CS 270 Fall 2011 Midterm

13 October 2011, 12:30 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 in 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 75, 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 50 (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 columns on in the table below. Don't fill up the last three columns as yet.

Don't write in these columns				Plan of Attack			Revised PoA		
Prob.	Topic	Max	Score	PoA	Start	End	PoA	Start	End
0	PoA	5	5	0	12:30	12:35			
1	Numbers & Data	15							
2	Gates/Comb Ckts	15							
3	Finite State Machines	10							
0.b	Revised PoA	5							
4	C Programming	15							
5	LC-3 Architecture	15							
	Total	85							

Problem 1: Numbers and Data [15 pts total]

Perform the following base conversions (show all our work if you want to be considered for partial credit):

a. $4626_7 = ????_5$ [3 pts]

b. $47657_8 = ????_4$ (there's a simple trick to do this problem, that you'll know it if you have practiced enough; if you use/explain this trick, you will get extra credit). [2 pts]

c. The table below shows two 16-bit values representing half-precision floating point numbers (A on line 1, and B on line 2). You are to add them up. Answer the following questions if you would like to get partial credit. [10 pts]

i. Which number (A, B or neither) is to be shifted? Right or Left? By how much? Justify your answer. [2 pts]

ii. Next, we must (circle the entries in the boxes, and fill in the values of Frac1 and Frac2 in the table below): add / subtract Frac1 to / from Frac2. [3 pts]

iii. Do this and show the partial result on line 5 (the table below has a few extra "workspace" lines for you). [2 pts]

iv. Does the result need to be normalized? yes / no. [1 pt]

v. Write the final answer (normalized as needed) with the appropriate exponent and sign in line 6. [2 pts]

A	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1
B	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	1
Frac1																
Frac2																
Sum/Diff																
Final																

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

a. We want to design an old fashioned subtractor circuit for unsigned binary numbers, with the usual, elementary school, "borrow" method, where we "propagate" a borrow signal from the LSB towards the MSB. We will first design a one-bit subtractor that has input signals, X, Y, and B_{in} and output signals, D (for difference) and B_{out}, to compute $X - Y$, and B denotes "borrow." Normally, X will be greater than Y, but if this is not the case, the b_{out} from the MSB will be 1.

i. Fill up the truth table of this module. [6 pts]

Bin	X	Y	D	Bout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

ii. Draw the circuit(s) for implementing these two functions using only NAND gates (partial credit for an otherwise correct implementation using other gates). [6 pts]

b. Show how an array of such circuits—viewed as "black-box" components—can be connected to build an 8-bit subtractor. For boundary (LSB and MSB) modules, what should be the B_{in} input, and what does the B_{out} denote? [3 pts]

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 3: Finite State Machines [10 pts total]

Consider the pattern-recognizer FSM that you did in HW2, that recognized sequences of the form 2323 (overlapping sequences were allowed). Draw the transition diagram of a similar FSM that recognizes the samme pattern, but this time overlaps are *not* allowed. For example, the input sequence 2302,3232,3232,3101 (commas are to improve readability, not part of the sequence) will produce the output sequence 0000,0010,0010,0000.

a. Draw the state-diagram of this machine, using only four states (you are allowed to have outputs produced during transitions, partial credit if you use more than four but get the behavior right). [5 pts]

b. (C programming) Write a snippet of C code that implements this machine. Your code must use a nested switch construct, the outer one to test the current state and the inner one to test the current input. Use the template below as a guide (the `FIX ME`'s are not the only places that need to be changed). [5 pts]

```
 char nextChar = 0;
 enum state {                                 }; // FIX ME: enumerate states
 enum state curState =                         ; // FIX ME: initial state

 while(nextChar != 'q' && nextChar != 'Q'){
   scanf("%c", &nextChar);
   switch(curState){
   case                                          // FIX ME
     switch(nextChar){
     case //
        printf("OUTPUT =  xxxx \n");             // FIX ME: replace xxxx
        curState = ;                             // FIX ME:
```

```
   default:
        printf("OUTPUT = xxxx \n");             // FIX ME: replace xxxx
        curState = ;                            // FIX ME:

//close braces as appropriate
```

Problem 4: C programming [15 pts total]

Q1. Declare a variable called `value1` of datatype int. Then, in a single statement, declare a variable called `valueP` of datatype *pointer to an int*, and assign the address of variable `value1` to it. [2 pts]

Q2. Given the following C struct:

`struct Multivalue { int intvalue; float floatvalue; char Charvalue;};`

In a single statement, declare a variable `mvarrayD` of type *pointer to struct Multivalue*, and assign to it a dynamically allocated array. This array should hold precisely 10 elements of type `Multivalue`. [3 pts]

Q3. Someone hands you the following snippet of C code. Describe in words what it does (tough question, manage your time). Remember, ">>" and "<<" are shift operators, and "&" and "|" are bit-wise AND and OR operators. [10 pts]

```
int x, y;
int a1, a2, b1, b2, v1, v2;
a1 = x & (1<<15);
a2 = y & (1<<15);
b1 = (x & (31<<10));
b1 = b1 >> 10;
b2 = (y & (31<<10));
b2 = b2 >> 10;
v1 = x & 1023;
v1 = v1 | (1<<10);
v2 = y & 1023;
v2 = v2 | (1<<10);
```

Q4. **Extra Credit:** Right after the above code is a line that starts with the `if` below. Complete this code (don't attempt this unless you got Q3. down pat).

```
if (a1==a1 && b1>b2)
```

Problem 5: Instruction Sets & the LC-3 [15 pts total]

c. Assume that the PC has 0x4000, and at address 0x4000 is an instruction LDI R2 0x24 (i.e., 0xA424, or 1010 010 000100100 in binary). Describe precisely what happens in each of the subsequent clock cycles until the instruction is completely executed (the first four cycles are given to you). [15 pts]

`MAR` ← `PC; Rd` ← `1; MEnable` ← `1;` #issue memory read

`MDR` ← `mem;` #load the value returned from memory (CPU inactive—the memory does this transfer

`IR` ← `MDR; PC` ← `PC+1;` # move instr into IR and increment PC

`DECODE` # The text book has one cycle for this. Based on the result of the decode, the CPU controller transitions to a sequence of states that complete the execution of the instruction. For our `LDI` instruction, what happens next, step by step?

Prob. No.	Topic	Max score	Your Score
1	Numbers	20	
2	Gates/Comb Ckts	15	
3	Seq ckts	10	
4	C	15	
5	LC3 Instruction Set	15	
	Total	75	