Homework 8 CS/ECE 252: Sec 1 & 2 [Due 11:59AM on Wed, May 07]

Primary contacts for this hw: Sujith Surendran [sujiths at cs dot wisc dot edu],
Murali Sivalingam [murali10 at cs dot wisc dot edu]

Important Notes:

- Answers to questions 1, 2, 3 and 4 should be handed to your instructor in hard copy on the due date during your class time.
- Question 5 must be submitted electronically to the Learn@UW dropbox before 11.59 AM on the due date. Submit only one archive file, named exactly like: <lastname_firstname>[.zip] to the dropbox folder homework8. You have to submit the .txt files (which contains the assembly code) for these problems and the files should be named hw8_p#.txt, i.e, the file for problem 5 should be named hw8_p5.txt. Since your code is tested automatically, it is important to stick to this naming convention, otherwise you will lose credit, even if your code is working correctly.

- The programs which you write should always start at address x3000 and end with a HALT instruction (HALT).

- You can submit your code for problem 5 as many times as you want, until 11:59 AM on Wednesday, May 07, 2014. We will consider your latest submission for grading.

Problem 1 (3 points)

For the below code, if the label "START " is at address x4FFF, what are the addresses of the labels WORD, SENT and DATA?

```
START .FILL xABCD
WORD  .BLKW 10
SENT  .STRINGZ “Hurray!! My last homework”
DATA  .FILL xBABA
```

WORD: 0x5000
SENT : 0x500A
DATA: 0x5024

Problem 2 (4 points)

Suppose we define a new Interrupt service routine starting at the memory location 0x5000 as below.

```
 .ORIG x5000
  ST R7 , TEMP_R7
 GETC
 OUT
 LD R7, TEMP_R7
 RET

TEMP_R7  .FILL x1234
```

The routine reads in a single character and echoes it to the screen. Assume the memory location 0x0045 contains the value 0x5000.

a. (2 points) Identify the instruction that would invoke this service routine.

(a) TRAP x45

b. (2 points) Does the service routine defined above work. If so explain what would happen to the registers when the service routine is executed ?

(b) Yes, this routine will work, but whatever value was in R0 before TRAP x45 is executed will be overwritten during the subroutine.

Problem 3 (5 points)

An LC-3 program is provided below:

```
            .ORIG x3000
         LD   R0, ASCII
         LD   R1, NEG
LOOP     LDI  R2, DSR
         BRzp LOOP
         STI  R0, DDR
         ADD  R0, R0, #-2
         ADD  R3, R1, R0
         BRp  LOOP
         HALT

ASCII    .FILL x0050
NEG      .FILL xFFBC
DSR      .FILL xFE04            ; Address of DSR
DDR      .FILL xFE06            ; Address of DDR
         .END
```

Note: You can download the program from here hw8_p3.txt

a) (3 Points) Run the program on PennSim and give a brief explanation of what the program does. (ie, specify what will be the output of the program)

Output will be PNLJHF

b) (1 Point) What value will be contained in R3 after the execution of the program?

0

c) (1 Point) What is the purpose of the Display Status Register (DSR) in the above program?

Bit [15] is one when device ready to display another char on screen.
Bit [14] : Enables monitor interrupt
DSR contains information about the current status of the monitor

Problem 4 (8 points)

The following code reads two numbers from the memory and finds if they have the same absolute value. If the absolute values are equal, 1 is stored in R5; otherwise 0 is stored in R5. The subroutine starting at the label "FINDABS" finds the absolute value
of the argument.

```
.ORIG   x3000; Instructions start at x3000
; Initialization
AND R5, R5, #0
AND R7, R7, #0   ; Counter to hold number of times a negative value is passed to FINDABS.
LD  R1, DATA1
LD  R2, DATA2
_ADD R0, R1, #0__________________  ; Prepare the arguments for DATA1
ST  R5, SaveR5        ; Save R5 before calling subroutine
JSR  FINDABS          ; Call subroutine FINDABS
LD  R5, SaveR5        ; Restore R5
_ADD R3, R0, #0______________  ; Store FINDABS(DATA1) in R3
ADD R0, R2, #0        ; Copy R2 to R0
ST  R5, SaveR5
JSR FINDABS           ; Call subroutine FINDABS
LD    R5, SaveR5
NOT   R0, R0
ADD   R0, R0, #1
ADD   R3, R3, R0        ; Find R3 - R0
BRnp  STOP
ADD   R5, R5, #1
STOP  HALT

; Subroutine for absolute value
; Argument is passed in register ___R0____ (fill)
FINDABS    ADD  R0, R0, #0 ; Set condition code based on R0
           BRzp ENDABS
           NOT  R5, R0
           ADD  R0, R5, #1
            ADD R7, R7, #1
ENDABS     RET   ; Value is returned in register _R0______ (fill)

; Values
SaveR5 .FILL x0000
DATA1   .FILL x000A
DATA2   .FILL xFFF6
  .END
```

a) (4 points) Some of the lines in the code are missing. Fill in the missing lines. Also fill in the blanks in the two comment lines as indicated.

b) (2 points) Identify whether FINDABS is a caller-save or callee-save subroutine. Give reasons to support your answer.
Caller save subroutine - Caller saves R5 before calling FINDABS

c) (2 points) There is a problem with the above program. Identify the error and how to fix it.
R7 is modified in FINDABS so the return address is incorrect, leading to an incorrect execution

Problem 5 (10 points)

Write a program to take a sentence from the user (not exceeding 30 characters) and reverse the words in the sentence and display the reversed string. For example, if the user inputs as " Done with my semester" , the output should be displayed as "semester my with Done". Assume that the user terminates his/her input with an enter key and no punctuations are used except spaces. Also assume that the first character is not a space and all words are separated by a single space(no multiple spaces in the input) .

N**<u>ote:</u>**

- You need to submit the program as hw8_p5.txt
- Use this template for writing your code: hw8_p5_template.txt
- Use this script for running your code: hw8_p5_script.txt

```
 .ORIG x3000
  LEA R0, MSG
  TRAP x22 ; PUTS -- Print the message to user to enter the string
  LEA R1, FCHAR
  ADD R3, R1, 0 ; R3 pointer to FCHAR
  LD  R2, ENT ; EOL comparator
  LD  R5, NEGSPACE

NEXT    TRAP x20  ; GETC read the char from user
        TRAP x21  ; OUT echo it back to the suer

  ADD R4, R0, R2
  BRz OUTPUT ; End of input get into output mode

  STR R0, R3, 0
  ADD R3, R3, 1
  BRnzp NEXT

; Data loaded into FCHAR
OUTPUT  ADD R3, R3, -1
  NOT R1, R1
  ADD R1, R1, 1
  LEA R0, PRINT
  TRAP x22 ; PUTS caption for output

LOOP    ADD R4, R1, R3
  BRz DONE
```

```
        LDR R0, R3, 0   ; LOAD VALUE POINTED BY R3 (POINTER) TO R0
        ADD R6,R0,R5    ; CHECK IF R0 IS A SPACE IF SO PRINT THE RESULTS SO
FAR
        BRZ REVERSE

       ; TRAP x21    ; OUT the reversed string
        ADD R3, R3, -1
        BRnzp LOOP

REVERSE AND R0, R0, 0 ; REPLACE RO WITH 0
        STR R0 ,R3, 0 ; REPLACE 0 WITH SPACE
        ADD R3, R3, 1 ; INCREMENT R3 TO POINT TO THE NEXT MEMORY LOCATION
TO PRINT DATA FROM THERE.
        ADD R0, R3, 0
        PUTS
        LD R0, SPACE
        OUT
        ADD R3, R3, -2  ; Iterate repeatedly
        BRnzp LOOP

DONE  ADD R0, R3, 0
      PUTS
      HALT
; DATA REGION

MSG       .STRINGZ  "Please enter a string (max length 30): "
PRINT     .STRINGZ  "Output: "
ENT       .FILL   -10 ; new line feed
NEGSPACE  .FILL   xFFE0 ; decimal -32, -20 - NEGHEX for space
SPACE     .FILL  x20
FCHAR     .BLKW   31 ; Used for storing the input results from user
    .END
```