process) would use the symbol table entry for STARTofFILE in another module to complete the translation of our revised line 2E.

In this way, the .EXTERNAL pseudo-op allows references by one module to symbolic locations in another module without a problem. The proper translations are resolved by the linker.

Exercises

7.1 An assembly language program contains the following two instructions. The assembler puts the translated version of the LDI instruction that follows into location x3025 of the object module. After assembly is complete, what is in location x3025?

```
PLACE   .FILL   x45A7
        LDI     R3, PLACE
```

7.2 An LC-3 assembly language program contains the instruction:

```
ASCII   LD  R1, ASCII
```

The symbol table entry for ASCII is x4F08. If this instruction is executed during the running of the program, what will be contained in R1 immediately after the instruction is executed?

7.3 What is the problem with using the string AND as a label?

7.4 Create the symbol table entries generated by the assembler when translating the following routine into machine code:

```
        .ORIG   x301C
        ST      R3, SAVE3
        ST      R2, SAVE2
        AND     R2, R2, #0
TEST    IN
        BRz     TEST
        ADD     R1, R0, #-10
        BRn     FINISH
        ADD     R1, R0, #-15
        NOT     R1, R1
        BRn     FINISH
        HALT
FINISH  ADD     R2, R2, #1
        HALT
SAVE3   .FILL   X0000
SAVE2   .FILL   X0000
        .END
```

7.5 *a.* What does the following program do?

```
        .ORIG   x3000
        LD      R2, ZERO
        LD      R0, M0
        LD      R1, M1
LOOP    BRz     DONE
        ADD     R2, R2, R0
        ADD     R1, R1, -1
        BR      LOOP
DONE    ST      R2, RESULT
        HALT
RESULT  .FILL   x0000
ZERO    .FILL   x0000
M0      .FILL   x0004
M1      .FILL   x0803
        .END
```

b. What value will be contained in RESULT after the program runs to completion?

7.6 Our assembler has crashed and we need your help! Create a symbol table and assemble the instructions at labels D, E, and F for the program below. You may assume another module deposits a positive value into A before this module executes.

```
        .ORIG   x3000
        AND     R0, R0, #0
D       LD      R1, A
        AND     R2, R1, #1
        BRp     B
E       ADD     R1, R1, #-1
B       ADD     R0, R0, R1
        ADD     R1, R1, #-2
F       BRp     B
        ST      R0, C
        TRAP    x25
A       .BLKW   1
C       .BLKW   1
        .END
```

In no more than 15 words, what does the above program do?

7.7 Write an LC-3 assembly language program that counts the number of 1s in the value stored in R0 and stores the result into R1. For example, if R0 contains 0001001101110000, then after the program executes, the result stored in R1 would be 0000 0000 0000 0110.

7.8 An engineer is in the process of debugging a program she has written. She is looking at the following segment of the program, and decides to place a breakpoint in memory at location 0xA404. Starting with the PC = 0xA400, she initializes all the registers to zero and runs the program until the breakpoint is encountered.

```
Code Segment:
...
0xA400  THIS1  LEA    R0, THIS1
0xA401  THIS2  LD     R1, THIS2
0xA402  THIS3  LDI    R2, THIS5
0xA403  THIS4  LDR    R3, R0, #2
0xA404  THIS5  .FILL  xA400
...
```

Show the contents of the register file (in hexadecimal) when the breakpoint is encountered.

7.9 What is the purpose of the .END pseudo-op? How does it differ from the HALT instruction?

7.10 The following program fragment has an error in it. Identify the error and explain how to fix it.

```
        ADD     R3, R3, #30
        ST      R3, A
        HALT
A       .FILL   #0
```

Will this error be detected when this code is assembled or when this code is run on the LC-3?

7.11 The LC-3 assembler must be able to convert constants represented in ASCII into their appropriate binary values. For instance, x2A translates into 00101010 and #12 translates into 00001100. Write an LC-3 assembly language program that reads a decimal or hexadecimal constant from the keyboard (i.e., it is preceded by a # character signifying it is a decimal, or x signifying it is hex) and prints out the binary representation. Assume the constants can be expressed with no more than two decimal or hex digits.

7.12 What does the following LC-3 program do?

```
        .ORIG   x3000
        AND     R5, R5, #0
        AND     R3, R3, #0
        ADD     R3, R3, #8
        LDI     R1, A
        ADD     R2, R1, #0
AG      ADD     R2, R2, R2
        ADD     R3, R3, #-1
        BRnp    AG
        LD      R4, B
        AND     R1, R1, R4
        NOT     R1, R1
        ADD     R1, R1, #1
        ADD     R2, R2, R1
        BRnp    NO
        ADD     R5, R5, #1
NO      HALT
B       .FILL   xFF00
A       .FILL   x4000
        .END
```

7.13 The following program adds the values stored in memory locations A, B, and C, and stores the result into memory. There are two errors in the code. For each, describe the error and indicate whether it will be detected at assembly time or at run time.

```
Line No.
1                .ORIG  x3000
2       ONE      LD  R0, A
3                ADD R1, R1, R0
4       TWO      LD  R0, B
5                ADD R1, R1, R0
6       THREE    LD  R0, C
7                ADD R1, R1, R0
8                ST  R1, SUM
9                TRAP  x25
10      A        .FILL x0001
11      B        .FILL x0002
12      C        .FILL x0003
13      D        .FILL x0004
14               .END
```

7.14 *a.* Assemble the following program:

```
        .ORIG   x3000
        STI     R0, LABEL
        OUT
        HALT
LABEL   .STRINGZ "%"
        .END
```

b. The programmer intended the program to output a % to the monitor, and then halt. Unfortunately, the programmer got confused about the semantics of each of the opcodes (that is, exactly what function is carried out by the LC-3 in response to each opcode). Replace exactly **one** opcode in this program with the correct opcode to make the program work as intended.

c. The original program from part *a* was executed. However, execution exhibited some very strange behavior. The strange behavior was in part due to the programming error, and in part due to the fact that the value in R0 when the program started executing was x3000. Explain what the strange behavior was and why the program behaved that way.

7.15 The following is an LC-3 program that performs a function. Assume a sequence of integers is stored in consecutive memory locations, one integer per memory location, starting at the location x4000. The sequence terminates with the value x0000. What does the following program do?

```
        .ORIG   x3000
        LD      R0, NUMBERS
        LD      R2, MASK
LOOP    LDR     R1, R0, #0
        BRz     DONE
        AND     R5, R1, R2
        BRz     L1
        BRnzp   NEXT
L1      ADD     R1, R1, R1
        STR     R1, R0, #0
NEXT    ADD     R0, R0, #1
        BRnzp   LOOP
DONE    HALT
NUMBERS .FILL   x4000
MASK    .FILL   x8000
.END
```

7.16 Assume a sequence of nonnegative integers is stored in consecutive memory locations, one integer per memory location, starting at location x4000. Each integer has a value between 0 and 30,000 (decimal). The sequence terminates with the value −1 (i.e., xFFFF).

What does the following program do?

```
        .ORIG   x3000
        AND     R4, R4, #0
        AND     R3, R3, #0
        LD      R0, NUMBERS
LOOP    LDR     R1, R0, #0
        NOT     R2, R1
        BRz     DONE
        AND     R2, R1, #1
        BRz     L1
        ADD     R4, R4, #1
        BRnzp   NEXT
L1      ADD     R3, R3, #1
NEXT    ADD     R0, R0, #1
        BRnzp   LOOP
DONE    TRAP    x25
NUMBERS .FILL   x4000
        .END
```

7.17 Suppose you write two separate assembly language modules that you expect to be combined by the linker. Each module uses the label AGAIN, and neither module contains the pseudo-op `.EXTERNAL AGAIN`. Is there a problem using the label `AGAIN` in both modules? Why or why not?

7.18 The following LC-3 program compares two character strings of the same length. The source strings are in the `.STRINGZ` form. The first string starts at memory location x4000, and the second string starts at memory location x4100. If the strings are the same, the program terminates with the value 0 in R5. Insert instructions at (a), (b), and (c) that will complete the program.

```
        .ORIG   x3000
        LD      R1, FIRST
        LD      R2, SECOND
        AND     R0, R0, #0
LOOP    -------------- (a)
        LDR     R4, R2, #0
        BRz     NEXT
        ADD     R1, R1, #1
        ADD     R2, R2, #1
        -------------- (b)
        -------------- (c)
        ADD     R3, R3, R4
        BRz     LOOP
        AND     R5, R5, #0
        BRnzp   DONE
NEXT    AND     R5, R5, #0
        ADD     R5, R5, #1
DONE    TRAP    x25
FIRST   .FILL   x4000
SECOND  .FILL   x4100
        .END
```

7.19 When the following LC-3 program is executed, how many times will the instruction at the memory address labeled LOOP execute?

```
        .ORIG   x3005
        LEA     R2, DATA
        LDR     R4, R2, #0
LOOP    ADD     R4, R4, #-3
        BRzp    LOOPk                TRAP    x25
DATA    .FILL   x000B
        .END
```

7.20 LC-3 assembly language modules (a) and (b) have been written by different programmers to store x0015 into memory location x4000. What is fundamentally different about their approaches?

a.

```
        .ORIG   x5000
        AND     R0, R0, #0
        AND     R0, R0, #15
        STI     R0, PTR
        HALT
PTR     .FILL   x4000
        .END
```

b.

```
        .ORIG   x4000
        .FILL   x0015
        .END
```

7.21 Assemble the following LC-3 assembly language program.

```
        .ORIG   x3000
        AND     R0, R0, #0
        LD      R1, MASK
        ADD     R2, R0, #10
        LD      R3, PTR1
LOOP    LDR     R4, R3, #0
        AND     R4, R4, R1
        BRzp    NEXT
        ADD     R0, R0, #1
NEXT    ADD     R2, R2, #-1
        BRp     LOOP
        STI     R0, PTR2
MASK    .FILL   x8001
PTR1    .FILL   x4000
PTR2    .FILL   x5000
        .END
```

What does the program do (in no more than 20 words)?

7.22 The LC-3 assembler must be able to map an instruction's mnemonic opcode into its binary opcode. For instance, given an ADD, it must generate the binary pattern 0001. Write an LC-3 assembly language program that prompts the user to type in an LC-3 assembly language opcode and then displays its binary opcode. If the assembly language opcode is invalid, it displays an error message.