ECE 120: Introduction to Computing Spring 2019 ZJUI

due at start of section on Friday 24 May 2019

Homework 13: Assembling and Debugging LC-3

1. **Protecting Your Bits!**

   The code below is intended to load a value into R3, then multiply the value by 12. Add instructions and/or data **at the locations indicated by comments** to prevent the code from changing R0 during the multiplication step. ** For credit, do not modify the current instructions, nor insert code/data other than in those locations indicated. Also, if you insert more than two instructions at any insertion point, your will receive no credit. **

```
          .ORIG x3000

          LEA R0,DSTR
          LD R3,STARTV
          ST R3,DSTR

          ; You may add code here.

          ADD R0,R3,R3 ; x12
          ADD R3,R0,R3
          ADD R3,R3,R3
          ADD R3,R3,R3

          ; You may add code here.

          PUTS
          ; R3 value is correct here (HALT may change it
          ; in the simulator)
          HALT
DSTR      .STRINGZ "I know how to fix it!"
STARTV    .FILL #1000

          ; You may add data here.

          .END
```

2. **Mimicking an Assembler**

   Write symbol tables for each of the codes below. Your symbol table should be similar in nature to that produced by the LC-3 assembler: for each label that appears in the code, your table should list the label and associate the label with an address in LC-3 memory. For an example, see P&P Section 7.3.3, pp. 186-187, or simply generate one on your own with **`lc3as`** (the output ending in **`.sym`** is a symbol table file).

   a. The program given for problem 7.5.a in Patt and Patel.
   b. The program given for problem 7.16 in Patt and Patel.
   c. The program given in Homework 12 Problem 5 (**before your modifications**).
   d. The program given in problem 2 above (**before your modifications**).

3. When Assemblers Find Errors

The code below is supposed to read a sequence of ASCII characters and print them to the monitor. The characters are stored in a sequential series of memory addresses, and the last such address contains a 0 (which should not be printed). However, the ASCII characters are stored in the low 8 bits of each location, which the high 8 bits are used for some other purpose (not known to you, and not relevant to this problem).

Prof. Lumetta has produced the following code to accomplish the stated task. Unfortunately, the code has a bug. Fortunately, the assembler will find the bug.

Your tasks are as follows:

a. Identify the bug and explain it,
b. State in which pass (first or second) the assembler identifies the bug, and
c. Explain how to fix the bug (**in words, not in code**).

```
        .ORIG x3000

        LEA R1,STUFF
LOOP    LDR R0,R1,#0
        AND R0,R0,xFF
        BRz DONE
        OUT
        ADD R1,R1,#1
        BRnzp LOOP
DONE    HALT
STUFF   .FILL x7768
        .FILL xAB65
        .FILL xEA6C
        .FILL xF06C
        .FILL x976F
        .FILL x1200

        .END
```