CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Professor Karthikeyan Sankaralingam
TAs: Suriyha Balaram Sankari, Kamlesh Prakash, Rebecca Lam, Newsha Ardalani, and Yinggang Huang

Examination 3
In Class (50 minutes)
Wednesday, Mar 28, 2012
Weight: 17.5%

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

This exam has 10 pages, including a blank page at the end. Plan your time carefully, since some problems are longer than others. You must turn in pages 1 to 8.

LAST NAME: _____

FIRST NAME: _____

SECTION: _____

CAMPUS ID# _____

EMAIL ID _____

Question	Maximum Points	Points
1	3	
2	3	
3	4	
4	6	
5	4	
6	6	
7	4	
Total	**30**	

Problem 1 (3 Points)

a) What is the **minimum value** we can represent as an immediate value within an LC-3 **ADD** instruction?

-(2^4) = -16

b) Write **ONE** LC-3 machine language instruction that clears the contents of **R6**.

0101 110 110 1 00000 (AND R6, R6, #0)

c) Explain when the following instruction differs from a NOP (NO OPERATION)

0001 0110 1110 0000 Add R3, R3, #0

Differs from a NOP in that it sets the Condition codes.

Problem 2 (3 Points)

Please enter the missing values in the following LC-3 machine language program to implement a **2 input OR**- function. Assume that the 2 inputs are stored in registers R0 and R1. The final output should be stored in register R2. (Adding comments to each machine language instruction will assist in awarding partial credit).

Instruction	Comments
1001 0000 0011 1111	R0 <- NOT(R0)
1001 0010 0111 1111	R1 <- NOT(R1)
0101 0100 0100 0000 (OR) 0101 0100 0000 0001	R2 <- R1 (AND) R0
1001 0100 1011 1111	R2 <- NOT(R2)

Problem 3 (4 Points)

For the following LC-3 program, answer the questions below.

Address	Instruction	Comment
0x3000	0101 0100 1010 0000	R2 <- 0
0x3001	0001 0010 0111 1110	R1 <- R1 - 2
0x3002	0001 0010 0111 1011	R1 <- R1 - 5
0x3003	0101 0010 0100 0001	R1 <- R1 AND R1
0x3004	0000 0000 0010 1101	NOP
0x3005	0001 0010 0111 1111	R1 <- R1 - 1
0x3006	0000 1000 0000 0010	BRn 0x3009
0x3007	0001 0100 1010 0001	R2 <- R2 + 1
0x3008	0000 1111 1111 1000	BRnzp 0x3001
0x3009	1111 0000 0010 0101	HALT

a) What is the **minimum** possible initial value of **R1** that cause the final value in **R2** to be **4**?

32

b) If the initial value of **R1** is **0x41**, what is the final value of **R2**?

8

c) If the initial value of **R1** is 0x5, what is the final value of **R2**?

0

Problem 4 (6 Points)

We are about to execute the following program.

Address	Instruction	Comment
0x3000	0010 0000 0000 0110	LD R0, 0x6
0x3001	0110 0010 1101 1101	LDR R1, R3, 0x1D
0x3002	1010 0100 1111 0000	LDI R2, 0xF0
0x3003	1110 0110 0000 1010	LEA R3, 0xA
0x3004	1111 0000 0010 0101	HALT

What are the final values of R0, R1, R2, R3 after the execution of program? The state of the machine before the program starts is given below. (Adding comments to each machine language instruction will assist in awarding partial credit).

Register	Initial Contents	Final Value (in HEX)
R0	0x200E	0xAAA0
R1	0x200E	0x2012
R2	0x3001	0x1331
R3	0x3001	0x300E

Address	Memory Contents	Address	Memory Contents
0x158E	0x0012	0x300E	0x92FE
0x200E	0x3258	0x301D	0x2121
0x2257	0x0000	0x301E	0x2012
0x2FFF	0x4567	0x30F3	0x3111
0x3002	0xA4F0	0x3111	0x1331
0x3006	0xABEB	0xABCD	0x1580
0x3007	0xAAA0	0xABDB	0x0001

Problem 5 (4 Points)

We are about to execute the following program.

Address	Instruction	Comment
0x3000	1110 0000 0000 0000	LEA R0, 0
0x3001	1110 1101 1111 1110	LEA R6, -2
0x3002	0000 1111 1111 1111	BR nzp 0x3002
0x3003	1111 0000 0010 0101	HALT

Determine the values of PC, R0 and R6 at the end of each successive instruction. The values in PC, R0 and R6 are initially as shown in the table below (i.e. before the start of the execution of the first instruction).

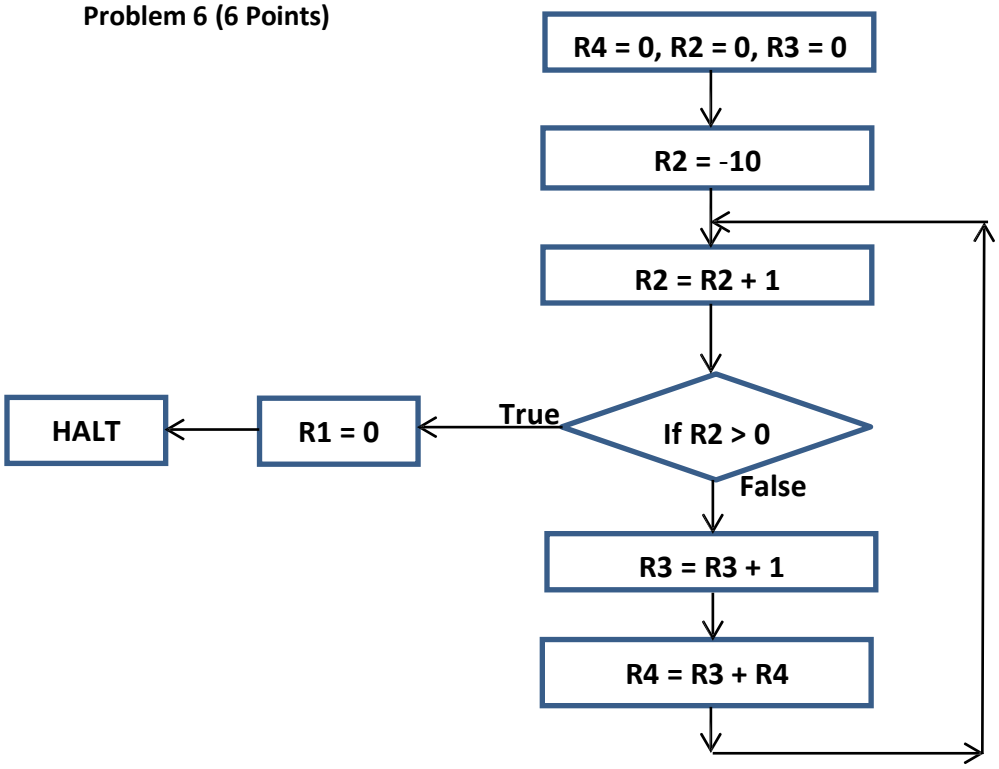
a) Fill in the missing values in the table below. (Adding comments to each machine language instruction will assist in awarding partial credit).

Instruction Count	PC	R0	R6	Next PC
1	0x3000	0xFEED	0x3004	0x3001
2	0x3001	0x3001	0x3004	0x3002
3	0x3002	0x3001	0x3000	0x3002

b) Is there a problem with the instruction at address 0x3002? Explain why or why not.

Infinite loop of executing the branch instruction at 0x3002 and the program never halts.

Problem 6 (6 Points)



a) Please enter the missing values in the table in accordance with the flow chart.

Address	Instructions	Comments
0x3000	0101 1001 0010 0000	Clearing the contents of R4
0x3001	0101 0100 1010 0000	Clearing the contents of R2
0x3002	0101 0110 1110 0000	Clearing the contents of R3
0x3003	0001 0100 1011 0110	R2 <- R2 - 10
0x3004	0001 0100 1010 0001	Incrementing contents of R2
0x3005	0000 0010 0000 0011	BRp 0x3009
0x3006	0001 0110 1110 0001	Incrementing contents of R3
0x3007	0001 1001 0000 0011 (OR) 1000 1100 0100	R4 <- R4 + R3
0x3008	0000 1111 1111 1011	BRnzp 0x3004
0x3009	0101 0010 0110 0000	Clearing the contents of R1
0x300A	1111 0000 0010 0101	HALT

b) What is the final value stored in register R4 after the completion of the above program?

55

Problem 7 (4 Points)

An LC-3 program starts execution at x3000. During the execution of the program, a snapshot of all 8 registers were taken at different times as shown below: before the program starts execution, after execution of instruction 1, after execution of instruction 2, after execution of instruction 3 and after execution of instruction 4. If we keep track of all values loaded into the MAR, MDR and PC as the program executes, we will get a sequence that starts as shown in the tables. Such a sequence of values is referred to as a trace.

Hint: Note that the first PC Trace entry matches with the first MAR Trace entry.

Fill in the missing entries in all the 3 tables below. (Indicated by the eight grey shaded boxes)

Registers	Initial Value	After 1st Instruction	After 2nd Instruction	After 3rd Instruction	After 4th Instruction
R0	0x4006	0x4050	0x4050	0x4050	0x4050
R1	0x5009	0x5009	0x5009	0x5009	0x5009
R2	0x4008	0x4008	0x4008	0x4008	0x4008
R3	0x4002	0x4002	0x8005	0x8005	0x8005
R4	0x4003	0x4003	0x4003	0x4003	0x4003
R5	0x400D	0x400D	0x400D	0x400D	0x400D
R6	0x400C	0x400C	0x400C	0x400C	0x400C
R7	0x6001	0x6001	0x6001	0x6001	0x400E

PC Trace (in HEX)
0x3000
0x3001
0x3002
0x400D

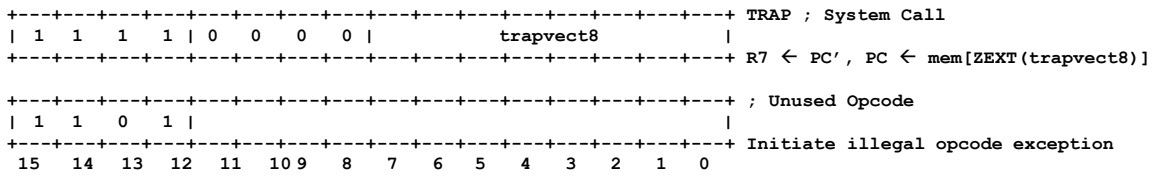
MAR Trace	MDR Trace
0x3000	0xA009
0x300A	0x3025
0x3025	0x4050
0x3001	0x1703
0x3002	0xC140
0x400D	0x4040

SCRATCH PAGE:

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