ECE190 Exam 1, Spring 2007

Thursday 15 February

Name:

- **Be sure that your exam booklet has 13 pages.**
- **The exam is <u>meant</u> to be taken apart!**
- **Write your name at the top of each page.**
- **This is a closed book exam.**
- **You may not use a calculator.**
- **You are allowed one 8.5×11" sheet of handwritten notes.**
- **Absolutely no interaction between students is allowed.**
- **Show all of your work.**
- **Don't panic, and good luck!**

"I've never said that."
—S. Lumetta, self-quotation

Problem 1	20 points	
Problem 2	14 points	
Problem 3	26 points	
Problem 4	20 points	
Problem 5	20 points	
Total	100 points	

Problem 1 (20 points): Short Answers

Please answer concisely. If you find yourself writing more than a few words or a simple drawing, your answer is probably wrong.

Part A (5 points): You join an intramural sports club consisting of six teams. As the club's Engineer, they ask you to keep track of the relative standings of each team. You need only keep information on the order of the six teams (best, second best, etc.). Assuming that there are no ties, how many bits do you need to represent the relative standings?

Part B (5 points):

A stoplight has four states:

- no lights are on,
- only the red light is on,
- only the yellow light is on, and
- only the green light is on.

Suppose that you are building a finite state machine to control such a light. Explain one possible advantage of using three bits of state rather than the minimal two bits.

Problem 1, continued:

Part C (5 points): Years in the future, your friend suggests that rather than using something complicated like floating-point, your team's ECE411 project should instead design a computer using fixed-point. In particular, your friend suggests a 32-bit fixed-point data type in which a given bit pattern represents its 2's complement value divided by 65536 (2^{16}). For example, the pattern

0000 0000 0000 0000 0000 1111 0000 0000

represents $(15 * 256)/65536 = 0.05859375$. Explain one advantage and one disadvantage of your friend's data type relative to using IEEE single-precision floating-point, which is also a 32-bit data type.

Part D (5 points): Assume that an LC-3 processor's R4 (register #4) holds the value 0. Without making any other assumptions, what is the minimum number of LC-3 instructions necessary to change R4's value to 16. Explain your answer.

Problem 2 (14 points): Combinational Logic

Part A (6 points): Your friend teaches you a great new game for deciding what to do on a Friday night. The game is called "Study-Party-Workout," involves **two players**, and works as follows:

- Each of the two players decides separately whether they want to STUDY, PARTY, or WORK OUT.
- Simultaneously, the two players reveal their choices using hand signals agreed upon in advance.
- If both chose the same activity, they must play again, potentially for the whole evening.
- Otherwise, the winning activity is chosen as follows:
  - STUDY beats PARTY, since it means a better grade and a brighter future.
  - PARTY beats WORK OUT, since it involves less pain.
  - WORK OUT beats STUDY, since it extends your life and gives your more time to STUDY later.

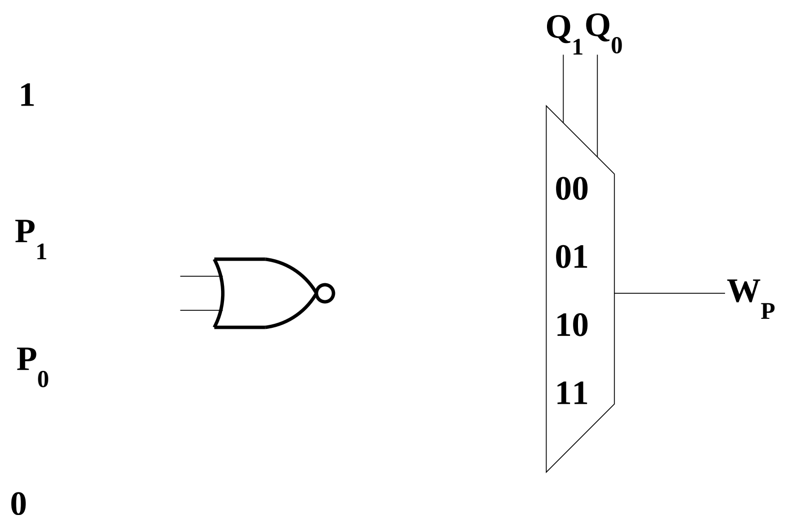
(You may have already heard of a derivative game called "rock-paper-scissors.")

Assuming that we encode STUDY as 00, PARTY as 01, and WORK OUT as 10, fill in the rest of the truth table below. The inputs P and Q represent the player's choices. The W outputs indicate that P or Q has won, and the A output indicates that both have chosen the same activity and must play again.

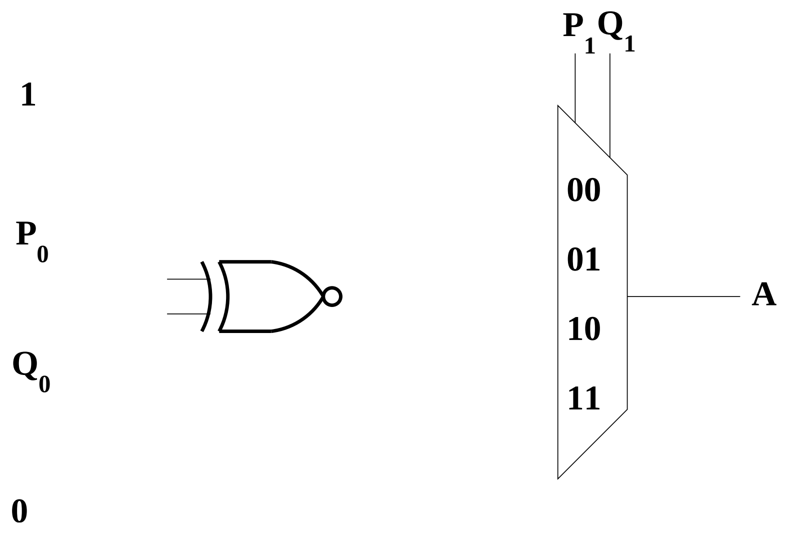
P_1 P_0 Q_1 Q_0	W_P	W_Q	A
0 0 0 0			
0 0 0 1			
0 0 1 0			
0 0 1 1	0	1	0
0 1 0 0			
0 1 0 1			
0 1 1 0			
0 1 1 1	0	0	0
1 0 0 0			
1 0 0 1			
1 0 1 0			
1 0 1 1	0	1	1
1 1 0 0	1	0	0
1 1 0 1	0	0	0
1 1 1 0	1	0	1
1 1 1 1	0	0	1

Problem 2, continued:

Part B (4 points): Using the two-input mux as configured below and the two-input NOR gate, complete the logic to generate the W_P signal. You are allowed to use 0s and 1s (also drawn in the diagram). **However, you may use no additional components or gates, only wires.** *Note that the "don't care" entries of the truth table on the previous page were carefully selected to simplify this logic.*

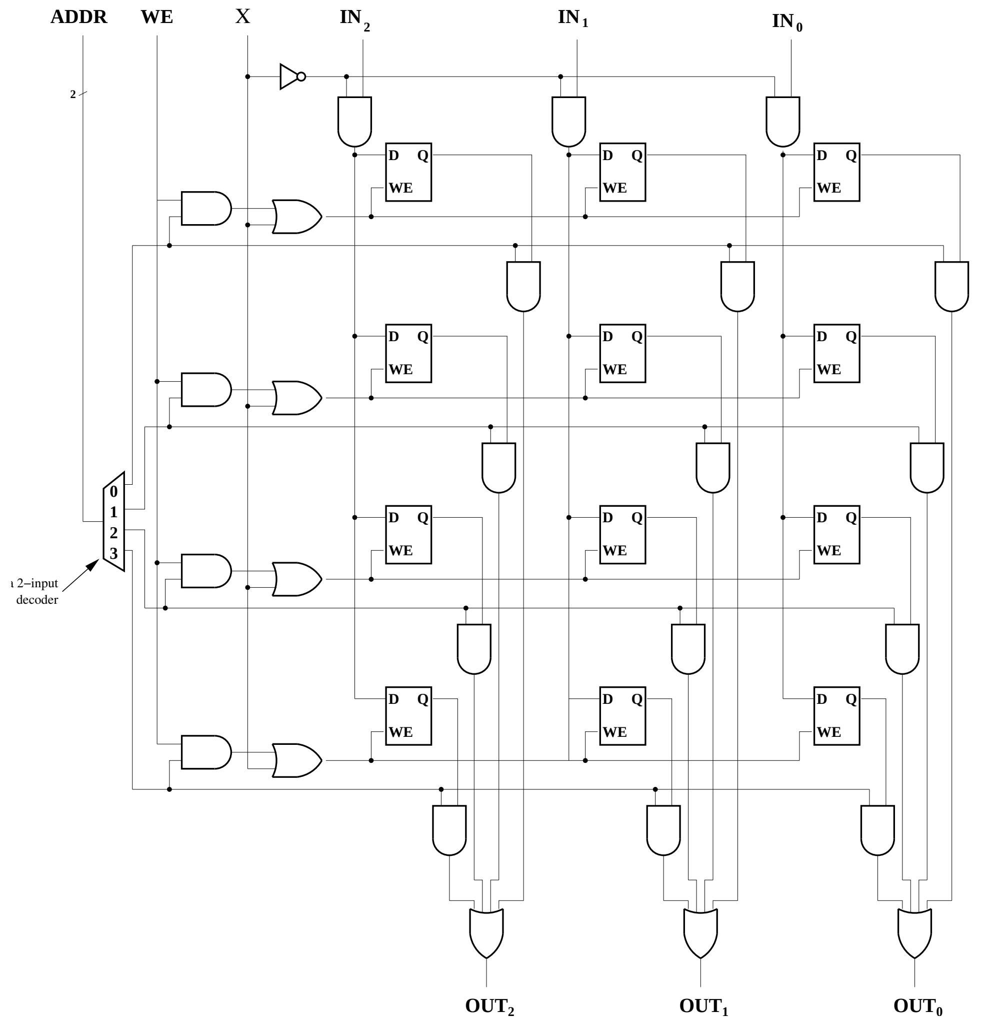


Part C (4 points): Using the two-input mux as configured below and the two-input XNOR gate (XOR followed by NOT), complete the logic to generate the A signal. You are allowed to use 0s and 1s (also drawn in the diagram). **However, you may use no additional components or gates, only wires.** *Note that the "don't care" entries of the truth table on the previous page were carefully selected to simplify this logic.*



Problem 3 (26 points): Memory and Finite State Machines

Consider the 4x3-bit memory shown below. Other than the input X and a few extra gates, it is a standard 4x3 memory. **Parts A** through **C** refer to it.



Part A (4 points): Explain the behavior of the memory when $X = 0$.

Part B (4 points): Explain the behavior of the memory when $X = 1$.

Part C (2 points): Explain the purpose of the input X.

Problem 3, continued:

After his last attempt to produce a layered drink dispenser for EOH, Professor Lumetta has decided to try something simpler. He has just completed the core of a simple soda machine, but needs your help wiring up the controls.

The goal for the dispenser is to accept **nickels** (5 cents), **dimes** (10 cents), and **quarters** (25 cents) towards the purchase of a can of soda. When the machine has received 40 cents or more, it dispenses a can of soda and returns any money over 40 cents. For example, if someone puts in two quarters, one after another, the machine returns a can of soda and 10 cents. The core of the machine is shown below.

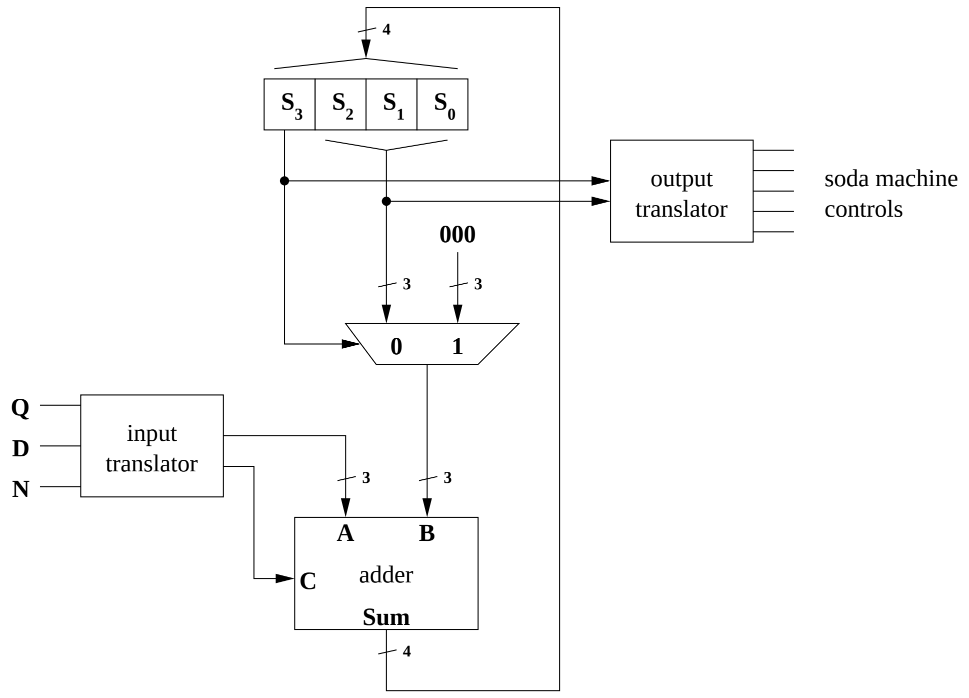
The four-bit register S records the amount of money currently held by the machine as an unsigned number of nickels. For example, if the machine has received 25 cents, $S = 0101$.

The input translator changes the signals for the coin types that have been inserted in a given cycle into a number of nickels to be added into the register S.

The adder is a standard three-bit adder except that the carry output is included as the most significant bit of the sum rather than being separated out. The C input is the carry input for the adder.

The output controller translates the current state S into control signals for dispensing a can of soda and for returning various types of coins.

The trapezoid in the middle of the figure represents three two-input multiplexers.



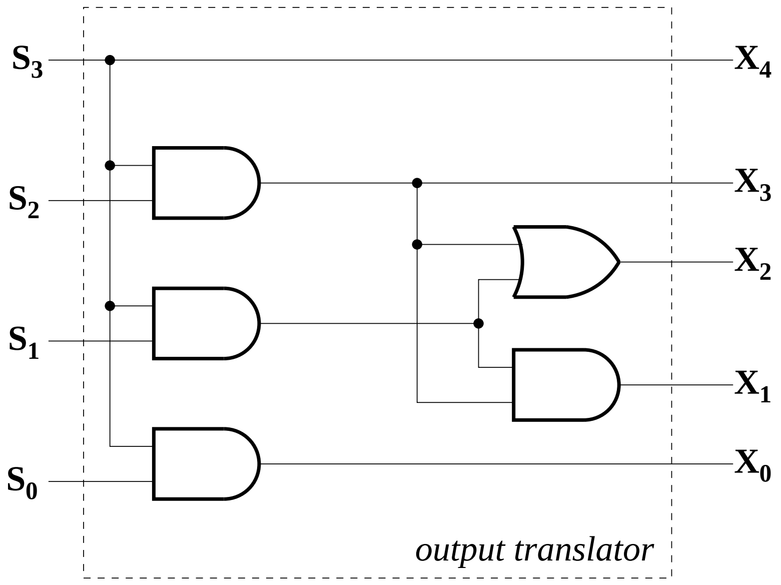
Part D (5 points): Explain the purpose of the multiplexer. *Hint: consider what happens if you repeatedly insert a quarter in every cycle.*

Problem 3, continued:

Part E (6 points): Prof. Lumetta has run out of components! Design the input controller for him using only wires. The Q input indicates that a quarter was inserted. The D input indicates that a dime was inserted. The N input indicates that a nickel was inserted. Any combination is possible in a cycle.

Q	A_2
D	A_1
N	A_0
	C

Part F (5 points): One of Prof. Lumetta's graduate students was kind enough to design the output controller for him (shown in the figure below). Unfortunately, the student did not produce documentation. The inputs from the FSM state S are labeled, but Prof. Lumetta is not sure what the X outputs are supposed to do. One of them, of course, must release the can of soda. The other four dispense coins, but what kind of coins (nickel, dime, or quarter)?



For each output, explain what it means by writing one of the following four words next to the output: SODA, NICKEL, DIME, QUARTER. *Hint: you may want to draw a truth table to help you.*

Name: ____

Problem 4 (20 points): The Von Neumann Model

Part A (8 points): Based on the contents of memory and registers shown at the bottom of the page, fill out the RTL in the table below for the four instructions beginning at x4000. All four instructions are some type of *load* instruction. A sample answer might appear as R3 ← M[PC-5] or R6 ← M[R2-7].

Address	RTL
x4000	
x4001	
x4002	
x4003	

The LC-3 is about to start the instruction cycle for the instruction at x4000. Update the diagram below to reflect the state of the machine following the execution of *four full instruction cycles*. Clearly denote the final values in the following components within the diagram:

Part B (8 points): Register File

Part C (2 points): IR and PC

Part D (2 points): MAR and MDR

	Memory		Register File
x4000	x2203	R0	x4007
x4001	xA402	R1	x0001
x4002	xE800	R2	x1337
x4003	x663F	R3	xCAFE
x4004	x4006	R4	x4004
x4005	x4001	R5	x4005
x4006	xF00D	R6	x4006
x4007	xBEEF	R7	xFFFF

MAR		**PC**	x4000
MDR		**IR**	

Problem 5 (20 points): LC-3 Instructions

Part A (3 points): Given an ST (store) instruction at address x5000, what is the smallest memory address that the store can change? What is the largest?

Part B (4 points): Consider the following code snippet (RTL on the right):

```
ADD   R2, R1, R0   ; R2 ← R1 + R0
ADD   R2, R2, #1   ; R2 ← R2 + 1
BRz   #10          ; if Z: PC ← PC + 10
```

If the branch is taken (*i.e.*, if the PC changes due to the branch), what do you know about the relationship between R0 and R1? (*An equation will not earn full credit.*)

Problem 5, continued:

Address	Instruction	RTL
x3000	0001 0010 0111 1111	
x3001	0000 1000 0000 0010	
x3002	0001 0000 0000 0000	
x3003	0000 1111 1111 1100	
x3004	0011 0000 0000 0001	

Part C (10 points): Using the space provided in the diagram, translate the code above into RTL (register transfer language). For any PC-relative addresses, perform the calculation and write the resulting address rather than writing "PC + ..." *Note that the point of the question is not RTL syntax, but the effect of each instruction must be clear from what you write.*

Part D (3 points): Assume that R0 contains the value 1 and that R1 contains a value between 0 and 10 before execution of the code in the diagram above. Explain the effect of the code. A description that requires more than a few words is a good hint that you have the wrong idea.

Name: ______________________________

Use this page as scratch paper.