CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Gurindar Sohi

TAs: Sujith Surendran, Lisa Ossian, Minsub Shin

Midterm Examination 4

In Class (50 minutes)

Wednesday, December 10, 2014

Weight: 17.5%

NO: BOOK(S), NOTE(S), OR CALCULATORS OF ANY SORT.

The exam has **ten** pages. **Circle your final answers**. Plan your time carefully since some problems are longer than others. You **must turn in the pages 1-8**. Use the blank sides of the exam for scratch work.

Note: LC-3 instruction set is provided on Page 9. Trap Codes and Assembler directives are provided on page 10

LAST NAME: ____

FIRST NAME: ____

ID#: ____

Problem	Maximum Points	Points Earned
1	10	
2	8	
3	4	
4	6	
5	2	
Total	30	

Problem 1: Short answer questions (10 points)

a) (1 point) How many accesses to memory are made after the instruction fetch phase of a LDI instruction? Show your work.

b) (1 point) For rare events, would you prefer interrupt-driven I/O or polling I/O? Justify your answer.

c) (1 point) Briefly explain the difference between asynchronous and synchronous I/O events.

d) (2 points) An LC-3 assembly program contains the following instruction:

```
MAIN       LD R5, MAIN
```

The symbol table entry for MAIN is `x4000`. What will be the value of R5 after the execution of the above instruction? Show your work.

e. **(2 points)** Briefly describe what happens during the linking and loading phases of an assembly program?

f. **(3 points)** Identify three **assembly** errors in the following code:

```
         .ORIG x3000

         LEA R1, NUMBER
         LD R1, NUMBER
LOOP     NOT R5, #2
         TRAP x29
         BRzp LOOP2
         AND R1, R1, FIVE
         LD R1, FIVE
         BRp LOOP

LOOP     HALT

FIVE     .FILL #5
NUMBER   .FILL x60
         .END
```

Problem 2: Two-pass assembly process (8 points)

a) **(3 points)** Consider the following LC-3 assembly program.

```
        .ORIG x3000

        LEA R2, STRING
        LD  R3, NUMBER
HERE    ADD R1, R2, R3
        ADD R2, R1, #0
        LDR R0, R1, #0
        BRz DONE
        OUT
        BR HERE

THIS   .BLKW 6
STRING .STRINGZ “2down_3to_go”
NUMBER .FILL x4

DONE   HALT

       .END
```

What would be the output on the console if you run the above code in Pennsim?

b) **(3 points)** In the first pass, the assembler creates the symbol table. Fill in the symbol table created by the assembler for this program

Symbol	**Address**

c) **(2 points)** In the second pass, the assembler creates a binary version (.obj) of the program, using the entries from the symbol table shown below. Given that the following symbol table entries were generated in the first pass of assembly (for another program), fill in the binary code generated by the assembler for the two instructions located at x3000 and x3001.

Symbol Table:

Label	**Address**
`ADDRESS`	`x3015`
`NEXT`	`x3016`

Address	**Assembly code**	**Binary Code**
`x3000`	`LD R0, ADDRESS`	
`x3001`	`BRnp NEXT`	

Problem 3 **(4 points)**

Consider the program below, the goal of which is to multiply the value in memory location corresponding to label Input1 with the value in memory location corresponding to label Input2 and store the result in the memory location corresponding to label RESULT.

```
       .ORIG x3000
        LD R2, ZERO
        LD R0, Input1
        LD R1, Input2

LOOP    BRn DONE
        ADD R2, R2, R0
        ADD R1, R1, -1
        BR LOOP

DONE    ST R2, RESULT
        HALT

RESULT .FILL    x0000
ZERO   .FILL    x0000
Input1 .FILL    x0007
Input2 .FILL    x0002
       .END
```

a. **(2 points)** What is the value at RESULT after executing the HALT instruction? Write the answer in hexadecimal. Show your work.

b. **(2 points)** From your answer from 3a, you would have noticed that the answer is not the result of multiplication of input1 and input2. Identify what caused this error, and how do you fix it?

Problem 4: Traps and Subroutines **(6 points)**

Suppose we want to write a new TRAP subroutine, TRAP x02. This subroutine takes an input from the caller of the subroutine through register R2. R2 has the memory address of the first character of a string. The subroutine then prints all characters that are not ‘a’. Fill in the missing blanks to complete this subroutine code. Assume that we are implementing a callee-save subroutine. Save only those registers that are necessary.

Assume that the trap vector table (also known as the system control block) is shown below:

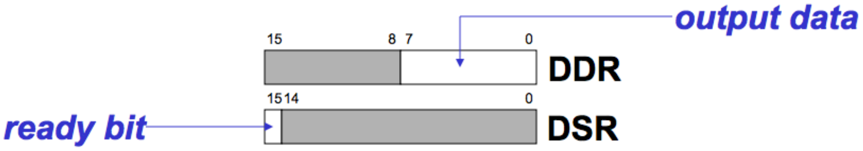
Address	Value
x0001	x2400
x0002	x2500
x0003	x2600

```
       .ORIG _____

STORE   ST __, SAVEREG1
        ST __, SAVEREG2
        ST __, SAVEREG3
        ST __, SAVEREG4

LOOP    LDR R0, R2, #0 ;Load a character from the string.
        ______________ ;If there are no more characters,goto RESTORE
        LD R5, neg_a   ;Load negative of ASCII of ‘a’ into R5.
        ADD R2, R2, #1 ;Increment pointer to get next character.
        ______________ ;Determine if current character equals ‘a’.
        BRz LOOP      ;If character is ‘a’, go load next character.
        ______________ ;Print the extracted character.
        BR LOOP        ;Branch to LOOP.


RESTORE LD __, SAVEREG1
        LD __, SAVEREG2
        LD __, SAVEREG3
        LD __, SAVEREG4
        RET

SAVEREG1     .BLKW 1
SAVEREG2     .BLKW 1
SAVEREG3     .BLKW 1
SAVEREG4     .BLKW 1
neg_a        .FILL 0xFF9D ; This is the negative of ASCII of ‘a’
        .END
```

Problem 5: I/O **(2 points)**

The following code segment should display the string specified at the "STRING" label on to the console. Write the missing assembly instructions of the program (without using PUTS/PUTC/OUT/TRAP instructions).

Hint: Make use of the DSR and DDR, as shown in the figure below.



```
         .ORIG x3000

         LEA R3, STRING
NEXT     LDR R0, R3, #0

_______________________

_______________________

_______________________

_______________________

         ADD R3, R3, #1 ; Point to the next character

         BR NEXT
END      HALT

STRING .STRINGZ "Enjoy_your_holidays!" ; String to display
DSR    .FILL xFE04 ; Display status register location
DDR    .FILL xFE06 ; Display data register location

         .END
```

LC 3 Instruction Set to be provided here

TRAP CODES

Code	*Equivalent*	*Description*
HALT	TRAP x25	Halt execution and print message to console.
IN	TRAP x23	Print prompt on console, read (and echo) one character from keybd. Character stored in R0[7:0].
OUT	TRAP x21	Write one character (in R0[7:0]) to console.
GETC	TRAP x20	Read one character from keyboard. Character stored in R0[7:0].
PUTS	TRAP x22	Write null-terminated string to console. Address of string is in R0.

ASSEMBLER DIRECTIVES

Opcode	*Operand*	*Meaning*
.ORIG	address	starting address of program
.END		end of program
.BLKW	n	allocate n words of storage
.FILL	n	allocate one word, initialize with value n
.STRINGZ	n-character string	allocate n+1 locations, initialize w/characters and null terminator