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.

9.2 (a) The trap vector is 8 bits wide. 256 trap routines can be implemented in the LC-3.

(b) After the TRAP routine is executed, program control must be passed back to the code that called the TRAP instruction. This is done by copying the value in R7 into the PC. The RET instruction provides this functionality. BRnzp does not restore the PC.

(c) One.

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.

9.4 (a) 1111000000100001 (xf021)

(b) x0430

(c) x0437

(d) HookemHorns

9.5 Note: This problem should be corrected to read as follows:

```
.ORIG  x3000
LEA    R0, LABEL
STR    R1, R0, #3
TRAP   x22
TRAP   x25
```

```
LABEL         .STRINGZ "FUNKY"
LABEL2        .STRINGZ "HELLO WORLD"
              .END
```

Answer: FUN

9.6
```
              .ORIG x3000
              LD  R2, LOWER    ; Load -A
              LD  R3, ASCII        ; Load ASCII difference
              LD  R4, UPPER     ; Load -Z
AGAIN   TRAP  x23              ; Request keyboard input
              ADD  R1, R2, R0
              BRn  EXIT
              ADD  R1, R4, R0
              BRp  EXIT
              ADD  R0, R0, R3    ; Change to lowercase
              TRAP  x21             ; Output to monitor
              BRnzp  AGAIN       ; ... and do it again!!

EXIT           TRAP  x25             ; Halt
LOWER       .FILL  xFFBF          ; FFBF = -A
UPPER       .FILL  xFFA6          ; FFA6 = -Z
ASCII       .FILL  x0020
```

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.

9.8 If the value in A is a prime number, 1 is stored in memory location RESULT; otherwise, 0 is stored in RESULT.

9.9 (a)
```
              ST    R1,    SaveR1
              ST    R2,    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,    R1,    R2  ;Mask out bits <7:0>
              LD    R2,    NMASK
```

```
            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
            RET
SaveR1      .FILL  x0000
SaveR2      .FILL  x0000
MBUSY       .FILL  x4001
MASK        .FILL  x00FF
NMASK       .FILL  x-00FF
```

(b)

```
            ST    R1,    SaveR1
            ST    R2,    SaveR2
            AND   R0,    R0,    #0   ;Zero out the
                                     ;return value
            LDI   R1,    MBUSY       ;Load r1 with the
                                     ;contents of the machine
                                     ;busy bit
            LD    R2,    MASK        ;Load the mask, x00FF
            AND   R1,    R1,    R2   ;Mask out bits <7:0>
            BRNP  Return             ;Branch if bit
                                     ;pattern is not x0000
                                     ;(some machines not busy)
            ADD   R0,    R0,    #1   ;All are busy, so
                                     ;return 1
Return      LD    R1,    SaveR1
            LD    R2,    SaveR2
            RET
SaveR1      .FILL  x0000
SaveR2      .FILL  x0000
MBUSY       .FILL  x4001
MASK        .FILL  x00FF
```

(c)

```
            ST   R1,    SaveR1
            ST   R2,    SaveR2
            ST   R3,    SaveR3
            ST   R4,    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
                                    ;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
                                     ;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
                                     ;next bit to be checked
            ADD  R3,    R3,    #-1   ;Decrement
                                     ;iteration counter
            BRp  Loop           ;Branch if counter is not zero
Return      LD   R1,    SaveR1
            LD   R2,    SaveR2
            LD   R3,    SaveR3
            LD   R4,    SaveR4
            RET
SaveR1      .FILL x0000
SaveR2      .FILL x0000
SaveR3      .FILL x0000
SaveR4      .FILL x0000
MBUSY       .FILL x4001
MASK        .FILL x0001
COUNT       .FILL #8
```

(d)

```
            ST   R1,    SaveR1
            ST   R2,    SaveR2
            ST   R3,    SaveR3
            ST   R4,    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 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 check
                                     ;if machine is busy
            BRz  Busy                ;Branch if machine
                                     ;is busy
            ADD  R0,    R0,    #1    ;Increment number
```

```
                                        ;of not
                                        ;busy machines
Busy    ADD  R2,    R2,    R2    ;Left shift mask to the
                                  ;next bit to be checked
        ADD  R3,    R3,    #-1   ;Decrement
                                  ;iteration counter
        BRp  Loop                ;Branch if counter is not zero
Return  LD   R1,    SaveR1
        LD   R2,    SaveR2
        LD   R3,    SaveR3
        LD   R4,    SaveR4
        RET
SaveR1  .FILL x0000
SaveR2  .FILL x0000
SaveR3  .FILL x0000
SaveR4  .FILL x0000
MBUSY   .FILL x4001
MASK    .FILL x0001
COUNT   .FILL #8
```

(e)

```
        ST   R1,    SaveR1
        ST   R2,    SaveR2
        ST   R3,    SaveR3

        AND  R0,    R0,    #0    ;Zero out the
                                  ;return value

ADD  R1,    R0,    #1
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
        ADD  R0,    R0,    #1
NotBusy LD   R1,    SaveR1
        LD   R2,    SaveR2
        LD   R3,    SaveR3
        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   R2,    SaveR2
        ST   R3,    SaveR3
        ST   R4,    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 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 check
                                 ;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
        ADD  R3,    R3,    #-1
        BRp  Loop                ;Branch if counter is not zero

Return  LD   R1,    SaveR1
        LD   R2,    SaveR2
        LD   R3,    SaveR3
        LD   R4,    SaveR4
        RET
SaveR1  .FILL x0000
SaveR2  .FILL x0000
SaveR3  .FILL x0000
SaveR4  .FILL x0000
MBUSY   .FILL x4001
MASK    .FILL x0001
COUNT   .FILL #8
```

9.10 Since the LC-3 ISA allows for an 8-bit trap vector, 256 service routines can be created using the current semantics of the LC-3 ISA. However, if the address specified by the TRAP instruction contained the first instruction in the service routine, the number of possible service routines would be greatly reduced. If each service routine required 16 locations, then

the number of possible service routines would only be 16 (256/16=16). The semantics of the TRAP instruction could be modified as follows: Change the trap vector to 4 bits (instead of 8); zero-extend the trap vector and shift it to the left by 4 to get the starting address of the service routine.

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.

9.12 The final values at DATA are the sorted version of the initial values at DATA in ascending order.

9.13 The linkage for JSR A is destroyed when JSR B is executed.

9.14 If the RUN latch is later set (manually), the service routine will restore the values in R0,R1, and R7 and return to the calling program. This use of the TRAP x25 instruction can be a useful tool in troubleshooting and debugging.

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.

9.16 Error 1: The line VALUE .FILL X30000 will generate an assembly error because 0x30000 does not fit in one LC-3 memory location.

only one error in current problem statement

9.17 (a) LD R3, NEGENTER
(b) STR R0, R1, #0
(c) ADD R1, R1, #1
(d) STR R2, R2, #0

9.18 (a) ADD R1, R1, #1
(b) TRAP x25
(c) ADD R0, R0, #5
(d) BRzp K

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