CS/ECE 252 Introduction to Computer Engineering

Spring 2017
Instructor: Radhakrishnan Rahul Nayar
TAs: Annie Lin, Mohit Verma
URL:http://pages.cs.wisc.edu/~rrahulnayar/cs252/Spring2017/index.html

Homework 7 [Due at lecture on Friday, April 14]

Primary contact for this homework: Annie Lin [elin23 at wisc dot edu]
You must do this homework **alone**.
Please staple your assignments in the top left-hand corner or you will receive a 2 point deduction.

Important Notes:

a. Problems 4 and 5 ask you to submit your assembly code as an **assembly file (*.asm**) to the Learn@UW dropbox.
b. It is important to test your code before submitting it, or you may end up losing points even after your hard work.

Submission guidelines:

a. For your your code submit only **one** archive file (*.zip) to the folder **homework7**.
b. Name the file with the following convention: NetID_hw7.zip. For example: If your NetID is bbadger10, then the file should be named **bbadger10_hw7.zip** .
c. Your archive file should contain the following (**The files MUST be named exactly like this**):
    a. hw7_p4.asm - Assembly code for problem 4
    b. hw7_p5.asm - Assembly code for problem 5

It is important that you follow the above submission guidelines since your code submission will be graded automatically. Any submission that deviates from the above guidelines will be penalized.

You can submit your code for problem 4 and 5 as many times as you want until the beginning of lecture (i.e., 9:55 AM) on Friday, April 14. After that time we will consider your latest submissions for grading.

Problem 1 (3 points)

(a) There are three assembly **syntax** errors in the program below. Find and correct the errors.

```
      .START x3000
      LD R4, CHECK
      LD R0, VALUE
      AND R3, R3, #0
LOOP  AND R2, R0, R4
      BRz SHIFT
      ADD R3, R3, #1
SHIFT ADD R4, R4, R4
      BRnp LOOP
      HALT
```

```
VAL   .FILL x4023
CHECK .FILL #1
      .STOP
```

b) Run the corrected program in PennSim. What does it do? What is the final value stored in R3 after execution ends?

Problem 2 (5 points)

```
      .ORIG x3000
      LD R3, PTR
      LD R2, MASK
LOOP  LDR R1, R3, #0
      BRz DONE
      AND R5, R1, R2
      BRz L1
      BRnzp NEXT
L1    ADD R1, R1, R1
      ADD R1, R1, R1
      STR R1, R3, #0
NEXT  ADD R3, R3, #1
      BRnzp LOOP
DONE  HALT
PTR   .FILL x4000
MASK  .FILL x8000
.END
```

a) The following table shows the data values in the memory locations **before** execution. Fill in the table to show the data values **after** execution of the above program.

Address	Value **before** execution	Value **after** execution
x4000	xF093	
x4001	x0039	
x4002	x3004	
x4003	x0140	
x4004	x0000	
x4005	x0BC0	

b) The above program loads a sequence of integers stored in consecutive memory locations starting from x4000, and performs some computation. Explain what this program does for each of the following cases:

i) when it loads a positive integer from a memory address.

ii) when it loads a negative integer from a memory address.

iii) when it loads an integer equal to zero from a memory address.

Indicate when the program ends.

(Remember that in 2's complement representation, a number is negative if the leftmost bit is 1.)

Problem 3 (6 points)

This program counts the number of occurrences of the character "a" in the string "cats hats". Answer the questions below:

```
        .ORIG x3000
SETUP   LEA R3, INPUT
        LD R1, INPUT2
        AND R4, R4, #0
        LD R2, SIZE
NEG     NOT R1, R1
        ADD R1, R1, 1
LOOP    LDR R0, R3, 0
        ADD R3, R3, 1
        ADD R5, R0, R1
        BRnp CONT
        ADD R4, R4, #1
CONT    ADD R2, R2, -1
        BRp LOOP
        HALT
INPUT   .STRINGZ "cats hats"
INPUT2  .STRINGZ "a"
SIZE    .FILL x9
.END
```

a) Create the symbol table for the program. You may not need to use all rows. You can add more rows if needed.

Symbol	**Address**

b) Load this program and run it in PennSim. What is the value stored inside R4 at the end of execution?

c) We now want to count the number of occurrences of the character "a" in the string "fat cat in the hat". What changes do we need to make to the code above to successfully accomplish this? **Clearly indicate** which lines of code you would change and **explain why**.

Problem 4 (14 points)

In this problem, you will write LC-3 assembly code to calculate some statistics about the performance of 30 students in a class, based on their scores for an exam.

Assume that the scores of the 30 students are present in contiguous memory locations, starting at 0x4000.

For each of the following, your program **must** implement **subroutines. Failure to do so will result in deduction of points even if you get the correct results.**

1) Implement a subroutine MAX_SCORES to find the highest score obtained by a student and store this value in the memory location 0x5000. (3 points)
2) Implement a subroutine MIN_SCORES to find the lowest score obtained by a student and store this value in the memory location 0x5001. (3 points)
3) Implement a subroutine AVG_SCORES to calculate the average score of the class and store this value in memory location 0x5002. If the average score is a fraction, round it off to the next higher integer. For example, if the average turns out to be 30.22, the average should be rounded off to 31. (5 points)
4) Implement a subroutine BELOW_AVG to calculate the number of students who have scored below the class average (which has been rounded off), and store this value in memory location 0x5003. (3 points)

Your code should start at memory location 0x3000. You can assume there is no overflow of integers at any step.

Problem 5 (8 points)

In this problem, you will write a LC-3 assembly code that removes blank spaces from a string. Assume that the string starts at memory location 0x4000, and is terminated by a '\0' character (ASCII value = 0). Your program should store the modified string in the memory location starting at 0x4100. You do not need to modify the original string stored at 0x4000. You can assume that the original string at 0x4000 will always be less than 100 characters in length, and it will always start with a letter (A-Z, lowercase or uppercase possible).

Note: If the modified string has 7 characters, and the original string has 15 characters, the last 8 characters of your modified string should be all '\0' (ASCII value = 0).

For example: If the original string at 0x4000 was "aa 12 d e f \0", the modified string at 0x4100 after your program completes execution should be "aa12def\0\0\0\0\0\0\0\0" . Note that the original string has 8 blank space characters, and the modified string has 8 extra "\0" in the end. **Your code should start at memory location 0x3000.**