CMPUT 329 - Computer Organization and Architecture II
Final Exam — Fall 2001

Prof. José Nelson Amaral
Computing Science Department
University of Alberta

Name:

CMPUT 329 Honor Code

By turning in the exam solution for grading, I certify that I have worked all the solutions on my own, that I have not copied or transcribed solutions from a classmate, someone outside the class, or from any other source. I also certify that I have not facilitated or allowed any of my classmates to copy my own solutions. I am aware that the violation of this honor code constitutes a breach of the trust granted me by the teaching staff, compromises my reputation, and subjects me to the penalties prescribed in Section 26.1 of the University of Alberta 2001/2002 Calendar.

Edmonton, December 13, 2001.

____

Question 1	/30
Question 2	/20
Question 3	/15
Question 4	/15
Question 5	/20
Total	/100
Curving	/100
Rank	/38

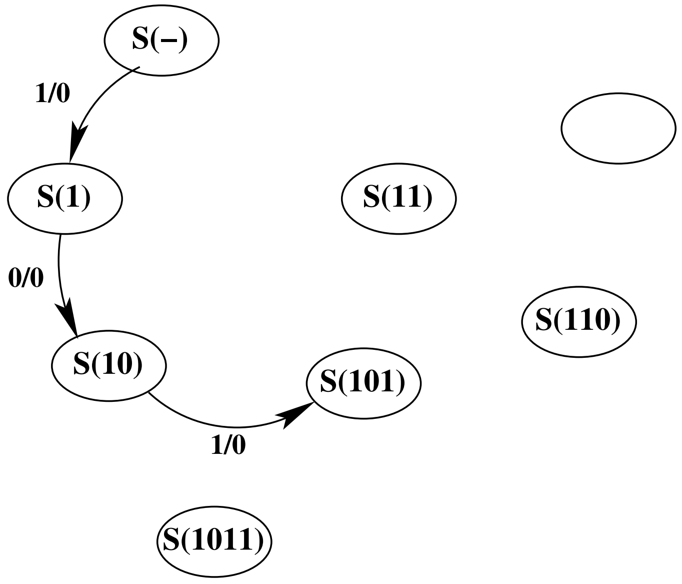

Figure 1: Mealy finite state machine for a sequence detector.

Current State	Next State I=0	Next State I=1	Output I=0	Output I=1
S(-)		S(1)		0
S(1)	S(10)		0	
S(10)		S(101)	0	
S(11)				
S(101)				
S(110)				
S(1011)				

Table 1: State Transition Table for the FSM of Figure 1.

Question 1 (30 points):

The state diagram in Figure 1 is a partial design for a Mealy finite state machine that implements a sequence detector. The sequence detector emits a 1 in its single output Z when either the sequence 1011, or the sequence 110 has been observed in its single input I.

You are asked to:

a. (10 points) Complete the design by adding the missing state transitions directly in the figure (you might want to try it out in your scratch paper first). When you are done, the finite state machine must be completely specified. Notice that you may not need to use all the states shown in Figure 1. You should use the minimum number of states that you need to implement the behavior specified. States without transition to or from them are not considered part of your design.

State	Assignment $Q_2Q_1Q_0$
S(-)	110
S(1)	001
S(10)	000
S(11)	111
S(101)	100
S(110)	010
S(1011)	011
	101

(a)

State	Assignment $Q_2Q_1Q_0$
S(-)	000
S(1)	010
S(10)	111
S(11)	011
S(101)	110
S(110)	101
S(1011)	001
	100

(b)

State	Assignment $Q_2Q_1Q_0$
S(-)	000
S(1)	111
S(10)	110
S(11)	001
S(101)	010
S(110)	100
S(1011)	101
	011

(c)

State	Assignment $Q_2Q_1Q_0$
S(-)	101
S(1)	010
S(10)	000
S(11)	111
S(101)	100
S(110)	011
S(1011)	001
	110

(d)

State	Assignment $Q_2Q_1Q_0$
S(-)	101
S(1)	010
S(10)	110
S(11)	001
S(101)	100
S(110)	111
S(1011)	011
	000

(e)

State	Assignment $Q_2Q_1Q_0$
S(-)	000
S(1)	111
S(10)	101
S(11)	010
S(101)	001
S(110)	110
S(1011)	100
	011

f)

Table 2: Six possible state assignments for the finite state machine of Figure 1.

b. (5 points) Fill the state transition table of Figure 1 according to your complete design. Again leave blank the rows corresponding to the states that are not part of your design.

c. (10 points) Table 2 presents six possible state assignments for the state machine of Figure 1. Consider that either complementing columns or exchanging the order of the columns of a state assignment does not change the cost of its implementation(McCluskey definition). Using this definition, examine the six possible state assignments presented in Table 2, and determine which ones are equivalent. Then indicate the state equivalences in the State Assignment Equivalence Table presented in Table 3. Write a 1 in the cell in the table if the assignment

	a	b	c	d	e	f
a	1					
b		1				
c			1			
d				1		
e					1	
f						1

Table 3: State Assignment Equivalence Table for the Assignments of Table 2.

in the row is equivalent to the assignment in the column. Leave blank cells that correspond to assignments that are not equivalent. Observe that the diagonal is already filled with 1s because, by definition, every state assignment is equivalent to itself. **(Warning: Be aware that although this table looks similar to the ones used to represent DAGs in class, there is no relation between what is asked in this question and the DAG representation).**

d. (5 points) The following rules were presented in class to guide the choice of state assignments:

Successor Rule: If the state machine has transitions $S_i \rightarrow S_j$ and $S_i \rightarrow S_k$ then states S_j and S_k should be close to each other.

Predecessor Rule: If the state machine has transitions $S_a \xrightarrow{I_n} S_c$ and $S_a \xrightarrow{I_n} S_d$, *for the same input condition* I_n, then states S_a and S_b should be close to each other.

Output Rule: States that produce the same output (with the same input condition for Mealy machines) should be close to each other.

According to these rules, which one of the six possible state assignments presented in Table 2 is the best assignment for the state machine of Figure 1?

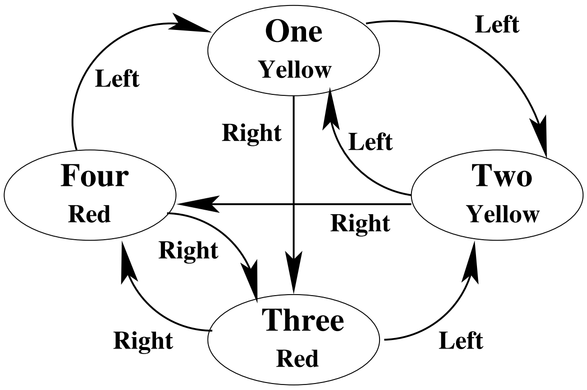

Figure 2: State diagram for your nephew's toy.

Question 2 (20 points):

At Christmas your nephew received a little toy for a gift. This toy has two push buttons, a left and a right one, and two lights, one yellow and one red. As the buttons are pressed the light that is on might stay on, or it might go off and the other light goes on. Both lights are never on at the same time. After observing your nephew, and your uncle, playing with the new toy for a while you conclude that the toy implements the finite state machine presented in Figure 2. Thus to proudly show your newly acquired knowledge of digital circuits, you draw the state diagram and explain it to your uncle. You tell him that with four states, the toy must have two flip-flops in order to implement its behavior. Your aunt that was listening to your explanation and looking over your shoulder observes: *"Hmmm, I wonder if they could not get a way with a single flip-flop for that...."*. Is she right? Can you get the same behavior with a single flip-flop? Explain your answer.

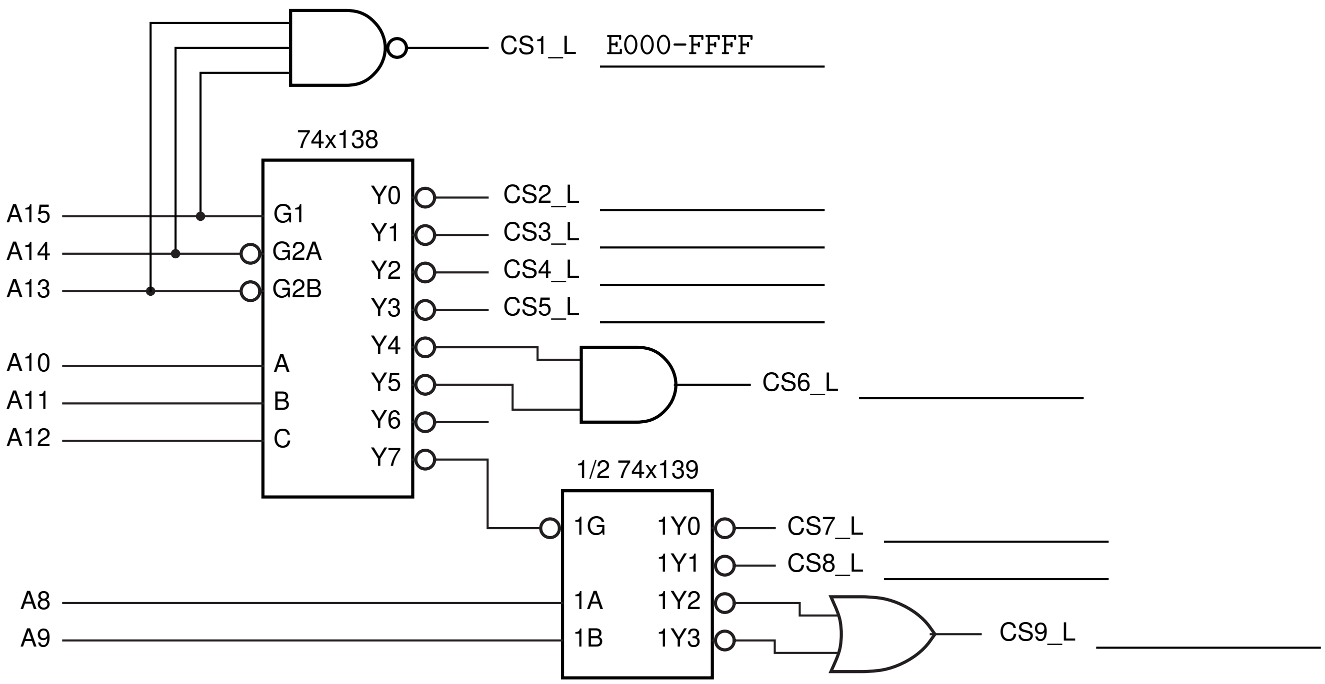

Figure 3: Address Decodification Circuit for a Memory System.

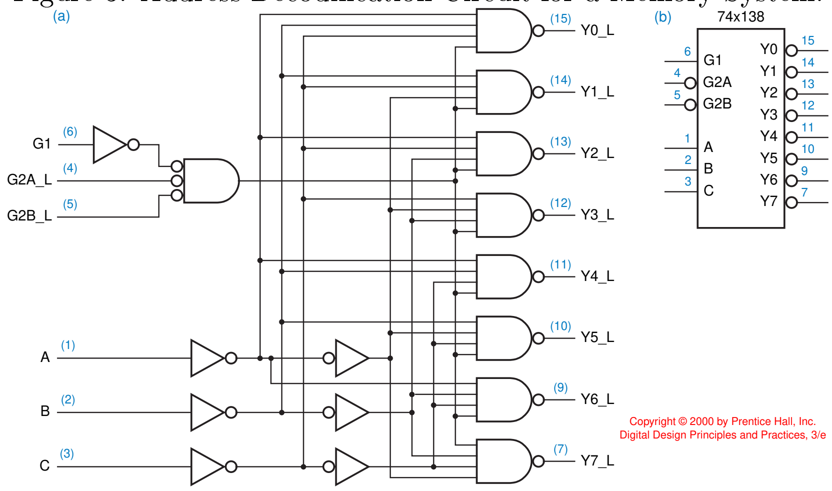

Figure 4: Internal Circuit of the 74x138 3-to-8 decoder.

Question 3 (15 points):

The circuit presented in Figure 3 is used in a microprocessor system as an address decoder to chip-select various memories and peripherals. The microprocessor has 16 address bits, $A[15:0]$ that enable it to specify a 64 Kbyte address space, hex addresses `0000-FFFF`. The first chip-select, `CS1_L`, decodes the address range `E000-FFFF`. In the blanks of Figure 3, write the address ranges decoded by the remaining chip-selects, `CS2_L` through `CS9_L`. To assist you, we present the internal circuit diagram of the 3-to-8 74x139 decoder in Figure 4.

Step name	Action
Instruction Fetch	`IR ← Memory[PC]` `PC ← PC + 4`

Instruction Decode	`A ← Registers[IR[25-21]]`
	`B ← Registers[IR[20-16]]`
	`Target ← PC + (sign-extend(IR[15-0]) << 2`

Branch Completion	`if (A == B) then PC ← Target`

Table 4: Steps in the execution of a branch instruction.

bne $1, $2, 100

Opcode	rs	rt	address
5	1	2	100

31 26	25 21	20 16	15 0
000101	00001	00010	0000000001100100

Table 5: Example of a branch when not equal (bne) instruction.

Question 4 (15 points):

Figure 5 presents a multi-cycle datapath for a RISC microprocessor. Table 4 summarizes the steps required for the execution of a branch instruction, and Table 5 shows an example of a branch executed in this datapath. You are asked to complete Table 6 with the value that each of the control signal listed in the columns of the table must have at each step of the branch execution in order for the branch to be executed properly. You must use one of three values for each signal: `0` when the signal is not active, `1` when the signal is active, and `X` if the branch instruction will execute correctly regardless of the value of the signal. Notice that it is incorrect to specify either `0` or `1` when the don't care value `X` should be specified because you would make the implementation of the control logic more complex than it has to be. For multiplexers with 3 or 4 inputs, you will have to specify two bits for the control signal.

Step	IRWrite	PCSource	PCWrite	PCWriteCond	ALUSrcA	ALUSrcB	RegWrite	RegDst
Instruction Fetch								
Instruction Decode								
Branch Completion								

Table 6: Control signals for the branch operation.

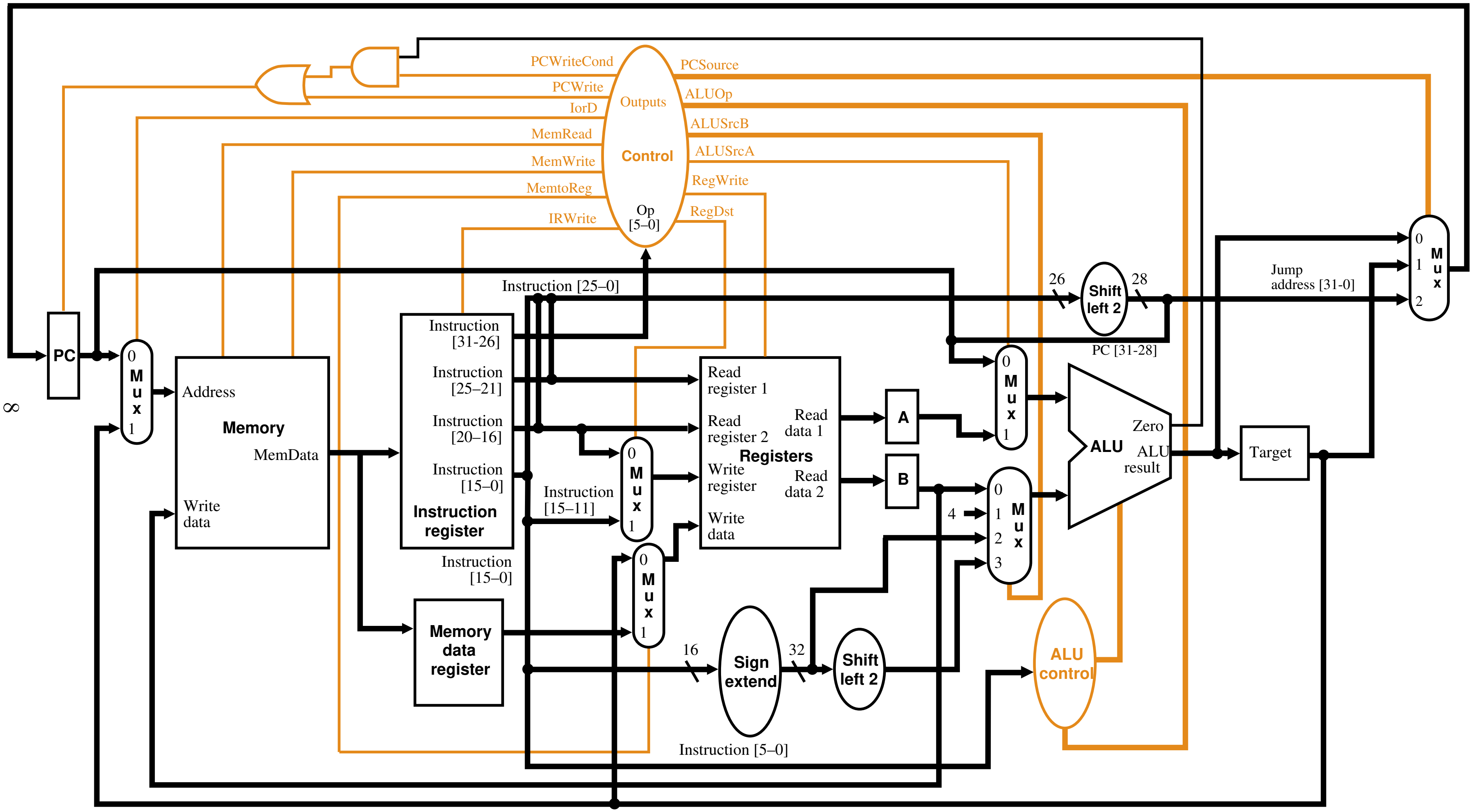

Figure 5: A Multi-cycle Datapath for a RISC Microprocessor

Question 5 (20 points):

This is a **true** or **false** question. For each statement you have to indicate whether the statement is true or false. Use the capital letter **T** to indicate true, and the capital letter **F** to indicate false. You win 4 points for each correct answer. **You lose 2 points for each incorrect answer.** You are neither rewarded nor penalized for questions that you do not answer. If you answer all items correctly you will make a maximum of 20 points in this question. Regardless of how many items you answer wrong, your score in this question cannot be below zero.

a. (   ) Clock skew is caused by differences in path length that the clock signal traverses before reaching two clocked devices in a synchronous design. Clock skew is undesirable because it adds to the amount of holding time available to the clocked devices that receive the clock signal later.

b. (   ) A circuit with a timing hazard is guaranteed to produce a glitch in the output whenever the signal traverses the path that has the hazard.

c. (   ) SRAM memories use two signals to determine when they should write data in the bus, a Chip Select (CS) signal, and an Output Enable (OE) signal. These two signals are necessary to simplify the implementation of the internal structure of each bit storage cell in the SRAM.

d. (   ) An Enhanced Data Output RAM (EDO-RAM) is a DRAM device that keeps the data valid in the data bus after the CAS signal has been made inactive, thus the EDO-RAM may allow the processor to complete a memory read operation in less clock cycles than a standard DRAM.

e. (   ) The most important advantage of the Rambus DRAM technology is that it simplifies the logic in the memory controller and thus enable faster clocks in the memory bus than the clock speeds implemented with SDRAM technology.