CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Gurindar S. Sohi

TAs Ryan Johnson, Ramachandran Syamkumar, and Maheswaran Venkatachalam

Midterm Examination 3

In Class (50 minutes)

Friday, April 9, 2010

Weight: 17.5%

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

The exam has **five** two-sided pages. **Circle your final answers.**

The **last** page has the **LC-3 Instruction Set.** It is **detachable.**

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

NAME: _____

SECTION: _____

ID# _____

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

Problem 1 (5 points)

a. (2 points) What is the largest magnitude negative number that we can represent as an immediate value within an LC-3 AND instruction?

   -2^4 = **- 16 decimal**

b. (3 points) Assume that a BRANCH (opcode 0000) instruction is present in memory location x3000. What is the range of the memory addresses that this instruction can branch to?

   Lowest memory location it can branch to = x3001 – x0100 = **x2F01**

   Highest memory location it can branch to = x3001 + x00FF = **x3100**

Problem 2 (3 points)

Many Instruction Set Architectures (ISAs) have an instruction called NOP (No Operation). A NOP instruction just increments the PC, while leaving the current state of the system unchanged (registers, memory, and condition codes are not modified). Provide a single LC-3 instruction which is functionally equivalent to a NOP. You must provide the 16 bit machine code for your instruction and not just the name of the instruction.

We can use the unconditional branch instruction for this purpose:

Binary 0000 111 000000000

Problem 3 (5 points)

The PC contains x3010. The following memory locations contain values as shown:

Memory Location	Contents
x3014	x712A
x712A	x712B
x712B	x0112
x712C	x824C

a. (3 points) The following LC-3 instructions are then executed, causing a value to be loaded into R6. What is that value?

Memory Location	Contents
x3010	1110 1000 0000 0011
x3011	0110 1011 0000 0000
x3012	0110 1101 0100 0000

LEA R4, x3014; R4 ← x3014

LDR R5, R4, x0000; R5 ← Mem[x3014] = x712A

LDR R6, R5, x0000; R6 ← Mem[x712A] = **x712B**

b. (2 points) We could replace the three-instruction sequence (in part a) with a single instruction (at memory location x3010). What is it? (Show the 16 bits of the instruction)

We could do this using an LDI instruction.

1010 110 000000011

Problem 4 (4 points)

The following table shows a part of LC-3 memory:

Memory Location	Contents
x3100	1001 001 001 111111
x3101	0001 010 000 000 001
x3102	1001 010 010 111111
x3103	0000 010 111111100

State what is known about R1 and R0 (before the execution of the program) if the conditional branch in x3103 redirects control of the program to location x3100.

R1 ← not R1

R2 ← R0 + R1

R2 ← not R2

The branch is taken implies that the the result of the third instruction was zero. We then just work backwards in an attempt to the find the relationship between R1 and R0. So R2 was x0000 at the end of execution of instruction 3. So R2 was xFFFF at the end of execution of instruction 2. This implies that R0 + R1 = xFFFF at the end of execution of instruction 1. i.e., R0 and R1 are NOTs of each other at the end of instruction 1. Thus, **R0 = R1** at the beginning of the program.

Problem 5 (8 points)

We are about to execute the following program:

Memory Location	Contents
x3000	0111 001 010 000101
x3001	1011 010 000000010
x3002	0110 011 001 000001
x3003	1010 100 000000000

The state of the machine before the program executes is given below:

Register R1: x4000

Register R2: x5000

Memory Location	Contents
x3004	x6000
x3005	x6001
x4000	x123A
x4001	x123B
x6000	x8000
x6001	x8001

a. (2 points) The instruction in location x3000 updates a certain memory location X with a certain value Y. What are the values of X and Y?

   X = R2 + 5 = **x5005**; **Y** = R1 = **x4000**

b. (2 points) The instruction in location x3001 updates a certain memory location X with a certain value Y. What are the values of X and Y?

   X = Mem[x3002 + x0002] = Mem[x3004] = **x6000**; **Y** = R2 = **x5000**

c. (2 points) The instruction in location x3002 updates a certain register X with a certain value Y. What are the values of X and Y?

X = R3; **Y** = Mem[R1 + x0001] = Mem[x4001] = **x123B**

d. (2 points) The instruction in location x3003 updates a certain register X with a certain value Y. What are the values of X and Y?

X = R4; Y = Mem[Mem[x3004]] = **x5000**

Problem 6 (5 points)

a. (2 points) Write an LC-3 instruction that clears the contents of R2.

0101 010 010 1 00000; AND R2, R2, x0000

b. (3 points) The following program increments R0 by 1, if R1 > R2. Fill in the missing instruction.

Memory Location	Contents
x3000	1001 010 010 111111
x3001	0001 010 010 1 00001
x3002	0001 010 001 000 010
x3003	_____
x3004	0001 000 000 1 00001
x3005	1111 0000 0010 0101 ; HALT

0000 110 000000001

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

LC-3 Instruction Set

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
```