CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Mark D. Hill

TAs: Mona Jalal, Preeti Agarwal, Pradip Vallathol, Rebecca Lam

Midterm Examination 4

In Class (50 minutes)

Wednesday, May 8, 2013

Weight: 17.5%

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

The exam has 12 pages. **Circle your final answers**. Plan your time carefully since some problems are longer than others. You **must turn in the pages 1-9**. The LC-3 instruction set is provided to you on the last page.

LAST NAME: ____

FIRST NAME: ____

ID# ____

Problem	Maximum Points	Points Earned
1	4	
2	5	
3	3	
4	5	
5	5	
6	3	
7	5	
Total	30	

Problem 1: Multiple Choice Questions (4 Points)

For the following questions, select the **best** answer. Choose only **one answer per question**.

i. The TRAP instructions in LC-3 are similar to which of the following instructions in terms of the number of memory accesses that are made to the fetch and execute the instruction?
   a. LD
   b. LEA
   c. LDI
   d. LDR

ii. Which of the following is **not** true about polling?
   a. The CPU keeps monitoring status register.
   b. CPU cannot perform other tasks during polling.
   c. Polling wastes a lot of CPU time.
   d. Polling requires changes to the Fetch and Decode logic of the CPU.

iii. Which of the following is **not** true about **comments** in an LC-3 program?
   a. It is used by the assembler to understand the program.
   b. Can be used to separate pieces of the program.
   c. Anything after the semicolon is a comment.
   d. They can be used multiple times in a program.

iv. **JSRR R5** is equivalent to
   a. LEA R5, #1
      JMP R7
   b. LEA R7, #1
      JMP R5
   c. LEA R5, #1
      JMP R5
   d. LEA R7, #1
      JMP R7
   e. All of the above are equivalent

Problem 2: Assembly Process **(5 Points)**

Answer the questions below for the following program:

```
       .ORIG x4000
       LD  R2, LOW_P
       NOT R2, R2
       ADD R2, R2, #1
       LEA R0, STRG
       ; Comment 1
L1     LDR R1, R0, #0
       BRz DONE
       ADD R3, R1, R2
       BRnp SKIP

       LD  R1, UPP_P
       STR R1, R0, #0
SKIP   ADD R0, R0, #1
       BRnzp L1
DONE   LEA R0, STRG
       PUTS        ; Display the string at the address in R0
       HALT
LOW_P  .FILL x70   ; ASCII Character 'p'
STRG   .STRINGZ "Salt and Pepper"
UPP_P  .FILL x50   ; ASCII Character 'P'
       .END
```

a. Fill out the following symbol table: **(3 Points)**

SYMBOL	**ADDRESS**
L1	
SKIP	
DONE	
LOW_P	
STRG	
UPP_P	

b. What is the output of this program? **(2 Points)**

Problem 3: Assembly Errors **(3 Points)**

Identify the assembly errors in the following assembly program.

```
          .ORIG x3000

          ADD R4, R4, R4
          ; OR R2, R3, R4

LOOP      ADD R1, R2, #21
          AND R3, R3, #0
          ADD R3, R3, #-1
          BRzp JUMP

STRG      .STRINGZ "Error"

HALT      STR R4, R4, #16
          TRAP x25

          .END
```

(a)

(b)

(c)

Problem 4: TRAPS **(5 Points)**

Suppose the following LC-3 subroutine implements a new service routine called **GETS**. The subroutine will store the input string starting at the address in R0 and then return to normal execution. It performs this operation by repeatedly taking input characters from the keyboard and storing it in the location specified by R0 until it sees the **'\n'** character.
Note: The most significant bit of the KBSR is 1 if keyboard has received a new character.

a. Fill in the blanks. **There should be only one instruction per line**. **(4 Points)**

```
     .ORIG x0540
     ST  R0, R0_TMP
     ST  R1, R1_TMP
     ST  R2, R2_TMP
L1   LDI R1, KBSR
     (a)_____________    ; Check KBSR
     (b)_____ R2, KBDR   ; Load value in the KBDR into R2
     LD  R1, NEGCHAR
     ADD R1, R1, R2
     BRz DONE            ; Check for '\n'
     STR R2, R0, #0
     ADD R0, R0, #1
     BRnzp     L1
DONE (c)______________
     STR R2, R0, #0      ; Store NULL CHAR
     LD  R2, R2_TMP
     LD  R1, R1_TMP
     LD  R0, R0_TMP
     (d)________________

KBSR      .FILL xFE00    ; Address of KBSR
KBDR      .FILL xFE02    ; Address of KBDR
NEGCHAR   .FILL xFFF6    ; Negative value of character '\n'
R0_TMP    .FILL 0
R1_TMP    .FILL 0
R2_TMP    .FILL 0
     .END
```

b. Assume the above assembly code is a service routine that can be called using TRAP x33. What is the address of the corresponding System Control Block entry and what are its contents? Give your answer in hex. **(1 Point)**

Address of trap vector table entry	**Contents at this memory location**

Problem 5: Subroutines **(5 Points)**

a. There is a problem with the below assembly code segment for a subroutine called PUTCH. What is it, and how can you fix the error? **(2 Points)**

```
          .ORIG x5010
PUTCH
          ST   R0, TMP_R0
          ADD  R0, R4, 0
          OUT             ; TRAP x21 which displays the
                          ; character in R0
          LD   R0, TMP_R0
          RET
TMP_R0    .FILL 0
          .END
```

b. Is the above subroutine **PUTCH** a callee-save or caller-save subroutine? Explain. **(1 Point)**

c. Given the following initial values of registers, what are the values of the registers after the execution of an instruction at address x5050: **JSR PUTCH**; and before the execution of the first instruction of the subroutine. **(2 Points)**

Register	**Initial**	**Final**
R0	0x5010	
R4	0x5050	
R7	0x5010	
PC	0x5050	

Problem 6: I/O **(3 Points)**

Let us monitor the contents of the KBSR (Keyboard Status Register), KBDR (Keyboard Data Register), DSR (Display Status Register) and DDR (Display Data Register) during the execution of **TRAP x23 (IN)** in LC-3. The leftmost bit of the block is the MSB and the rightmost bit is the LSB of the registers. **Note: TRAP x23 (IN)** prints prompt to console, read and echo a character from the keyboard.

Below fill in the contents of the different registers at the different steps b, c, and d during the execution of the trap handler for **TRAP x23**.

a. Initial State:

KBDR

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KBSR

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b. The user types in character **“V”** on the keyboard, but the character is not read.

KBDR

KBSR

c. The character **“V”** is read from the keyboard and no new character is typed.

KBSR

d. The display is ready but the character is not yet written to the Display Data Register.

DSR

Problem 7: General Questions (5 Points)

Answer the following short answer questions using **1-2** sentences.

a. What do labels represent in an LC-3 assembly program? **(1 Point)**

b. What is the difference between Memory Mapped I/O and Special I/O instructions? **(2 Points)**

c. Why are two passes required during the assembly process? **(1 Point)**

d. What is the difference between a subroutine call and a branch instruction? **(1 Point)**

Scratch page. You do not need to turn this page in.

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.

```
 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
```

ASCII Table

Character	Hex	Character	Hex	Character	Hex	Character	Hex
nul	00	sp	20	@	40	`	60
soh	01	!	21	A	41	a	61
stx	02	“	22	B	42	b	62
etx	03	#	23	C	43	c	63
eot	04	$	24	D	44	d	64
enq	05	%	25	E	45	e	65
ack	06	&	26	F	46	f	66
bel	07	‘ *(Apostr.)*	27	G	47	g	67
bs	08	(	28	H	48	h	68
ht	09	)	29	I	49	i	69
lf	0A	*	2A	J	4A	j	6A
vt	0B	+	2B	K	4B	k	6B
ff	0C	, *(Comma)*	2C	L	4C	l	6C
cr	0D	-	2D	M	4D	m	6D
so	0E	. *(Period)*	2E	N	4E	n	6E
si	0F	/	2F	O	4F	o	6F
dle	10	0	30	P	50	p	70
dc1	11	1	31	Q	51	q	71
dc2	12	2	32	R	52	r	72
dc3	13	3	33	S	53	s	73
dc4	14	4	34	T	54	t	74
nak	15	5	35	U	55	u	75
syn	16	6	36	V	56	v	76
etb	17	7	37	W	57	w	77
can	18	8	38	X	58	x	78
em	19	9	39	Y	59	y	79
sub	1A	:	3A	Z	5A	z	7A
esc	1B	;	3B	[	5B	{	7B
fs	1C	<	3C	\	5C	|	7C
gs	1D	=	3D	]	5D	}	7D
rs	1E	>	3E	^	5E	~	7E
us	1F	?	3F	_ *(Undrscre)*	5F	del	7F