CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Mark D. Hill

TAs: Sujith Surendran, Pradip Vallathol

Midterm Examination 3

In Class (50 minutes)

Monday, Nov 18, 2013

Weight: 17.5%

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

The exam has 8 pages. **Circle your final answers**. Plan your time carefully since some problems are longer than others. You **must turn in the pages 1-7**. Use the blank sides of the exam for scratch work.

Note: LC-3 instruction set is provided on the Last Page

LAST NAME: _____

FIRST NAME: _____

ID# _____

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

Problem 1 **(2 Points)**

Assume that you wrote a program to print "CS/ECE 252 Rocks!!" on the screen. But when you executed it, you saw that none of the characters were displayed. While debugging, you find that you had wrongly placed an instruction just before printing "CS/ECE 252 Rocks!!". The instruction you placed is:

```
0000 111 111111111
```

Now, can you explain why "CS/ECE 252 Rocks!!" did not get printed on the screen?

The instruction is "*Branch to PC'-1*". So this results in an infinite loop and the instructions below this never gets executed. So "CS/ECE 252 Rocks!!" never gets printed on the screen.

Problem 2 **(2 Points)**

Your friend Chandler has written a code which has the following two instructions. Since you are smarter than him, write one instruction (in hexadecimal) which has the same effect as the combination of the following two instructions:

```
0010 011 000000011  (R3 ← Mem[PC'+3])
0111 100 011 000000 (Mem[R3 +0] ← R4)
```

Ans: Store indirect (`1011 100 000000011`) = 0xB803

Problem 3 **(5 Points)**

The following is a code snippet from a program which your best friend Ross has written for his class project:

Address	Instruction	Comments
0x3000	0110 100 011 000001	(R4 ← Mem[R3+1])
0x3001	0001 101 101 000 100	(R5 ← R5 + R4)
0x3002	0001 011 011 1 00001	R3 ← R3 + 1
0x3003	0001 010 010 1 11111	R2 ← R2 – 1
0x3004	0000 101 111111011	(If(R2!=0)branch to 0x3000)
0x3005	1111 0000 0010 0101	Halt

a) **(3 Points)** As you can see, Ross has not commented few of the instructions. Write comments for these uncommented instructions.
Note: LC-3 instruction set is provided on the last page

b) **(2 Points)** Suppose Ross also told you that the initial values of the memory locations 0x3100 to 0x3103 before executing the code are as shown below. He has also informed you that the values of **R3 = 0x3100, R2 = 0x2, and R5 = 0** before executing the code. Now, he asks you what the value of **R5** is after running the code. What would be the correct answer?
Note: Show your work for partial credit.

Address	Initial Value
0x3100	0000 0000 0000 0001
0x3101	0000 0000 0000 0010
0x3102	0000 0000 0000 0011
0x3103	0000 0000 0000 0100

Answer = Mem[0x3101] + Mem[x3102] = 2 + 3 = 5

Problem 4 **(5 Points)**

Assume that you are asked to work in a team and create a program for class assignment and you teamed with Rachel this time. The program you both are supposed to create should load a value from memory location **x2012** into **R1** if the value in register **R1** is **odd**. If R1 is **even**, then the value at memory location **x2010** should be stored into **R1**. The program should start at memory location x2000. Luckily, Rachel has written comments for all instructions. She has also written one instruction all by herself. Complete the missing instructions of the program.

Address	Instruction	Comments
0x2000	0101 001 001 1 00001	Generate condition if value in R1 is odd
0x2001	0000 001 000000010	Branch to 0x2004 if condition is true
0x2002	0010 001 000001101	R1 ← Mem[0x2010]
0x2003	0000 111 000000001	Branch to HALT (0x2005)
0x2004	0010 001 000001101	R1 ← Mem[0x2012]
0x2005	1111 0000 0010 0101	HALT

Problem 5 **(2 Points)**

Assume that you need to store one of your favourite numbers, **0x3040**, into R1 using an instruction placed at 0x3000. However, your friend Phoebe tells you that it is impossible to put 0x3030 into R1 using a single instruction. Do you agree with her? If you agree, give a reason why it is not possible to do this. If you do not agree, then write the instruction **(in hex)** which stores the value 0x3030 into R1 using just one instruction placed at 0x3000.

Note: You cannot assume the values of any of the registers or memory locations.

Ans: 1110 001 000111111 (LEA PC'+ x3040 – x3001) = 0xE23F

Problem 6 **(6 Points)**

Assume that the initial value at R3 is `1111111111111111`. Using only **one AND instruction**, mention if it is possible to change this value to any of the following values (given below). If yes, give the instruction **(in hex)** which will cause this change. If not, argue why this cannot be done.

Note: You cannot assume the values of any other register.

a. `0000000000001111`

Yes. The instruction is 0101 011 011 1 01111 (ie, AND R3,R3,#15) = 0x56EF

b. `1111111111110000`

Yes. The instruction is 0101 011 011 1 10000 (ie, AND R3,R3,-16) = 0x56F0

c. `1111111111100000`

No. This instruction cannot be executed in one instruction because we are able to specify only 5 bits for AND immediate instruction and it sign extends the rest. However, for generating this instruction, we need atleast 6 offset bits and sign extention of the 6^{th} bit.

Problem 7 **(8 Points)**

Assume that you finally decided to write a program alone for your 2nd class project. After you successfully ran the code and found it to be working, you showed it to your friend Monica for suggestions. Monica suggested that you **replace** 4 sets of instructions (which is just a rearranged version of your code, but she finds that "better"!). Your code as well as Monica's suggestions is shown below. For each of these sets, specify if the code would produce the same effect if your code was replaced with Monica's suggestion. Provide reasons to support your answer in the comments section. Assume that **R3=1, R4=2** before executing each of these sets. **Make no assumptions about any other registers or memory locations**.

Set #	**Your code**	**Monica's suggestion**
1	0101 100 100 1 00000 (R4 ← R4 AND 0) 0101 011 011 1 00000 (R3 ← R3 AND 0) 0000 010 000000011 (Branch if Z to PC'+3)	0101 011 011 1 00000 (R3 ← R3 AND 0) 0101 100 100 1 00000 (R4 ← R4 AND 0) 0000 010 000000011 (Branch if Z to PC'+3)
	Comments: Yes, both these codes generate the same effect since both of them puts values of R3 and R4 to 0. Since the condition codes before branch is same in both the cases, the branch also gets executed in both the cases. So the result is the same for both the codes.	
2	0001 100 100 1 11111 (R4 ← R4-1) 0001 011 011 1 11111 (R3 ← R3-1) 0000 010 000000011 (Branch if Z to PC'+3)	0001 011 011 1 11111 (R3 ← R3-1) 0001 100 100 1 11111 (R4 ← R4-1) 0000 010 000000011 (Branch if Z to PC'+3)
	Comments: No, both these codes behave differently. This is because for the first code, the branch will be taken because R3=0. But for the second code, the branch will not be taken because R4=1	
3	0001 100 100 1 11111 (R4 ← R4-1) 0011 011 000000011 (R3 ← Mem[PC'+3])	0011 011 000000011 (R3 ← Mem[PC'+3]) 0001 100 100 1 11111 (R4 ← R4-1)
	Comments: No, both these codes behave differently. This is because PC' is different for these codes. So, the addresses from where the load happens is different for both these pairs.	
4	0001 100 011 1 11111 (R4 ← R3-1) 0111 011 011 000011 (R3 ← Mem[R3+3])	0111 011 011 000011 (R3 ← Mem[R3+3]) 0001 100 011 1 11111 (R4 ← R3-1)
	Comments: No, because values of R4 will be different in both the cases.	

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.