★ Explain the basic purpose of program loaders.

★ Explain the functioning of the individual instructions that make up the Bootstrap Loader program.

★ Describe how the Bootstrap Loader is used to bring the Absolute Loader program into memory and how the Absolute Loader, in turn, is used to enter other paper tape programs.

★ Explain why the Bootstrap Loader and Absolute Loader programs share several memory locations and show how they interact when they are executed.

additional resources

• **PDP-11/04/05/10/35/40/45 Processor Handbook**	Review Chapter 5, Paragraphs 5.1–5.3, Programming Techniques.
• **PDP-11 Peripherals Handbook**	Read Chapter 2, Programming.
• **PDP-11 Paper Tape Software Programming Handbook**	Read Chapter 6. (Paragraph 6.1 describes the Bootstrap Loader; Paragraph 6.2 describes the Absolute Loader.)

review material

The following material is covered in this study unit:

Topic	*Key Points*	*Visual Ref.*
interface	★ Peripherals consist of the peripheral device and an interface unit which connects the peripheral to the Unibus. The interface performs four primary functions: ● Recognize device address ● Interrupt processor operations ● Control and monitor peripheral operation ● Buffer data	7–8
data buffer register (DBR)	★ Holds data sent to or from the peripheral. Used in most peripheral interfaces, it is usually an 8-bit byte buffer or a 16-bit word buffer. ● The DBR provides serial to parallel data conversion to the Unibus and parallel to serial data conversion to devices such as the Teletype.® ● The DBR compensates for slow operating devices that otherwise would tie up the Unibus for long periods.	10–14
control and status register (CSR)	★ A 16-bit register that performs command and monitoring functions. Included in all interfaces. A DATO or DATOB is used to load the CSR with control information. A DATI is used to read or monitor the peripheral status information contained in the CSR.	15–17

® Teletype is a registered trademark of Teletype Corporation.

read on ➧

Topic	*Key Points*	*Visual Ref.*
CSR bit assignments	★ CSR is byte addressable. Typical bit assignments are: Bit 0 – Start or enable Bits 1–3 – Device functions Bits 4–5 – 18-bit extended address Bit 6 – Interrupt enable Bit 7 – Done or ready Bits 8–10 – Unit selection Bit 11 – Device busy Bits 12–15 – Errors	18–24
types of CSR bits	★ There are three types of CSR bits ● Read only – such as bit 11, Device Busy ● Write only – such as bit 0, Start or Enable ● Read or write – such as bit 6, Interrupt Enable It is not possible to read from write-only bits or write to read-only bits. A load command such as a MOVB instruction is used to set bits in a CSR. A MOV instruction can be used to read the CSR status.	22–26

read on

review material

Topic	Key Points	Visual Ref.
CSR–DBR, integrated operation	★ The CSR and DBR function together. Example: read data – • Set CSR control bits (start and interrupt enable) which start read operation • Data sent from device to DBR; done bit is set • Interface sends interrupt to the CPU when all the data is assembled in the DBR • Start interrupt service routine • Read data from DBR to Unibus	27–32
special purpose registers	★ Some (DMA) devices also have other registers: • Word count register • Memory address register • Data address register • Maintenance mode register	36
addressing peripheral registers	★ Each peripheral register is assigned a discrete address which is located in the top 4K of possible addresses. The CPU makes no distinction between memory addresses and peripheral addresses. • 16-bit addresses containing 1's in bits 13, 14 and 15 are converted to 18-bit addresses; i.e. address bits 13–17 are all ones. • Address bits 3–12 specify the peripheral device. Bits 1 and 2 identify the particular CSR or DBR within the interface. Bit 0 specifies the type of operation, word or byte.	38–45

read on

Topic	*Key Points*	*Visual Ref.*
teletype	★ Contains two separate units: A reader/keyboard and a printer/punch. Each unit has a CSR and a DBR. ● Reader/keyboard DBR is used during DATI operations to hold read data destined for the Unibus. Printer/punch DBR is used to store data sent to the Teletype during DATO and DATOB operations. ● Reader/keyboard CSR is called the keyboard status register or TKS. Printer/punch CSR is called the punch status register or TPS. The TKS uses 4 bits: Reader enable, interrupt enable, done, and busy. The TPS uses 3 bits: maintenance, interrupt enable, and ready.	46–55
ASCII data	★ Data is coded on Teletype tape in ASCII format. Data is transmitted and received by Teletype in Teletype code. Teletype code is translated to ASCII code by subtracting 200_8.	49
instruction set	★ The PDP-11 has one instruction set. These instructions can be used in I/O programming. Examples: INCB@#177560 (Set reader enable bit) MOVB#101, TKS (Set reader enable and interrupt enable bits) MOVB@#177562,R3 (Move data in reader DBR to GPR 3) TSTB@#177564 (Test state of ready bit in punch status register. Used with a branch instruction) MOV#340,@#PS (set processor priority level to 7; PS=177776)	58–63

read on ➧

review material

Topic | *Key Points* | *Visual Ref.*

programmed data transfer | 66–79

★ Program remains in a loop (CPU waits) until peripheral is ready to send or receive data. The program loop compensates for the relatively slow speed of the peripheral.

- In a DATO or DATOB operation the data is transferred from a memory location to the TPB. The program loops until the TPB is ready to accept the data.
- In a DATI operation the data is transferred from the TKB to a memory location. The program loops until the data is completely assembled in the TKB.

data echoing | 80–86

★ Allows keyboard entries to be printed. Information from the reader keyboard unit of the Teletype is transferred to the printer/punch unit by way of the CPU.

★ Example of keyboard echo routine.

```
ECHO:     INC TKS          ; start reader.
LOOP1:    TSTB TKS         ; wait for
          BPL LOOP1        ; reader done.
LOOP2:    TSTB TPS         ; wait for
          BPL LOOP2        ; punch ready.
          MOVB TKB, TPB    ; move byte from
                           ; reader to punch.
```

program interrupts | 88–93, 99–105

★ Better use is made of processor time using program interrupts in place of the program loop method.

- Once the peripheral is started, the main program is re-entered and other operations can be performed. Then, when the data is ready to be sent or received, an interrupt is issued by the interface. The main program is stopped and an interrupt service routine is executed which transfers the data. Finally the main program is again re-entered and its execution is continued.

read on ➧

Topic	*Key Points*	*Visual Ref.*
interrupt sequence	★ The issuance of an interrupt involves an orderly progression of events. It begins when the peripheral is ready to send or receive data: ● Peripheral interface sends Bus Request (BR). ● Processor responds with Bus Grant (BG). ● Peripheral interface asserts SACK and clears Bus Request (BR). ● SACK causes processor to clear BG. ● Peripheral interface asserts BBSY as soon as BBSY and SSYN are clear on the Unibus. ● Peripheral interface raises INTR and the vector address on the Unibus data lines. SACK is cleared. ● Processor responds to the vector address with SSYN. ● SSYN clears the INTR, vector address and BBSY sent from the peripheral interface. ● Processor now asserts BBSY and becomes bus master in order to service the interrupt.	95–97
interrupt programming considerations	★ The programmer is responsible for several program components: ● Instruction to set up stack pointer. Usually at beginning of main program. ● Instructions to load new PC and PSW into interrupt vector locations. ● Instructions to set certain bits in peripheral CSR (device enable, interrupt enable, etc.). ● Interrupt service routine which concludes with an RTI or RTT instruction.	106–124

read on ➧

Topic	*Key Points*	*Visual Ref.*
interrupt program chronology	★ The typical I/O program sequence begins during execution of the main program:	125–128

- Main program needs data from the peripheral.
- JSR instruction in main program causes a read subroutine to be entered. The JSR instruction includes a label that defines the start of the subroutine.
- Start read subroutine. Initial instructions prepare for interrupt by loading new PC and PSW into vector address.
- Subroutine instruction sets device enable and interrupt enable bits in peripheral CSR. Peripheral starts to read data.
- RTS instruction in subroutine returns operation to main program. Main program can now perform other unrelated functions.
- Data is read and assembled in reader DBR. Done bit is set. Interrupt and vector address are sent to CPU.
- CPU loads new PSW and PC. Old PSW and PC is pushed on the stack. CPU exits from main program and starts interrupt service routine.
- Interrupt service routine transfers data in reader DBR to a GPR or memory address. Data is then tested and used as required. An RTI instruction concludes the service routine.
- The old PSW and PC are popped from the stack and reloaded. The main program picks up where it was interrupted.

read on ➧

Topic	*Key Points*	*Visual Ref.*
program loaders	★ Program loaders are small programs that allow the operator to input or load other larger and more complex programs into PDP-11 memory.	138
loader programs	★ When first installed a computer's memory does not have the capability of accepting programs.	144
	● A small loader program is manually entered into memory through the console. This loader program allows the CPU to input information.	145
bootstrap and absolute loader programs	★ The PDP-11 uses two loader programs to bring in other programs stored on paper tape – a bootstrap loader and an absolute loader.	147
	● The bootstrap's function is to bring in the absolute loader program.	148
	● The absolute loader, in turn, brings in any standard paper tape program the operator wishes to input.	
location of bootstrap program in memory	★ The bootstrap program is manually entered through the switch register and is composed of fourteen instruction words that reside in the highest 4K of memory.	149–150
	● The two most significant octal digits of each instruction's address are determined by the size of the PDP-11 memory.	150–151
procedure for running bootstrap program	★ First the bootstrap program is toggled in through the switch register. Then the absolute loader paper tape program is inserted into the input device, i.e., usually either a high speed paper tape reader or a Teletype reader. Finally the starting address of the bootstrap loader is toggled in and the LOAD ADDRESS and START switches are then pressed to initiate execution of the bootstrap program.	152–153

read on ➧

review material

Topic	*Key Points*	*Visual Ref.*
operation of the bootstrap loader program	★ The bootstrap program causes the absolute loader program to be read from the paper tape and deposited in memory. Each time the CPU loops through the bootstrap program, one byte from the absolute loader program is stored in memory.	155–156
operation of the absolute loader program	★ The end of the absolute loader program overlays, in memory, the beginning of the bootstrap loader program.	157–158
	● This overlay modifies the beginning of the bootstrap causing the program counter to break out of the bootstrap program and branch into the absolute loader program.	
	● A portion of the absolute loader program is now executed restoring the bootstrap to its original keyed in condition. The program then jumps to a halt instruction and any of the paper tape programs can be loaded through the input device simply by pressing the continue switch.	160

read on ➧

TABLE A
BOOTSTRAP LOADER PROGRAM

ADDRESS	CONTENTS	MNEMONICS		
	017 400	LOAD	=	17400
	017 744	●	=	LOAD+344
017 744	016 701	START:	MOV DEVICE, R1	
017 746	000 026			
017 750	012 702	LOOP:	MOV#.-LOAD+2, R2	
017 752	000 352			
017 754	005 211	ENABLE:	INC@R1	
017 756	105 711	WAIT:	TSTB@R1	
017 760	100 376		BPL WAIT	
017 762	116 162		MOVB 2(R1), LOAD(R2)	
017 764	000 002			
017 766	017 400			
017 770	005 267		INC LOOP+2	
017 772	177 756			
017 774	000 765	BRNCH:	BR LOOP	
017 776	177 560 (TK)	DEVICE:		
	or 177 550 (PR)			

NOTE

a. LOAD=START-344. Data cannot be loaded into memory below this address.

b. ●=LOAD+344 defines the starting address (017 744) of the Bootstrap Loader.

read on

THE BOOTSTRAP LOADER (TABLE A)

Visual Ref
163–201

Instruction Mnemonic	General Statement	Analysis/Comments
START: MOV DEVICE,R1	GPR 1 is loaded with the address of the input device's CSR.	Input device is usually a high speed paper tape reader or teletype.
LOOP: MOV #.-LOAD+2,R2	GPR 2 is loaded with an address displacement.	Since immediate mode is used, the address displacement is in location LOOP+2.
ENABLE: INC @R1	The input device is enabled.	The increment instruction places a logical 1 into the reader enable bit (the LSB).
WAIT: TSTB@R1	The program waits until a byte is read from the absolute loader tape and assembled in the DBR.	Monitors the reader CSR done bit (7). When bit 7 is a logical 1, the DBR is ready to transfer data.
BPL WAIT	When a byte is read, the program counter falls through the branch to next instruction.	The branch instruction keeps the CPU in a wait loop until bit 7 is a 1; when bit 7 switches to a 1, the loop is broken.
MOVB 2(R1),LOAD(R2)	The byte of information assembled in the DBR is moved into memory.	Load and address displacement are summed to develop the storage address.
INC LOOP+2	Address displacement is incremented.	Generates next sequential storage address.
BRNCH: BR LOOP	Returns program counter back to the beginning of program to repeat read and storage sequence.	Unconditional branch instruction.

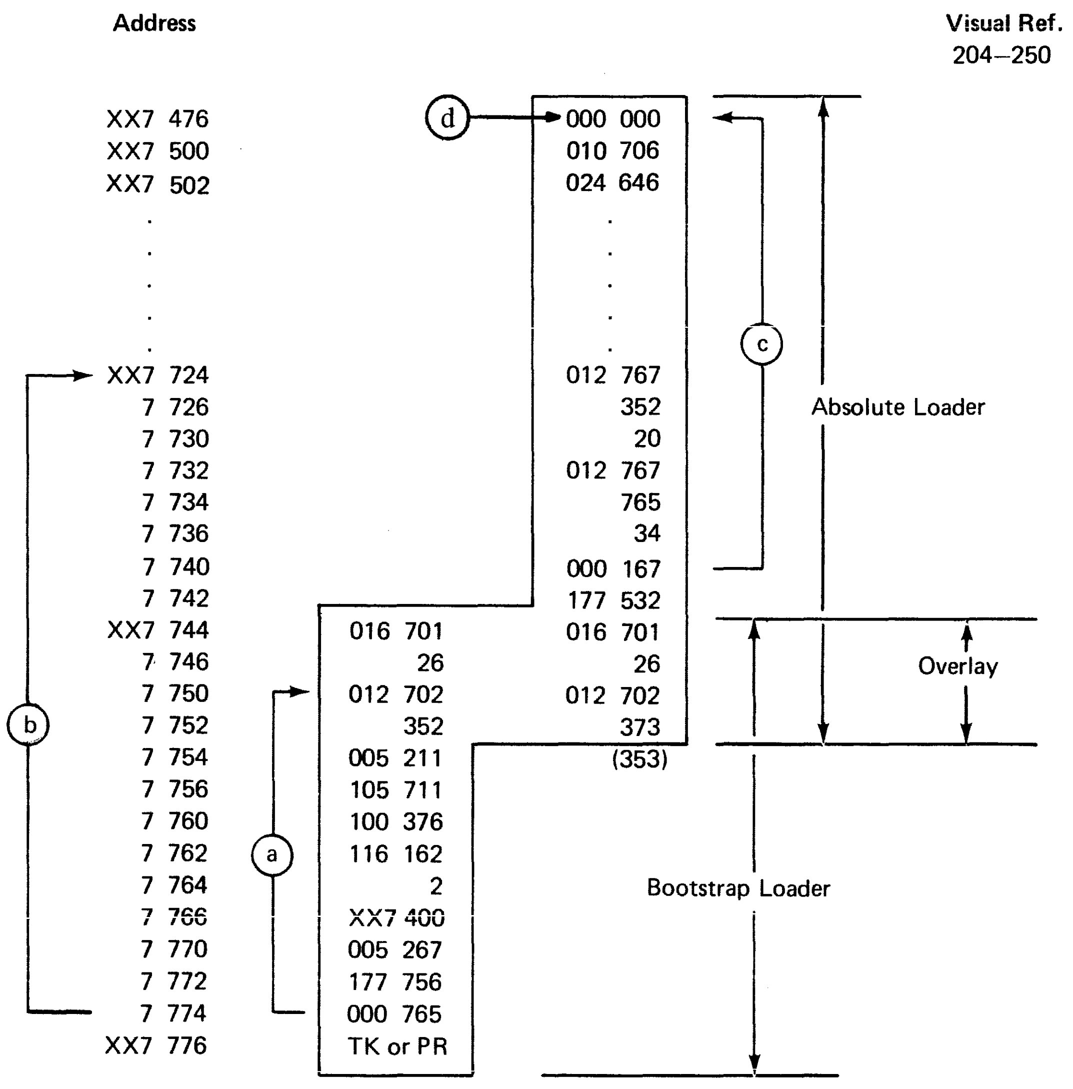
The relationship between the bootstrap loader and absolute loader is shown in Figure 1.

NOTE

For an additional explanation, refer to Paragraph 6.1.6.3 in the *PDP-11 Paper Tape Software Handbook.*

read on ➧

Figure 1
MEMORY LAYOUT OF BOOTSTRAP & ABSOLUTE LOADER



TK = 177 560 Keyboard
PR = 177 550 Hi Speed Reader
XX is a function of memory size

(a) Branch in Loc. XX7 774 returns PC to Loc. XX7 750 before 373 is overlayed in Loc. XX7 752.

(b) Branch in Loc. XX7 774 directs PC to Loc. XX7 724 after 373 is overlayed in Loc. XX7 752.

(c) Absolute loader jumps to Halt in Loc. XX7 476 after restoring Bootstrap Loader.

(d) Halt after bootstrap load.

read on

I/O PROGRAMMING

The study exercises in this section of your workbook are intended to increase your familiarity with I/O programming concepts. Check the answer at the back of the workbook after working each problem. Remember, this answer reflects *one* possible solution to the problem. It is not the only answer; yours may be just as correct. After you've completed all the exercises, turn the A/V playback unit back on and complete parts C and D of this study unit. You may also wish to replay the first portion of this study unit – feel free to do so.

EXERCISE 1

The status register may be used to indicate an error or a fault in the peripheral device. For example, bit 15 is set in the high speed paper tape reader CSR, or PRS, if the reader is out of tape. Before each read operation it is desirable to test the state of this bit. If bit 15 of the status register is set, the data in the register is considered a negative value. The following two instructions provide one way to test this bit:

```
TST PRS

BMI NOTAPE
```

What happens in this program when the reader has run out of tape?

read on ➧

EXERCISE 2

The BIT instruction can be used for testing any of the 16 bits in a status register. For example, the following two instructions are used to check on the availability of data in the Teletype reader buffer. As you recall, this is indicated by bit 7 of the TKS being set:

```
WAIT:   BIT #200, TKS

        BEQ WAIT
```

Here the contents of TKS is *logically* ANDed with 200. Until the DONE bit (#7) is set, the sum is zero; the conditions for the BEQ are met and the program remains in the loop. The DONE bit is then set when the data is available in the TKB. Now the sum of 200 and the TKS contents is no longer zero and the program falls through to the next instruction.

At this time we'd like you to use the BIT instruction in a two instruction sequence that causes a branch to an error routine (NOCARD) if the input hopper of the high-speed punched card reader (CD11) is found to be empty. The card reader CSR is called the CDST and its address is 172460. Bit 2 of the CDST is the hopper check bit.

read on ➧

EXERCISE 3

What is the purpose of the following subroutine? Also indicate, in the space provided, the function of each instruction.

(R1) = 5000

```
DATA1:    MOV #100, R0

ENCODE:   MOVB (R1)+, R2

          ADD #200, R2

WAIT:     BIT #200, @#177564

          BEQ WAIT

          MOV R2, @#177566

          DEC R0

          BNE ENCODE

          RTS R5
```

read on

EXERCISE 4

Write a subroutine that allows 100_8 keyboard generated characters to be stored in a memory area with a starting address of INDATA. Afterwards return to the main program. Explain what each of your instructions does.

read on ➧

EXERCISE 5

In the preceding problem (Exercise 4) we stored 100_8 characters in a memory area with a starting address of INDATA. The stored data is in the Teletype code. Write a subroutine (TRANS) that converts the Teletype-coded data to the ASCII code and stores it in a memory area with a starting address of TDATA. Explain what each instruction does.

read on

EXERCISE 6

Programs can be written that allow the computer operator to manually control the program's operation. In the program below the operator is allowed to store up to 64_{10} (100_8) characters from the keyboard. If the SPACE bar is pressed before all 64 characters are stored, the program is halted; additional characters are not stored.

```
        CLR TALLY

        MOV # INDATA, R1

PAWS:   TSTB TKS

        BPL PAWS

        CMP TKB, #240

        BEQ FINI

        MOVB  TKB,(R1)+

        INC TALLY

        CMP TALLY, #100

        BLT PAWS

FINI:   HALT
```

Modify the program so that when the space bar is pressed, or 64 characters have been stored, the program types out all the stored characters and then halts.

read on ➧

EXERCISE 7

The BIS and BIC instructions are useful for modifying one bit of a peripheral status register while leaving the other bits undisturbed. For example, bit 9 of the VR 20 Color CRT status register controls the color of the display, 0 for green and 1 for red. To change the color, it is necessary to change only this one bit. The instruction BIS #1000, CRTSR sets bit 9 in the status register without disturbing the other bits. They remain set or clear as the case may be. Write the instruction to set the interrupt enable bit of the Teletype reader status register, TKS.

read on

EXERCISE 8

Individual bits in a status register can be cleared with the BIC instruction. Write an instruction to clear the external clock enable bit, bit 1, of the ADCS, the Analog to Digital Subsystem CSR.

read on

EXERCISE 9

The interrupt mode of operation is used to reduce computer waiting time. In the program loop method, as we learned, the computer continually checks the state of the DONE flag (bit 7) in the peripheral CSR. Using the interrupt mode, the computer actually ignores the peripheral, running its own low-priority program until the peripheral requests service by setting the DONE bit. (The interrupt enable bit in the CSR must have been set at some prior point.) The computer completes the instruction being executed and then starts the interrupt service routine. The service routine may transfer the data involved and then perform some calculations with it. After the interrupt service routine has been completed, the computer resumes the program that was interrupted by the peripherals higher priority request.

In the program below assume that the high speed reader, which at some earlier time had been commanded to read a character, now initiates an interrupt while the instruction opposite the arrow is being executed.

```
MOV DATA, STORE

INC COUNT           ←————

CMP COUNT, TOPNUM

BLOS MOREAD
```

What instruction in this program is executed first after the interrupt service routine has been completed?

read on ➧

EXERCISE 10

For peripheral devices there are four different interrupt priority levels, 4 to 7, of which 7 is the highest. Some peripherals are assigned a higher priority than others. For example, the Teletype keyboard and the high speed punch both operate at a priority level of 4, while the card reader has a priority level of 6. The CPU normally operates its background program at a level below 4 so that it may be interrupted by any of the peripherals mentioned above. A peripherals service routine can be interrupted, but only by a peripheral which has a higher priority than the one being serviced.

Assume that the above mentioned peripherals are included in a system. At this time the CPU is running its background program. What happens if –

a) The card reader requests interrupt service?

b) Then, while the card reader service routine is running, the Teletype keyboard requests service?

read on ➧

EXERCISE 11

Considering the same system (Exercise 10), what happens if the CPU program is running and the keyboard requests service? Then, what happens when the card reader requests service after the keyboard service routine has been started.

read on

EXERCISE 12

A Teletype keyboard service routine might store data in memory as in the example below. Each time the keyboard is hit the service routine is entered.

```
KEYSRV:  MOV TKB, STOR

         RTI
```

What does the second line of the service routine do?

read on ➧

EXERCISE 13

It cannot be assumed that registers used in an interrupt service routine are not used elsewhere – in the main or background program, for example. Other peripheral service routines may also require the use of the same register. Therefore, there are two basic considerations when programming service routines:

a) Whenever a register is used in a service routine, it should never be assumed that it previously contained useless information. The first operation within the service routine should be to PUSH the present contents of each required register on a stack. The last operation just before the RTI should POP the contents back into the registers from the stack.

```
MOV R1, -(SP)          ; PUSH R1 on stack
          .
          .
          .
          .
MOV (SP)+, R1          ; POP stack into R1
```

b) Register data needed in a service routine should be stored via a label at the end of the service routine, freeing the register for use in the background program or in other service routines.

```
MOV R1, -(SP)
          .
          .
          .
MOV R1, SUM
MOV (SP)+, R1
```

Write the PUSH and POP instructions needed to preserve the original contents of R1 and R2 during the keyboard service routine.

read on ➧

EXERCISE 14

During a service routine, the data produced by a peripheral could be stored in memory, say at an address defined by the label STORWD. Two instructions are required for this purpose:

a) An instruction *executed at the beginning of the background program* to initialize an arbitrary label DASTOR with the first storage address.

```
MOV # STORWD, DASTOR
```

b) An instruction *within the interrupt service routine* to initialize a register with the contents of DASTOR.

```
MOV DASTOR, R1
```

Complete the following service routine labeled KEYSRV:

```
KEYSRV:  MOV R1, -(SP)     ; PUSH R1 on the stack

         - - - - - -       ; Initialize R1 with the first storage address.

              •

              •

              •

         MOV (SP)+, R1     ; Restore R1

         RTI               ; Return to previous program
```

read on ➧

EXERCISE 15

To the previous routine (Exercise 14) we can add instructions to:

a) Store the one word of data produced by the Teletype keyboard

b) Then save the next storage address in DASTOR

```
KEYSRV:  MOV R1,-(SP)       ;PUSH R1 on stack

         MOV DASTOR, R1     ;Initialize R1 with address

         MOVB TKB,(R1)+     ;Store a character

         MOV R1, DASTOR     ;Store next address in DASTOR

         MOV (SP)+, R1      ;POP R1 from stack

         RTI                ;Return to previous program
```

TITLE, a service routine for the Teletype printer, causes a character to be printed from a list whose first address is identified by the label MESAGE. Write the address initialization instruction required at the beginning of the background program as well as the service routine itself.

read on ➧

EXERCISE 16

Write an interrupt service routine that starts at an address with a label of PROFIL and keeps track of the number of times each letter (A through Z) is pressed on the keyboard. Use the label OCCUR as the starting address for the memory area where the individual counts are stored. If you refer to the ASCII code listing, you can see that an offset address can be derived from the code for each character. Remember, there is a difference between the ASCII code and the transmitted Teletype code.

read on

EXERCISE 1

Bit 15 is set. Then, when the TST instruction is executed, the N bit in the PSW is set because the PRS data appears as a negative value. With the N bit set, the BMI instruction causes the program to branch to the NOTAPE subroutine. This subroutine might, for example, type an error message on the teletypewriter.

EXERCISE 2

```
BIT #4, @#172460

BNE NOCARD
```

The contents of the CDST is logically ANDed with 4. If the hopper check bit is set prior to the BIT instruction being executed, the sum is 4 (not equal to zero) and the program branches to NOCARD.

Remember, the BIT instruction can be used to check more than one bit in a CSR.

EXERCISE 3

This subroutine uses the Teletype printer/punch to punch 64_{10} (100_8) characters on paper tape. The data is obtained from successive memory locations starting at address 5000. Each character is converted from ASCII to Teletype code before it is transferred to the data buffer.

```
DATA 1:    MOV #100, R0          ; Initialize byte count

ENCODE:    MOVB (R1)+, R2        ; Transfer data byte to GPR2

                                 ; Auto-increment source address

           ADD #200, R2          ; ASCII to Teletype code

                                 ; Conversion

WAIT:      BIT #200, @#177564    ; Wait until TPS ready bit

           BEQ WAIT              ; Is set

           MOVB R2, @#177566     ; Transfer character to TPB

                                 ; And punch

           DEC R0                ; Decrement byte count

           BNE ENCODE            ; Return to ENCODE and punch

                                 ; Another character until

                                 ; Byte count = 0

           RTS R5                ; Return to main program
```

EXERCISE 4

```
KDATA:   CLR COUNT           ; Clear character count
         MOV # INDATA, R1    ; Move the initial storage address to GPR1
WAITK:   BIT #200, TKS       ; Loop on WAITK until
                             ; A key is pressed and
         BEQ WAITK           ; DONE is set
         MOVB TKB, (R1)+     ; Transfer character to storage address
         INC COUNT           ; Increment character count
         CMP COUNT, #100     ; Compare character count
                             ; with maximum allowed
         BLT WAITK           ; Repeat subroutine if
                             ; fewer than 100 characters
                             ; stored
         RTS R5              ; Return to main program
                             ; Assume that the subroutine
                             ; is entered via the
                             ; instruction JSR R5, KDATA
```

In the compare (CMP) instruction the destination is subtracted from COUNT. Until COUNT = 100 the result is a negative number (less than zero). Therefore, until 100 characters have been stored, the BLT instruction conditions are met and the program loops on WAITK.

EXERCISE 5

```
TRANS:   CLR COUNT           ; Clear character count
         MOV # INDATA, R0    ; Move initial Teletype
                             ; code storage address to
                             ; GPR 0
         MOV # TDATA, R1     ; Move initial ASCII
                             ; code storage address to
                             ; GPR 1
REPET:   SUB #200, (R0)      ; Teletype to ASCII conversion
         MOVB (R0)+, (R1)+   ; Transfer data to new store
                             ; area. Auto-increment both addresses
         INC COUNT           ; Increment character count
         CMP #100, COUNT     ; Compare character count
                             ; with 100
         BGT REPET           ; branch to REPET until
                             ; count = 0
         RTS R5              ; Return to main program
```

EXERCISE 6

```
          CLR TALLY
          MOV # INDATA, R1
PAWS:     TSTB TKS
          BPL PAWS
          CMP TKB, #240
          BEQ OUTPUT
          MOVG TKB, (R1)+
          INC TALLY
          CMP TALLY, #100
          BLT PAWS
OUTPUT:   MOV #INDATA, R2
WAITPR:   BIT #200, TPS
          BEQ WAITPR
          MOVB (R2)+, TPB
          CMP R1, R2
          BGT WAITPR
          HALT
```

Care must be used in selecting the branch instruction that follows the CMP R1, R2 instruction. If, for example, only one character has been stored, then R1 and R2 would have the same contents after this single character was printed. At this point the program should halt, not branch back. Therefore, branching back must take place if the contents of R1 are greater than that of R2, and halt when they are equal. The BGT instruction provides the correct branching in this case. Most branching errors involve one loop too many or one too few.

EXERCISE 7

```
BIS #100, TKS        ; After setting bit 6 the
                     ; TKS = 100 if it was
                     ; previously clear
```

EXERCISE 8

```
BIC #2, ADCS
```

This instruction clears bit 1 without disturbing the other bits. The BIC instruction is also useful for masking out unwanted portions of a data word. For example BIC 40, XXXX would convert lower case ASCII characters a through z to upper case characters A through Z.

EXERCISE 9

```
CMP COUNT, TOPNUM
```

EXERCISE 10

a) The CPU background program is stopped and the card reader service routine is run.

b) The Teletype keyboard's request is accepted only after the card reader service routine is completed.

When all interrupt service routines have been completed, the CPU background program is resumed at the point where it was interrupted.

EXERCISE 11

a) The CPU program is terminated.

b) The keyboard service routine at Priority Level 4 is started.

c) The keyboard service routine is stopped before its completion.

d) The card reader service routine (Priority Level 6) is started and run to completion.

e) The keyboard service routine is resumed at the point where it was stopped and run to completion.

f) The CPU program is resumed at the point where it was stopped.

EXERCISE 12

The instruction RTI, Return From Interrupt, is *always* the last executed instruction of a service routine. It causes a return to the previously interrupted program.

EXERCISE 13

```
MOV R1,-(SP)

MOV R2,-(SP)

    .

    .

    .

MOV (SP)+, R2

MOV (SP)+, R1
```

EXERCISE 14

```
KEYSRV:  MOV R1, -(SP)

         MOV DASTOR, R1

             .

             .

             .

             .

         MOV (SP)+, R1

         RTI
```

EXERCISE 15

```
         MOV # MESAGE, TYPOUT     ; Background initialization

TITLE:   MOV R3, -(SP)            ; PUSH R3

         MOV TYPOUT, R3           ; Initialize R3 with

                                  ; address

         MOV (R3)+, TPB           ; Move data. Start printing.

         MOV R3, TYPOUT           ; Store new data address

         MOV (SP)+, R3            ; POP stack into R3

         RTI                      ; Return to previous program
```

The TITLE service routine is entered each time a character is to be printed. This happens each time the peripheral raises its DONE flag, indicating readiness to print another character.

EXERCISE 16

```
PROFIL:   MOV R5,-(SP)          ; Saves original R5 contents
          CLR INCHAR            ; Clear temporary storage location
          MOVB TKB, INCHAR      ; Transfer character to
                                ; temporary storage location
          SUB #300, INCHAR      ; INCHAR now contains an
                                ; offset of from 1 to 32₈
          MOV INCHAR, R5        ; R5 now contains the offset
          INCB OCCUR(R5)        ; Increment the count for the
                                ; character. Destination
                                ; address is determined by
                                ; adding the offset in R5
                                ; to the address OCCUR
          MOV (SP)+, R5         ; POP previous R5 data
          RTI                   ; Return to previous routine
```

TEST – I/O PROGRAMMING

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. A function of the data buffer register in an interface is to:

   a. Transfer control information to and from the peripheral.

   b. Convert parallel data from the device to a serial format for the Unibus.

   c. Hold data until it is completely assembled and ready for transfer to the Unibus or to the peripheral.

   d. Monitor the status of the peripheral.

2. Match the following CSR bits to their functions.

( )	Bit 0	a.	Allows an interrupt.
( )	Bits 1–3	b.	Indicates an error has occurred.
( )	Bit 6	c.	Selects any one of 8 devices.
( )	Bit 7	d.	Device is not available.
( )	Bits 8–10	e.	Starts device operation.
( )	Bit 11	f.	Device is done or ready.
( )	Bit 15	g.	Specifies one of 8 possible device functions.

3. Explain how the TSTB instruction is used with the CSR READY or DONE bit to control program operation.

read on ➧

4. Match each instruction with its corresponding function.

a.	TSTB @#177560	( )	Initiates a read operation in the Teletype.
b.	MOVB (R0)+, TPB	( )	Starts the Teletype's reader and allows an interrupt as soon as data is available in the buffer.
c.	MOV #776, R6	( )	Checks on availability of data in Teletype's reader buffer.
d.	MOV #340, @#62	( )	Determines when the Teletype printer/punch is ready to accept data.
e.	INCB @#177560	( )	Transfers data into the Teletype printer/punch buffer.
f.	MOV (SP)+, R1	( )	Initializes the stack pointer.
g.	MOVB #101, TKS	( )	Places the starting address of the Teletype's service routine into a vector location.
h.	RTS R5	( )	Sets up a PSW that will inhibit all device interrupts.
i.	MOV #2000, @#60	( )	Returns CPU to the main program.
j.	TSTB TPS	( )	Pops data from the hardware stack.

read on ➧

5. The program loop method of I/O programming, although wasteful of CPU time, is often used. Which of the following circumstances would call for its use:

   a. If the stack is overloaded.

   b. If program continuation is entirely dependent on the data the CPU is waiting for.

   c. If the peripheral operates at the same speed as the CPU.

   d. If the computer system contains more than one peripheral device.

6. When a JSR instruction is executed what address is placed in the PC?

   a. The starting address of the interrupt service routine.

   b. The address of the instruction that follows the JSR.

   c. The starting address of the subroutine specified in the destination portion of the JSR.

   d. The address contained in the register indicated in the JSR instruction.

7. Two loader programs were discussed in this study unit. The first program is deposited in memory through the console and is called the ____________ loader program. The second loader program is entered through the paper tape reader and is called the __________ loader program.

8. The starting address of the bootstrap loader program is xx7 744, where "xx" is a function of the memory size.

   a. What is the value "xx" for 8K of memory? ____________

   b. What is the value "xx" for 28K of memory? ____________

read on ➧

9. Listed below are the instructions that make up the bootstrap program. Briefly, explain the function of each instruction.

   a. START: MOV DEVICE,R1

   b. LOOP: MOV #.-LOAD+2,R2

   c. ENABLE: INC @R1

   d. WAIT: TSTB @R1

   e. BPL WAIT

   f. MOVB 2(R1), LOAD(R2)

read on ➧

g. INC LOOP+2

h. BRNCH: BR LOOP

10. One of the instructions in the bootstrap program moves an "address displacement" into general-purpose register R2. How is this address displacement used?

11. As the absolute loader program is stored in memory, part of it overlays the original bootstrap program. What is the purpose of this overlay?

read on ➧

12. The following steps occur when the absolute loader is deposited in memory. Place these steps in the correct order.

( ) The CPU stops executing the bootstrap loader and starts executing the absolute loader.

( ) The absolute loader program jumps to a halt instruction.

( ) The absolute loader program is read from the tape and is deposited in successive memory locations.

( ) The absolute loader paper tape is placed in the paper tape reader and the start switch is pressed.

( ) The absolute loader program is executed and the bootstrap program is restored to its original condition.

( ) A portion of the absolute loader is overlayed onto the bootstrap loader program, thus modifying the bootstrap program.

( ) The absolute loader program can now be used to bring in other programs stored on paper tape.

( ) The bootstrap program is toggled in through the console switch register.

( ) The leader code is read from the absolute loader tape and the CPU continues looping through the bootstrap program.

notes

notes