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; change to .ORIG
LD R4, CHECK
LD R0, VALUE ; change to VAL
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; should be .END not .STOP
```
b) Run the corrected program in PennSim. What does it do? What is the final value stored in R3 after execution ends?

Checks the number of 1s in the binary representation of VAL. R3 = 4

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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value before execution</th>
<th>Value after execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>x4000</td>
<td>xF093</td>
<td>xF093</td>
</tr>
<tr>
<td>x4001</td>
<td>x0039</td>
<td>x00E4</td>
</tr>
<tr>
<td>x4002</td>
<td>x3004</td>
<td>xC010</td>
</tr>
<tr>
<td>x4003</td>
<td>x0140</td>
<td>x0500</td>
</tr>
<tr>
<td>x4004</td>
<td>x0000</td>
<td>x0000</td>
</tr>
<tr>
<td>x4005</td>
<td>x0BC0</td>
<td>x0BC0</td>
</tr>
</tbody>
</table>

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.)

Leaves negative numbers alone, quadruples positive numbers, stops at the first zero.

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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETUP</td>
<td>x3000</td>
</tr>
<tr>
<td>NEG</td>
<td>x3004</td>
</tr>
<tr>
<td>LOOP</td>
<td>x3006</td>
</tr>
<tr>
<td>CONT</td>
<td>x300B</td>
</tr>
<tr>
<td>INPUT</td>
<td>x300E</td>
</tr>
<tr>
<td>INPUT2</td>
<td>x3018</td>
</tr>
<tr>
<td>SIZE</td>
<td>x301A</td>
</tr>
</tbody>
</table>
b) Load this program and run it in PennSim. What is the value stored inside R4 at the end of execution?

\[ R4 = 2 \]

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.

Change input to "fat cat in the hat" and modify SIZE to be x12 (decimal 18). Alternatively, you can rewrite the code to use a pointer to the string and continue running until you reach the '\0' character but this is more complicated.

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.
LD R0, GRADES ; Main function, calls all subroutines
JSR MAX_SCORES
JSR MIN_SCORES
JSR AVG_SCORES
JSR BELOW_AVG
HALT

MAX_SCORES
    AND R1, R1, #0 ; Stores the max score seen till now, initialized to 0
    LD R2, NUM_STUDENTS ; R2 = 30
    LOOP1:    LDR R3, R0, #0 ; R3 = Student's grade
               ADD R0, R0, #1 ; Incrementing to get next student's grade
               NOT R4, R1
               ADD R4, R4, #1 ; R4 = -R1, R1 has the max score seen till now!
               ADD R5, R3, R4
               BRnz SKIP_MAX_UPDATE
               AND R1, R1, #0
               ADD R1, R1, R3 ; Update max score to current student's grade
               SKIP_MAX_UPDATE:    ADD R2, R2, #-1 ; Decrement the counter R2
               BRp LOOP1 ; Continue to loop, not yet seen 30 students
    STI R1, ANSWERS ; Store max score to x5000
    LD R0, GRADES ; Restore R0 to x4000 for next subroutine
    RET ; Return to main

MIN_SCORES
    LD R1, MAX_POS_SCORE ; Stores the min score till now, initialized to \(2^{15} - 1\)
    LD R2, NUM_STUDENTS ; R2 = 30
    LOOP2:    LDR R3, R0, #0 ; R3 = Student's grade
               ADD R0, R0, #1 ; Incrementing to get next student's grade
               NOT R4, R3
               ADD R4, R4, #1 ; R4 = -R3, R3 has the current student's score!
               ADD R5, R1, R4
               BRnz SKIP_MIN_UPDATE
               AND R1, R1, #0
               ADD R1, R1, R3 ; Update max score to current student's grade
               SKIP_MIN_UPDATE:    ADD R2, R2, #-1 ; Decrement the counter R2
               BRp LOOP2 ; Continue to loop, not yet seen 30 students
               LD R6, ANSWERS ; R6 = x5001
               STR R1, R6, #1 ; Store max score to x5001
               LD R0, GRADES ; Restore R0 to x4000 for next subroutine
               RET ; Return to main
AVG_SCORES
    AND R1, R1, #0 ; Stores the sum of scores seen till now, initialized to 0
    LD R2, NUM_STUDENTS ; R2 = 30
    ; Code for calculating sum of student's scores
    LOOP3:    LDR R3, R0, #0 ; R3 = Student's grade
              ADD R0, R0, #1 ; Incrementing to get next student's grade
              ADD R1, R1, R3 ; Update the sum of scores in R1
              ADD R2, R2, #-1 ; Decrement the counter R2
              BRp LOOP3 ; Continue to loop, not yet seen 30 students

    ; Code for division follows!
    LD R2, NUM_STUDENTS ; R2 = 30
    NOT R2, R2
    ADD R2, R2, #1 ; R2 = -30
    AND R3, R3, #0 ; R3 will store the division result

    LOOP_DIV:  ADD R1, R1, R2 ; Decrement remaining number by 30
               BRnz DIV_OVER
               ADD R3, R3, #1 ; R3 incremented
               BRnzp LOOP_DIV ; Continue dividing
    DIV_OVER:   ADD R3, R3, #1 ; Division just got over, and we need to increase quotient!
                LD R6, ANSWERS ; R6 = x5000
                STR R3, R6, #2 ; Store avg score (in R3) to x5002
                LD R0, GRADES ; Restore R0 to x4000 for next subroutine
                RET ; Return to main

BELOW_AVG
    LD R6, ANSWERS
    LDR R6, R6, #2 ; R6 = avg score calculated from AVG_SCORES function, residing in x5002
    NOT R6, R6
    ADD R6, R6, #1 ; R6 = -avg score
    AND R1, R1, #0 ; Stores the number of students with score < average, initialized to 0
    LD R2, NUM_STUDENTS ; R2 = 30
    LOOP4:    LDR R3, R0, #0 ; R3 = Student's grade
              ADD R0, R0, #1 ; Incrementing to get next student's grade
              ADD R4, R3, R6 ; R3 has current Student's score! If R4 < 0,
              increment num of ; students below average score
              BRzp SKIP_COUNT_UPDATE
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\0\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.**

```
.ORIG x3000
LD R0, INPUT_STRING ;R0 will track the location at the input string
LD R1, OUTPUT_STRING ;R1 will track the location at the output string
LD R6, BLANK_SPACE ;R6 = 32, ASCII value of blank space ''
NOT R6, R6
ADD R6, R6, #1 ;R6 = -32, useful for comparison later on
LOOP: LDR R2, R0, #0 ;R2 contains the character at location R0+0
BRnz INPUT_OVER ;if current character <=0, string over. ASCII value is >= 0
ADD R0, R0, #1 ;Increment R0 to access next character in input string
ADD R5, R2, R6 ;Check and skip storing if character is blank space
BRz SKIP_STORING_SPACE
STR R2, R1, #0 ;Store current character (in R2) to output string
ADD R1, R1, #1 ;Increment R1 to store next character in output string
SKIP_STORING_SPACE: BRnzp LOOP ;jump to loop
INPUT_OVER: NOT R4, R1
```

ADD R1, R1, #1 ;Update count, include current student with score below avg
SKIP_COUNT_UPDATE: ADD R2, R2, #-1 ;Decrement the counter R2
BRp LOOP4 ;Continue to loop, not yet seen 30 students
LD R6, ANSWERS ;R6 = x5000
STR R1, R6, #3 ;Store count to x5003
RET ;Return to main

GRADES: .FILL x4000
NUM_STUDENTS: .FILL x001E
ANSWERS: .FILL x5000
MAX_POS_SCORE: .FILL x7fff
ADD R4, R4, #1 ; R4 = -R1
LD R6, MAX_SIZE
ADD R5, R0, R6
ADD R3, R5, R4 ; R3 = number of blank spaces!
AND R2, R2, #0 ; R2 = 0
LOOP1: ADD R3, R3, #1 ; Decrement R3
BRn END
STR R2, R1, #0 ; Store 0 to current location in output string
ADD R1, R1, #1 ; Increment R1 to the next location in output string
BRnzp LOOP1
END: STR R2, R1, #0 ; Store 0 at the end of string
HALT

INPUT_STRING: .FILL x4000
OUTPUT_STRING: .FILL x4100
BLANK_SPACE: .FILL x20
MAX_SIZE: .FILL x100