Appendix F

Selected Solutions

F.7 Chapter 7 Solutions

7.1 0xA7FE

7.3 Using an instruction as a label confuses the assembler because it treats the label as the opcode itself so the label AND will not be entered into the symbol table. Instead the assembler will give an error in the second pass.

7.5 (a) The program calculates the product of values at addresses M0 and M1. The product is stored at address RESULT.

mem[RESULT] = mem[M0] * mem[M1]

(b) x200C

7.7 The assembly language program is:

```
           .ORIG   x3000
           AND     R5, R5, #0
           ADD     R5, R5, #1 ;R5 will act as a mask to
                              ;mask out the unneeded bit
           AND     R1, R1, #0 ;zero out the result register
           AND     R2, R2, #0 ;R2 will act as a counter
           LD      R3, NegSixt
MskLoop    AND     R4, R0, R5 ;mask off the bit
           BRz     NotOne     ;if bit is zero then don't
                              ;increment the result
           ADD     R1, R1, #1 ;if bit is one increment
                              ;the result
NotOne     ADD     R5, R5, R5 ;shift the mask one bit left
           ADD     R2, R2, #1 ;increment counter (tells us
                              ;where we are in bit pattern)
           ADD      R6, R2, R3
           BRn      MskLoop    ;not done yet go back and
                               ;check other bits
           HALT
NegSixt    .FILL    #-16
           .END
```

7.9 The .END pseudo-op tells the assembler where the program ends. Any string that occurs after that will be disregarded and not processed by the assembler. It is different from HALT instruction in very fundamental aspects:

1. It is not an instruction, it can never be executed.
2. Therefore it does not stop the machine.
3. It is just a marker that helps the assembler to know where to stop assembling.

7.11

```
        ; Prog 7.11
        ; This code does not perform error checking
        ; It accepts 3 characters as input
        ; The first one is either x or #
        ; The next two is the number.

        .ORIG   x3000
        IN                 ; input the first char - either x or #
        AND     R3, R3, #0
        ADD     R3, R3, #9 ; R3 = 9 if we are working
                   ; with a decimal or 16 if hex
        LD      R4, NASCIID
        LD      R5, NHEXDIF

        LD      R1, NCONSD
        ADD     R1, R1, R0
        BRz     GETNUMS
        LD      R1, NCONSX
        ADD     R1, R1, R0
        BRnp    FAIL
        ADD     R3, R3, #6   ; R3 = 15

GETNUMS IN
        ST      R0, CHAR1
        IN
        ST      R0, CHAR2
        LEA     R6, CHAR1
        AND     R2, R2, #0
        ADD     R2, R2, #2   ; Loop twice
; Using R2, R3, R4, R5, R6 here
        AND     R0, R0, #0   ; Result
```

```
LOOP    ADD     R1, R3, #0
        ADD     R7, R0, #0
LPCUR   ADD     R0, R0, R7
        ADD     R1, R1, #-1
        BRp     LPCUR

        LDR     R1, R6, #0
        ADD     R1, R1, R4

        ADD     R0, R0, R1

        ADD     R1, R1, R5
        BRn     DONECUR
        ADD     R0, R0, #-7   ; for hex numbers
DONECUR
        ADD     R6, R6, #1
        ADD     R2, R2, #-1
        BRp     LOOP

        ; R0 has number at this point

        AND     R2, R2, #0
        ADD     R2, R2, #8

        LEA     R3, RESEND
        LD      R4, ASCNUM
        AND     R5, R5, #0
        ADD     R5, R5, #1

STLP    AND     R1, R0, R5
        BRp     ONENUM
        ADD     R1, R4, #0
        BRnzp   STORCH
ONENUM  ADD     R1, R4, #1
STORCH  ADD     R5, R5, R5
        STR     R1, R3, #-1
        ADD     R3, R3, #-1
        ADD     R2, R2, #-1
        BRp     STLP
        LEA     R0, RES
        PUTS
FAIL    HALT
CHAR1   .FILL   x0
CHAR2   .FILL   x0
```

```
ASCNUM  .FILL   x30
NHEXDIF .FILL   xFFEF   ; -x11
NASCIID .FILL   xFFD0   ; -x30
NCONSX  .FILL   xFF88   ; -x78
NCONSD  .FILL   xFFDD   ; -x23

RES     .BLKW 8
RESEND  .FILL x0
        .END
```

7.13 Error 1:
Line 8: ST R1, SUM
SUM is an undefined label. This error will be detected at assembly time.

Error 2:
Line 3: ADD R1, R1, R0
R1 was not initialized before it was used; therefore, the result of this ADD instruction may not be correct. This error will be detected at run time.

7.15 This program doubles all the positive numbers and leaves the negative numbers unchanged.

7.17 There is not a problem in using the same label in separate modules assuming the programmer expected the label to refer to different addresses, one within each module. This is not a problem because each module has its own symbol table associated with it. It is an error on the otherhand if the programmer expected each label AGAIN to refer to the same address.

7.19 The instruction labeled LOOP executes 4 times.

7.21 Correction: Please use the following LC-3 assembly language program for this problem:

```
        .ORIG x3000
        AND     R0, R0, #0
        ADD     R2, R0, #10
        LD      R1, MASK
        LD      R3, PTR1
LOOP    LDR     R4, R3, #0
        AND     R4, R4, R1
        BRz     NEXT
        ADD     R0, R0, #1
NEXT    ADD     R3, R3, #1
        ADD     R2, R2, #-1
        BRp     LOOP
        STI     R0, PTR2
        HALT
MASK    .FILL   x8000
PTR1    .FILL   x4000
PTR2    .FILL   x5000
```

Solution:
The assembled program:

```
0101 0000 0010 0000 ( AND R0, R0, #0 )
0001 0100 0010 1010 ( ADD R2, R0, #10 )
0010 0010 0000 1010 ( LD R1, MASK )
0010 0110 0000 1010 ( LD R3, PTR1 )
0110 1000 1100 0000 ( LDR R4, R3, #0 )
0101 1001 0000 0001 ( AND R4, R4, R1 )
0000 0100 0000 0001 ( BRz NEXT )
0001 0000 0010 0001 ( ADD R0, R0, #1 )
0001 0110 1110 0001 ( ADD R3, R3, #1 )
0001 0100 1011 1111 ( ADD R2, R2, #-1 )
0000 0011 1111 1001 ( BRp LOOP )
1011 0000 0000 0011 ( STI R0, PTR2 )
1111 0000 0010 0101 ( HALT )
1000 0000 0000 0000
0100 0000 0000 0000
0101 0000 0000 0000
```

This program counts the number of negative values in memory locations 0x4000 - 0x4009 and stores the result in memory location 0x5000.

7.23 (a) ADD R1, R1, #-1
(b) LDR R4, R1, #0
(c) ADD R0, R0, #1
(d) ADD R1, R1, #-1
(e) BR LOOP

7.25 This is an assembler error. The number 0xFF004 does not fit in one LC-3 memory location and therefore this .FILL cannot be assembled.