CS 270 Spring 2012 Midterm

8 March 2012, 9:30 pm

Name _____

Please read these instructions completely before proceeding. Then, sign below to indicate that you have understood them. Your exam will not be graded without your signature.

- This is a closed book exam, but you are allowed to bring one page (one single side) of notes. The last sheet of this exam has the LC-3 opcodes (the back cover of the text) and the datapath of the LC3 (page 142). You are free to tear out this page and use it as a reference—no need to turn it in with your exam.
- This midterm will last 75 minutes. The total score is 85, the weight of each problem corresponds roughly to the time you should spend on it.
- You are allowed to use only paper/pen and your brain—no calculator, laptop, phone, ipod, or any electronic device. Please turn off cellphones, and please refrain from using any listening device (music, etc.) You shouldn't be wearing earphones unless they are for a medical reason.
- Answer all questions. The exam is designed so that the average score is about 57 (66.7%). Do not be discouraged if you cannot answer all the questions.
- Do not turn this page until you are asked to.

I have read and understood the above instructions. I promise to do the exam honestly and fairly.

Signature _____

Problem 0: Plan of Attack [5 pts]

Quickly read through the exam and make a plan of attack. For each question, think about what skills it's testing for, how comfortable you feel, and rate its difficulty level for you. Based on this, fill up the PoA column (the order in which you plan to answer the questions, and the time you will spend on each one) in the table below. Don't fill up the last two columns as yet.

Don't write in these columns				Plan of Attack		Revised PoA	
Prob.	Topic	Max	Score	PoA	Time	PoA	Time
0	Plan	5	5	0	5 mins		
1	Numbers & Data	15					
2	Gates/Combinational Circuits	15					
3	Sequential Circuits, FSMs	10					
0.b	Revised PoA	5					
4	C Programming	15					
5	LC-3 Architecture	20					
	Total	85					

Problem 1: Numbers and Data [15 pts total]

The first two problems deal with numbers less than 1, i.e., with a "radix point." Give your answers in this form, not as a fraction.

Part a: Convert 0.714_8 to base 10. [4 pts]

Part b: Convert the decimal fraction 0.52734375_{10} to base 4. [4 pts]

Part c: Convert 20.8984375 to a 16-bit floating point (half precision) number. [Hint: **Part a** may help you] [4 pts]

Part d: Adding floating point numbers in scientific notation. [3 pts]

(i) Add 7.2177 E 12 and −7.2217 E 12

(ii) Add 7.6217 E 12 and 7.2177 E 11

Problem 2: Gates and Combinational Circuits [15 pts total]

a. We want to design a comparator for 8-bit unsigned binary numbers, but we want the design to work from "left to right," by chaining up 8 cells. Each cell has inouts X_i and Y_i, the two input bits and also two inputs signals from the left (higher bits) called D_{in} and R_{in}. It produces two signals to the next cell on the right called D_{out} and R_{out}. So the truth table of the cell has 16 entries, but there are many simplifications.

Din	Rin	X	Y	Dout	Rout
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

If $D_{in} = 1$, then the decision (about which is the larger of X and Y, has already been made, and the signal R_{in} (R of result) tells this result (1 means that $X > Y$). The cell should just propagate this information to the right. So the bottom half of the truth table has been filled up for you. But if $D_{in} = 0$, then the cell has the opportunity to make a decision, based on the values of the other inputs. Fill up the remainder of the

truth table for the case when $D_{in} = 0$. [7 pts]

ii. Consider the following truth table that implements a two-input Boolean function. Draw the circuit for implementing this function using only NOR gates (partial credit for an otherwise correct implementation using other gates). [8 pts]

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Problem 3: Finite State Machines [10 pts total]

Draw the state diagram of a controller for a vending machine with the following beavior. It delivers some candy (output 1) when the user inserts coins (nickels, dimes or quarters) that add up to 25 cents. It accepts three inputs (N, D, Q). If the user inserts extra money, it delivers candy (produces an output) but does not return change. Rather, it uses the extra money as credit towards the next purchase.

Problem 0.b: Revised PoA [5 pts]

This is a 5 minute, strategy/stretch break. First, take a quick 1-minute break. Close you eyes, calm down, breathe deeply and relax. Get up and physically stretch and try to ease tension. Make eye contact with your friends and smile. Look at Sanjay and scowl. Now, revisit your plan. Draw a line through all problems that you have finished, and revise the plan as needed. Budget the remaining time appropriately.

Problem 4: C programming [15 pts total]

Complete the following piece of C code, to do the extra credit part of PA2. It extracts the sign, exponent and fractional parts of the two half-precision numbers, x and y, and computes their sum in answer.

```
uint16_t x, y, answer;  //answer should contain the sum of x and y
int a1, a2, b1, b2, v1, v2, temp; //
a1 = x & (1<<15);
a2 = y & (1<<15);
b1 = (x & (31<<10)) >> 10;
b2 = (y & (31<<10)) >> 10;
v1 = (x & 1023); //Note: this has only 10 bits: implicit 1 is still missing
v2 = (y & 1023); //Note: this has only 10 bits: implicit 1 is still missing

if (a1==a2 && b1>b2) {// x and y have the same sign, but x has bigger exponent
// ignore this case
}
else if (a1==a2 && b2>=b1) {// same sign but other case of exponent
{   // Study thia code and understand the logic (including renormalization)
    result = (v2 | 1024) + ((v1 | 1024) >> (b2-b1));
    if (result | 1 << 11){
        result = (result >> 1) & 1023;
        answer = a1 | (b2+1) << 10 | result;
    }
    else {
        result = result & 1023;
        answer = a1 | b2 << 10 | result;
    }
}



//  Study the code above and then finish the rest of it on the next page

}
```

Complete the case when the signs are different.

```
if (a1 != a2)
{  //   FIXME: First Identify the different subcases
   //  Then handle one of them completely
   //  Finally, indicate how the others are handled









}
```

Problem 5: LC-3 [20 pts total]

Remember that the LC-3 has three types of instructions: operate (e.g., ADD), load/store (that transfer data to/from a register from/to a memory address), and control (BR, JMP). When we ask you to write one or more LC-3 instructions, we first want you to clearly describe *what* the instruction is doing in a comment, then we want the answer in binary/hex. For example, 0x1DAA, which is 0001,1101,1010,1010 is an instruction that adds the immediate value 10 to R6 and stores the result back in R6. You should write this as follows (the comment on the right in the register transfer notation is more important than the hex)

```
0x1DAA; Comment:  R6 <- R6 + 10
```

Part a. We want you to write a "program:" a sequence of operate instructions, to compute the bit-wise OR of the contents of R3 and R4, and store the result in R0. No other registers should be affected by your program. First, describe your thinking in terms of comments in register transfer notation, and then write the actual binary/hex code. [10 pts]

Part b. At a certain moment in time, the PC of an LC-3 computer contains, 0x2000, in the memory, at address 0x2000, is a word whose most significant 4 bits are 0001 (opcode ADD). The clock goes through 4 cycls so that the instruction in 0x2000 is executed. The values in the 7 registers before and after the execution are as follows.

	R0	R1	R2	R3	R4	R5	R6	R7
Before	0x0000	0x1111	0x2222	0x3333	0x4444	0x5555	0x6666	0x7777
After	0x0000	0x1111	0x2222	0xFFF2	0x4444	0x5555	0x6666	0x7777

Based on this information, deduce the contents of memory at address 0x2000 (the complete instruction). Describe your thinking is you want to be considered for partial credit. [10 pts]