CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN-MADISON

Prof. Mark D. Hill
TAs Marc de Kruijf & Sanghamitra Roy

Midterm Examination 4
In Class (50 minutes)
Wednesday, May 9, 2007
Weight: 15%

CLOSED BOOK, NOTE, CALCULATOR, PHONE, & COMPUTER.

The exam in two-sided and has **NINE** pages, including two blank pages and a copy of the ***LC-3 Instruction Set handout*** on the final page (please feel free to detach this final page, but insert it into your exam when you turn it in).

Plan your time carefully, since some problems are longer than others.

NAME: ____

ID# ____

Problem Number	Maximum Points	Points Awarded
1	5	
2	4	
3	4	
4	8	
5	4	
6	3	
7	2	
Total	30	

Problem 1 (5 points)

An assembly language LC-3 program is given below:

```
        .ORIG   x3003
        LEA     R1, DATA
        LDR     R2, R1, #0
LOOP    ADD     R2, R2, #-3
        BRzp    LOOP
        HALT

DATA    .FILL   x000C
        .END
```

a. Create a symbol table for the program:

Symbol	Address

b. How many times will the instruction at the memory address labeled LOOP execute?

Problem 2 (4 points)

a. What is the purpose of the HALT statement?

b. Is it meaningful to have more than one HALT statement in a single-file LC-3 program? Explain.

c. What is the purpose of .END pseudo-op?

d. Is it meaningful to have more than one .END in a single-file LC-3 program? Explain.

Problem 3 (4 points)

Regarding the assigned reading "RFID Inside" on RFID implants:

a. Give two different potential benefits of RFID implants.

b. Give two different potential drawbacks of RFID implants.

c. In what way was Wisconsin mentioned in the article?

Problem 4 (8 points)

The following program calculates the sum of absolute values of two numbers and stores the sum in R4. The subroutine at the label "ABS" finds the absolute value of the argument.

```
      .ORIG  x3000    ; Instructions start at x3000;
      AND R4, R4, #0  ; Clearing R4
      LD  R1, VAL1
      LD  R2, VAL2

      _______________ ; Prepare argument VAL1 (fill)
      JSR ABS         ; Call subroutine ABS
      ADD R4, R0, #0  ; Add Abs(VAL1) to R4
      _______________ ; Prepare argument VAL2 (fill)
      JSR ABS         ; Call subroutine ABS
      ADD R4, R4, R0  ; Add Abs(VAL2) to R4
      HALT

                      ; Argument passed in register ___ (fill)
ABS   ST  ___, SaveR  ; Save register ___ (fill)
      ADD R0, R0, #0  ; Set condition code based on R0
      BRzp    NEXT
      NOT R4, R0
      ADD R0, R4, #1
NEXT  LD  ___, SaveR  ; Restore register ___ (fill)
      RET             ; Value is returned in register ___ (fill)

                      ; Values
SaveR .FILL    x0000
VAL1  .FILL    x0005  ;  5
VAL2  .FILL    xFFFB  ; -5
      .END
```

a. Fill in the blanks in the above program at all places indicated by "**(fill)**".

b. What is the value in register R4 at the end of program execution?

Problem 5 (4 points)

An LC-3 assembly language program is given below. Carefully read the program and answer the questions that follow. Adding comments will help in partial credit.

Label	**Assembly language instruction**
START	LDI R1, KBSR ;
	BRzp START ;
	LDI R0, KBDR ;
LOOP	LDI R1, DSR ;
	BRzp LOOP ;
	STI R0, DDR ;
	HALT ;
KBSR	.FILL xFE00 ;
KBDR	.FILL xFE02 ;
DSR	.FILL xFE04 ;
DDR	.FILL xFE06 ;
	.END

a. What does this program do?

b. What is the purpose of the KBSR?

Problem 6 (3 points)

a. What does the JSR instruction do? How does it differ from a JMP instruction?

b. Why must a RET instruction be used to return from a TRAP routine? Why won't a BR (Unconditional branch) instruction work instead?

Problem 7 (2 points)

In lecture, we discussed implementing a program denoted Halt(P,I).

a. What is Halt(P,I) supposed to do?

b. What are the alternative answers Halt(P,I) may return?

Scratch Sheet 1 (in case you need additional space for some of your answers)

Scratch Sheet 2 (in case you need additional space for some of your answers)

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. Page 2 has an ASCII character table.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ ADD DR, SR1, SR2 ; Addition
| 0   0   0   1 |    DR     |    SR1    | 0 | 0   0 |    SR2    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← SR1 + SR2 also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ ADD DR, SR1, imm5 ; Addition with Immediate
| 0   0   0   1 |    DR     |    SR1    | 1 |       imm5        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← SR1 + SEXT(imm5) also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ AND  DR, SR1, SR2 ; Bit-wise AND
| 0   1   0   1 |    DR     |    SR1    | 0 | 0   0 |    SR2    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← SR1 AND SR2 also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ AND DR, SR1, imm5 ; Bit-wise AND with Immediate
| 0   1   0   1 |    DR     |    SR1    | 1 |       imm5        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← SR1 AND SEXT(imm5) also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ BRx, label (where x = {n,z,p,zp,np,nz,nzp}) ; Branch
| 0   0   0   0 | n | z | p |             PCoffset9             | GO ← ((n and N) OR (z AND Z) OR (p AND P))
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ if (GO is true) then PC ← PC′ + SEXT(PCoffset9)

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ JMP BaseR ; Jump
| 1   1   0   0 | 0   0   0 |   BaseR   | 0   0   0   0   0   0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ PC ← BaseR

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ JSR label ; Jump to Subroutine
| 0   1   0   0 | 1 |                PCoffset11                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ R7 ← PC′, PC ← PC′ + SEXT(PCoffset11)

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ JSRR BaseR ; Jump to Subroutine in Register
| 0   1   0   0 | 0 | 0   0 |   BaseR   | 0   0   0   0   0   0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ temp ← PC′, PC ← BaseR, R7 ← temp

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ LD DR, label ; Load PC-Relative
| 0   0   1   0 |    DR     |             PCoffset9             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← mem[PC′ + SEXT(PCoffset9)] also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ LDI DR, label ; Load Indirect
| 1   0   1   0 |    DR     |             PCoffset9             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← mem[mem[PC′ + SEXT(PCoffset9)]] also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ LDR DR, BaseR, offset6 ; Load Base+Offset
| 0   1   1   0 |    DR     |   BaseR   |        offset6        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← mem[BaseR + SEXT(offset6)] also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ LEA, DR, label ; Load Effective Address
| 1   1   1   0 |    DR     |             PCoffset9             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← PC′ + SEXT(PCoffset9) also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ NOT DR, SR ; Bit-wise Complement
| 1   0   0   1 |    DR     |    SR     | 1 | 1   1   1   1   1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ DR ← NOT(SR) also setcc()

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ RET ; Return from Subroutine
| 1   1   0   0 | 0   0   0 | 1   1   1 | 0   0   0   0   0   0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ PC ← R7

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ RTI ; Return from Interrupt
| 1   0   0   0 | 0   0   0   0   0   0   0   0   0   0   0   0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ See textbook (2nd Ed. page 537).

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ ST SR, label ; Store PC-Relative
| 0   0   1   1 |    SR     |             PCoffset9             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ mem[PC′ + SEXT(PCoffset9)] ← SR

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ STI, SR, label ; Store Indirect
| 1   0   1   1 |    SR     |             PCoffset9             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ mem[mem[PC′ + SEXT(PCoffset9)]] ← SR

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ STR SR, BaseR, offset6 ; Store Base+Offset
| 0   1   1   1 |    SR     |   BaseR   |        offset6        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ mem[BaseR + SEXT(offset6)] ← SR

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ TRAP ; System Call
| 1   1   1   1 | 0   0   0   0 |           trapvect8           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ R7 ← PC′, PC ← mem[ZEXT(trapvect8)]

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ ; Unused Opcode
| 1   1   0   1 |                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ Initiate illegal opcode exception
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
```