CS/ECE 252 Introduction to Computer Engineering
Spring 2015 Section 2
Instructor: Mark D. Hill
TAs: Lisa Ossian and Sujith Surendran
Course URL: http://www.cs.wisc.edu/~markhill/cs252/Spring2015/

Homework 6 [Due at lecture on Wed, Apr 8]

Primary contact for this homework on Piazza is Sujith Surendran

Question 1 (6 Points)

Load the below LC-3 program in PennSim, and answer the following questions:

Address	Memory Content	Comment
x3000	0010 0100 0000 1010	R2 = 0
x3001	0010 0000 0000 1010	R0 = 3
x3002	0010 0010 0000 1010	R1 = 4
x3003	0000 0100 0000 0011	BRz x3007
x3004	0001 0100 0000 0010	R2 = R2 + R0
x3005	0001 0010 0111 1111	R1 = R1 - 1
x3006	0000 1111 1111 1100	BR x3003
x3007	0010 0110 0000 0010	Load x4000 into R3
x3008	0111 0100 1100 0000	Store 12 (xC) at x4000
x3009	1111 0000 0010 0101	HALT
x300A	0100 0000 0000 0000	Data value x4000
x300B	0000 0000 0000 0000	Data value x0
x300C	0000 0000 0000 0011	Data value x3
x300D	0000 0000 0000 0100	Data value x4

a) (3 points) Fill in the comments column with a summary of what each instruction does.

b) (1 point) How many times does the instruction at address x3003 execute?

5

d) (2 points) Describe what this program does in 1-2 sentences (ie, specify how the value at x4000 at the end of the execution relates to the Data Values at x300C and x300D)

Ans => Mem[x4000] = Mem [x300C] * Mem[x300D]

Question 2 (8 Points)

The following LC-3 program compares the numbers in memory locations x4000 and x4001. It then puts the larger number among the two into location x4002. Fill in the missing instructions of the code.

Suggestion: Verify your solution by executing it in PennSim.

```
  0011 0000 0000 0000     ; Program starts at x3000
(a) 1010 001 000001001    ; Load value at x4000 into R1
(b) 1010 010 000001001    ; Load value at x4001 into R2
(c) 1001 011 010 111111   ; R3 = NOT(R2)
(d) 0001 011 011 1 00001  ; R3 = R3 + 1 (Now R3 = - Mem[4001])
(e) 0001 100 001 0 00011  ; R4 = R1 + R3 (ie, R4 = Mem[4000] - Mem [4001])
(f) 0000 100 000000010    ; Branch if negative flag is set to STORE_R2
(g) 1011 001 000000101    ;  Store the value of R1 to x4002
(h) 0000 111 000000001    ;  Branch to HALT
(i) 1011 010 000000011    ;  STORE_R2 : Store the value of R2 to x4002
  1111 0000 0010 0101 ; HALT
  0100 0000 0000 0000 ; DATA1: x4000
  0100 0000 0000 0001 ; DATA2: x4001
  0100 0000 0000 0010 ; DATA3: x4002
```

Question 3 (6 Points)

Write a small LC-3 binary code which counts the sum of consecutive integers starting from 1 up to the number stored in memory location x4000. It then stores this number in x4001. Your code should start at x3000

For example, if the number stored at x4000= 5, then after running your program, the number at x4001 should be 1+2+3+4+5 = 15.

Assembly:

```
        .ORIG x3000
        LDI R1, x3009
        AND R0, R0, #0
        AND R2, R2, #0

LOOP:   AND R0, R0, #1
        ADD R2, R2, R0
        ADD R1, R1, #-1
        BRp LOOP

        STI R2, x300A
        HALT
```

Binary code:

```
0011 0000 0000 0000
1010 001 000001000
0101 000 000 1 00000
0101 010 010 1 00000
0001 000 000 1 00001
0001 010 010 0 00 000
0001001001111111
0000 001 111111100
1011 0010 0000 0010
1111 0000 0010 0101
0100 0000 0000 0000
0100 0000 0000 0001
```

Problem 4 (10 points)

(4 Points) Consider an algorithm which counts the number of times the string 'cs' occurs in a string that is stored starting at location x5000. The count should be stored at the location x4000.

For example, if the string at location x5000 is “abcs53cs23c4s3cs”, the count at x4000 should be 3.

Show the algorithm as a flowchart by decomposing it into its basic constructs.

Note: The first character of the string is stored at memory location 0x5000. The last character of the string is the NULL character (having an ASCII value 0x00). One character is stored in each memory location (the lower order 8 bits represent the ASCII character and higher order 8 bits are 0).

(6 Points) Convert the above algorithm to an LC-3 program. Submit the **binary code** as a **.txt file** to the dropbox. Print out a screenshot of your code running in PennSim. Your screenshot should show the final value in x4000 and x4001 before the program executes the HALT instruction (keep a breakpoint at HALT, and take a screenshot when the program hits the breakpoint). **Turn in the screenshot as a hard copy.** Also mention on the screenshot which string you have stored starting at x5000 while testing your code.

Assembly:

```
            .ORIG x3000
            LD R0, START_LOC
            LD R1, INDEX
            LD R4, NEG_C
            LD R6, COUNT
            LD R7, NEG_S

CHECK_FOR_C    ADD R1, R1, #1     ;Increment the index.
            ADD R2, R1, R0        ;R2 has the address of next data
            LDR R3, R2, #0        ;R3 has the data
            BRz END_OF_DATA       ;If there is no more data, leave program.
            ADD R5, R3, R4        ;If data is C, R5 will have value 0
            BRnp CHECK_FOR_C

CHECK_FOR_S    ADD R2, R2, #1     ;Point to the next address
            LDR R3, R2, #0        ;Check if R3 is s
            BRz END_OF_DATA
            ADD R5, R3, R7        ;If data is s, R5 will have value 0
            BRnp CHECK_FOR_C
            ADD R6, R6, #1
            BRnzp CHECK_FOR_S

END_OF_DATA    STI R6, RESULT_LOCATION

             BRz END

END          HALT

START_LOC          .FILL x5000
INDEX              .FILL #-1
NEG_C              .FILL  #-99
COUNT              .FILL x0
NEG_S              .FILL #-115
RESULT_LOCATION .FILL x4000
```