Appendix F

Selected Solutions

F.9 Chapter 9 Solutions

- 9.1 The most important advantage of doing I/O through a trap routine is the fact that it is not necessary for the programmer to know the gory low-level details of the specific hardware's input/output mechanism. These details include:
 - the hardware data registers for the input and output devices
 - the hardware status registers for the input and output devices
 - the asynchronous nature of the input relative to the executing program

Besides, these details may change from computer to computer. The programmer would have to know these details for the computer she's working on in order to be able to do input/output. Using a trap routine requires no hardware-specific knowledge on part of the programmer and saves time.

- 9.3 (a) Some external mechanism is the only way to start the clock (hence, the computer) after it is halted. The Halt service routine can never return after bit 15 of the machine control register is cleared because the clock has stopped, which means that instruction processing has stopped.
 - (b) STI R0, MCR This instruction clears the most significant bit of the machine control register, stopping the clock.
 - (c) LD R1, SaveR1
 - (d) The RET of the HALT routine will bring program control back to the program that executed the HALT instruction. The PC will point to the address following the HALT instruction.
- 9.5 Note: This problem should be corrected to read as follows:

.ORIG x3000 LEA RO, LABEL

```
STR R1, R0, #3
TRAP x22
TRAP x25
LABEL .STRINGZ "FUNKY"
LABEL2 .STRINGZ "HELLO WORLD"
.END
```

Answer: FUN

9.7 Note: This problem belongs in chapter 10.

The three errors that arose in the first student's program are:

- 1. The stack is left unbalanced.
- 2. The privilege mode and condition codes are not restored.
- 3. Since the value in R7 is used for the return address instead of the value that was saved on the stack, the program will most likely not return to the correct place.

9.9	(a)	ST ST	R1, R2,	SaveR1 SaveR2		
		AND	R0,	R0,	#0	;Zero out the ;return value
		LDI	R1,	MBUSY		;Load the ;contents of ;machine busy bit ;pattern into R1
		LD	R2,	MASK		;Load the mask, x00FF
		AND		R1,	R2	;Mask out bits <7:0>
		LD	R2,	-		•
		ADD	R1,	R1,	R2	
		BRnp	Return	·		;Branch if bit pattern
						;is not x00FF (some
						;machines busy)
		ADD	R0,	R0,	#1	;No machines are busy,
						;so return 1
	Return	LD	R1,	SaveR1		
		LD	R2,	SaveR2		
	C D 1	RET	0 0 0 0			
		.FILL	x0000			
		.FILL				
		.FILL				
		.FILL				
	NMASK	. F'ILL	x-00FF			
	(b)					
		ST	R1,	SaveR1		
		ST	R2,	SaveR2		

		AND	R0,	R0,	#	0 ;Zero out the	
		LDI	R1,	MBUS	Y	<pre>;return value ;Load r1 with the</pre>	
		201	111,	112001	-	; contents of the machine	
						;busy bit	
		LD	R2,	MASK		;Load the mask, x00FF	
		AND	R1,		R	2 ;Mask out bits <7:0>	
		BRNI	P Retu	rn		;Branch if bit	
						;pattern is not x0000	
		ADD	RO,	RO,	#	;(some machines not busy) 1 ;All are busy, so	
		ADD	ΝΟ,	1.0,	π	;return 1	
	Return	LD	R1,	Savel	R1		
		LD	R2,	Savel	R2		
		RET					
		.FIL					
			L x000				
	MBUSY		L x400				
	MASK	• []	L x00F	<u> </u>			
(c)		O.E.	D 1	C D 1			
		ST ST	R1, R2,	SaveR1 SaveR2			
			-	SaveR2 SaveR3			
		ST		SaveR4			
		AND	R0,	R0,	#0	;Zero out the	
						;return value	
		LDI	R1,	MBUSY		;Load R1 with the	
						;machine busy bit pattern	
		LD	R2,	MASK		;R2 will act as a mask	
		T D	D 2	COLINE		; to mask out the bit needed	
		LD	R3,	COUNT		;R3 will act as the ;iteration counter	
	Loop	AND	R4,	R1,	R2	;Mask off the bit to	
	Tool		,	111/		; check if machine is busy	
		BRp NotBusy				;Branch if machine	
						;is not busy	
		ADD	R0,	R0,	#1	;Increment number	
						; of busy machines	
	NotBusy	ADD	R2,	R2,	R2	;Left shift mask to the	
		ADD	R3,	R3,	#-1	<pre>;next bit to be checked ;Decrement</pre>	
		עטע	1/0,	1\0,	π-1	;iteration counter	
		BRp	Loop		;Br	;Branch if counter is not zero	
	Return	LD	R1,	SaveR1	•		
		LD	R2,	SaveR2			

MBUSY

.FILL x4001

```
R3,
                        SaveR3
           LD
                R4,
                        SaveR4
           LD
           RET
           .FILL x0000
   SaveR1
   SaveR2 .FILL x0000
           .FILL x0000
   SaveR3
   SaveR4 .FILL x0000
           .FILL x4001
  MBUSY
   MASK
           .FILL x0001
   COUNT
           .FILL #8
(d)
           ST
                R1,
                        SaveR1
           ST
                R2,
                        SaveR2
           ST
                R3,
                        SaveR3
           ST
                        SaveR4
                R4,
           AND
                               #0
                R0,
                        R0,
                                     ; Zero out the
                                      ;return value
           LDI
                R1,
                       MBUSY
                                      ;Load R1 with the machine
                                      ; busy bit pattern
           LD
                R2,
                       MASK
                                      ;R2 will act as a mask to
                                      ; mask out the bit needed
                                      ;R3 will act as the
           LD
                R3,
                       COUNT
                                      ;iteration counter
                                      ; Mask off the bit to check
                R4,
                        R1,
                               R2
   Loop
           AND
                                      ; if machine is busy
           BRz
                Busy
                                      ;Branch if machine
                                      ;is busy
           ADD
                               #1
                                      ; Increment number
                R0,
                        R0,
                                      ; of not
                                      ; busy machines
   Busy
                R2,
                        R2,
                               R2
                                      ;Left shift mask to the
           ADD
                                      ; next bit to be checked
           ADD
                R3,
                        R3,
                               \# - 1
                                      ; Decrement
                                      ;iteration counter
                Loop
                                      ;Branch if counter is not zero
           BRp
   Return
           LD
                R1,
                        SaveR1
           LD
                R2,
                        SaveR2
                        SaveR3
           LD
                R3,
           LD
                R4,
                        SaveR4
           RET
          .FILL x0000
   SaveR1
          .FILL x0000
   SaveR2
   SaveR3
          .FILL x0000
   SaveR4 .FILL x0000
```

```
MASK
         .FILL x0001
          .FILL #8
  COUNT
              R1,
(e)
          ST
                    SaveR1
          ST
              R2,
                     SaveR2
          ST
              R3,
                    SaveR3
          AND RO,
                    RO, #0
                                 ; Zero out the
                                  ;return value
  ADD R1,
                    #1
             R0,
  ADD R3,
             R5,
                    #0
  BRz Check
  LP1 ADD R1,
                R1, R1 ; Left-shift R1
  ADD R3, R3, \#-1
  BRnp LP1
          LDI R2,
                    MBUSY
                                 ;Load R2 with the machine
                                  ; busy bit pattern
  Check AND R1,
                     R1,
                            R2
          BRz NotBusy
                                   ;Branch if machine
                                  ;is busy
                            #1
          ADD RO, RO,
  NotBusy LD
              R1,
                     SaveR1
              R2,
                     SaveR2
          LD
          LD
                    SaveR3
              R3,
          RET
  SaveR1 .FILL x0000
  SaveR2 .FILL x0000
  SaveR3 .FILL x0000
  MBUSY
         .FILL x4001
(f); This code assumes that at least one machine is free
          ST
              R1, SaveR1
          ST
                     SaveR2
              R2,
          ST
              R3,
                     SaveR3
          ST
              R4,
                     SaveR4
          AND RO,
                     RO, #0
                                 ;Zero out the
                                  ;return value
                                  ;Load R1 with the machine
          LDI R1,
                     MBUSY
                                  ; busy bit pattern
          LD
              R2,
                     MASK
                                  ;R2 will act as a mask to
                                  ; mask out the bit needed
          LD
              R3,
                     COUNT
                                  ;R3 will act as the
                                  ;iteration counter
```

```
R2
                                     ; Mask off the bit to check
Loop
         AND
              R4,
                      R1,
                                     ; if machine is busy
         BRz
              Return
                                     ;Branch if machine is free
ADD
     R2,
             R2,
                     R2
                            ;Left shift mask to the
                                     ; next bit to be checked
ADD
     R0,
             R0,
                     #1
                              \# - 1
              R3,
                      R3,
         ADD
         BRp
              Loop
                                     ; Branch if counter is not zero
Return
         LD
              R1,
                      SaveR1
         LD
              R2,
                      SaveR2
         LD
              R3,
                      SaveR3
         LD
              R4,
                      SaveR4
         RET
         .FILL x0000
SaveR1
SaveR2
         .FILL x0000
SaveR3
         .FILL x0000
SaveR4
         .FILL x0000
MBUSY
         .FILL x4001
MASK
         .FILL x0001
COUNT
         .FILL #8
```

- 9.11 The label S_CHAR cannot be represented in 9-bit signed PC offset for the ST R0, S_CHAR and LEA R6, S_CHAR instructions. The range for a PCoffset9 instruction (such as LEA or ST) is only from -256 to 255 locations. Due to the number of locations that have been set aside for BUFFER, the location labeled S_CHAR falls oustide of this range for the ST and LEA instructions. This problem can be fixed by switching the lines BUFFER .BLKW 1001 and S_CHAR .FILL x0000.
- 9.13 The linkage for JSR A is destroyed when JSR B is executed.
- 9.15 (a) TRAP x72
 - (b) Yes, this routine will work, but whatever value was in R0 before TRAP x72 is executed will be overwritten during the subroutine.
- 9.17 (a) LD R3, NEGENTER
 - (b) STR R0, R1, #0
 - (c) ADD R1, R1, #1
 - (d) STR R2, R2, #0 STR R2, R1, #0;吧?
- 9.19 (a) LD R2, MASK8
 - (b) JSR HARDDISK
 - (c) BR END
 - (d) LD R2, MASK4
 - (e) JSR ETHERNET

- (f) BR END
- (g) LD R2, MASK2
- (h) JSR PRINTER
- (i) BR END
- (j) JSR CDROM
- (k) HALT