CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Mark D. Hill

TAs: Sujith Surendran, Pradip Vallathol

Midterm Examination 4

In Class (50 minutes)

Wednesday, Dec 11, 2013

Weight: 17.5%

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

The exam has 10 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 the last page.

LAST NAME: _____

FIRST NAME: _____

ID# _____

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

Problem 1: Assembly Errors **(4 Points)**

Identify 4 errors in the following Assembly language program:

```
          .ORIG x3501

          LD R0, DEFAULT
LOOP      TRAP x20
          TRAP x21
          AND R0, R0, R0
          BRzp LOOP
          TRAP x20
          NOT R2, #3
          ADD R0, R0, ONE
          LD R1, R2, #3
          BRp LOOP

LOOP      HALT
ONE       .FILL     x1
DEFAULT   .ASCII    #2
          .END
```

(a)

(b)

(c)

(d)

Problem 2: Two Pass Assembly Process **(8 Points)**

(a) **(2 Points)** Consider the following assembly code. What will be the output on the console if you run this code on PennSim?

```
        .ORIG x3800
        LEA R3, INPUT
        LD  R1, SIZE
        ADD R3, R3, R1

LOOP    LDR R0, R3, 0
        TRAP x21
        ADD R3, R3, -1
        ADD R1, R1, -1
        BRp LOOP

        HALT

INPUT   .STRINGZ  "RtbY"
STRING  .BLKW   3
SIZE    .FILL   3
        .END
```

Answer:

(b) **(4 Points)** In the first pass, the assembler creates the symbol table. Fill in the symbol table created by the assembler for the program in Problem 2(a).

Label	Address

(c) **(2 Points)** In the second pass, the assembler creates a binary (.obj) version of the program, using the entries from the symbol table. Assume that there exists another program at 0x3000, whose assembly instructions are as shown below. If the following symbol table entries were generated in the first pass of the assembly for this program, write the binary code generated by the assembler for the two instructions at 0x3000 and 0x3001.

Symbol Table:

Label	Address
INT	x3021
LOOP	x3011

Generated Binary code:

Address	Instruction	Binary Code
x3000	LD R0, INT	
x3001	BRp LOOP	

Problem 3: Subroutines and Traps **(8 Points)**

Suppose we want to write a new TRAP subroutine, TRAP x33, which takes a string input from the user. The trap subroutine starts from address x2200 and does the following:

1) It takes a character input from the user
2) It then displays this character (which the user inputs) on the console
3) After that, it stores the user input characters in consecutive memory locations starting from the address location present in register R1. It then repeats (1), (2) and (3) until user inputs 'Z'.
4) It uses a "callee-save" strategy and ensures that none of the register values are modified by it.
5) It uses R2 to store the ASCII value corresponding to -Z

(a) **(6 Points)** Fill in the missing parts of the trap subroutine.

```
        .ORIG _________
            ST R0, SAVEREG1
            ST __, SAVEREG2
            ST R2, SAVEREG3
            ST __, SAVEREG4

            LD  R2, NEGZ

NEXT        TRAP ___
            TRAP ___
            STR R0, R1, #0

            _______________
            ADD R0, R0, R2
            BRnp NEXT

            LD R0, SAVEREG1
            LD __, SAVEREG2
            LD R2, SAVEREG3
            LD __, SAVEREG4

            RET
;Data Region
NEGZ        .FILL xFFA6      ; xFFA6 = FFFF – ASCII value of Z
SAVEREG1    .BLKW 1
SAVEREG2    .BLKW 1
SAVEREG3    .BLKW 1
SAVEREG4    .BLKW 1
```

(b) **(2 Points)** Given the following Trap vector table entry:

Address	Content
x44	X26

Give the assembly instruction that you would use to call the TRAP routine corresponding to this entry. Provide reasons to justify your answer.

Problem 4: I/O **(4 Points)**

(a) **(4 Points)** The following code segment should display the string specified at the "INPUT" label on to the console. Write the missing assembly instructions of the program (without using PUTS/PUTC/TRAP instructions).

- **Note:** The instructions which are missing should jump to halt if it is the end of the string. Else, it should print the character on the console.

```
        .ORIG   x3000

        LEA R2, INPUT

NEXT    LDR R0, R2, #0
_____________________
_____________________
_____________________
_____________________

        ADD R2, R2, #1      ; Point to the next character
        BR  NEXT

END     HALT


INPUT   .STRINGZ   "All the best!"  ; String to display
DSR     .FILL   xFE04   ; Display status register location
DDR     .FILL   xFE06   ; Display data register location
        .END
```

Problem 5: Short Answer Questions **(6 Points)**

(a) **(1 Point)** Briefly state a scenario where you would prefer interrupt-driven I/O over polling based I/O?

(b) **(1 Point)** Suppose two I/O devices sends interrupts to the CPU at the same time. How does the CPU decide which interrupt to service first?

(c) **(2 Points)** An LC-3 assembly program contains the following instruction:

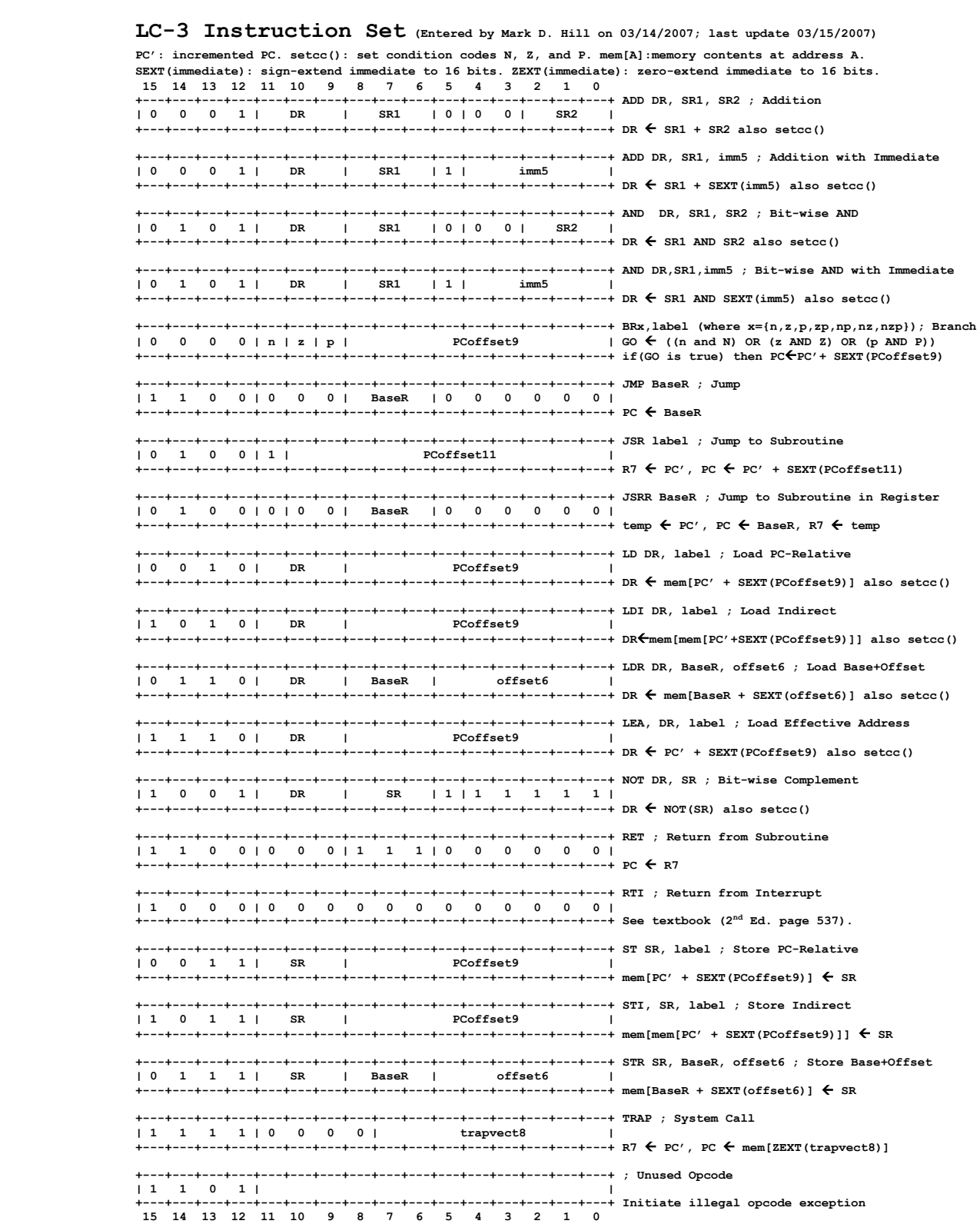
```
FLOAT     LD R2, FLOAT
```

The symbol table entry for FLOAT is x3000. What will be the value of R2 after the execution of the above instruction?

(d) **(2 Points)** Briefly state what happens in Linking and Loading phases for an assembly program.

LC-3 Instruction Set (Entered by Mark D. Hill on 03/14/2007; last update 03/15/2007)

PC': incremented PC. setcc(): set condition codes N, Z, and P. mem[A]:memory contents at address A.
SEXT(immediate): sign-extend immediate to 16 bits. ZEXT(immediate): zero-extend immediate to 16 bits.



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