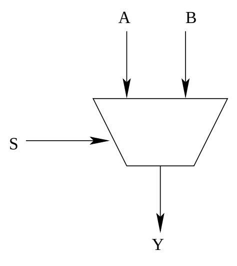
Name: ____

Intro. to Computer Architecture	**Homework 3**
CSE 240 Autumn 2004	**DUE: Fri. 1 October 2004**

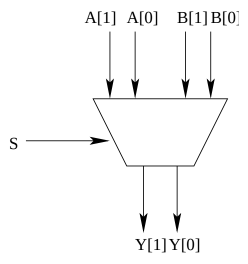
Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible. Please show your work. Due at the ***beginning of class***. Total points: 34.

1. [6 Points] **Muxes and Memory.**

   (a) A 2-bit 2-to-1 mux (below, right) is similar to a 1-bit 2-to-1 mux (below, left) except the former selects among 2-bit inputs rather than 1-bit inputs. More specifically, the output of 2 bit 2-to-1 mux is defined as follows: if $S = 0$, $Y_i = A_i$ and if $S = 1$, $Y_i = B_i$, for $i = 0, 1$. Construct a circuit with the behavior of a 2-bit 2-to-1 mux using two 1-bit 2-to-1 muxes.

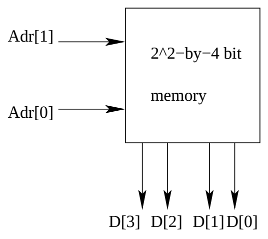


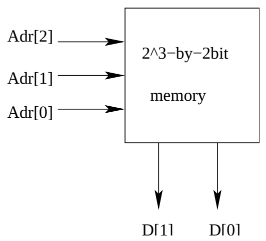
1 bit 2−to−1 Mux



2 bit 2−to−1 Mux

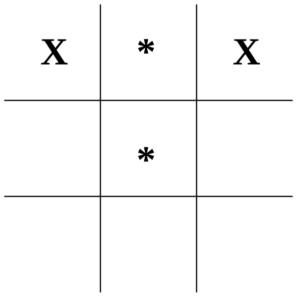
(b) Memories come in many shapes and sizes. If we need a memory of a particular size and addressability, we can often construct it out of other, differently-structured memories. Construct a 2^3-by-2-bit memory (below, right) using a 2 bit 2-to-1 mux and a 2^2-by-4-bit memory (below, left). Assume that the memories are read-only memories (ROMs), so you need only be concerned with reading.





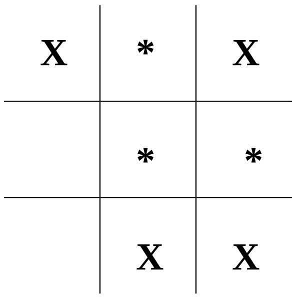
Name: ____

2. [8 Points] **Combinational Logic Circuits.** Do you remember the game of tic-tac-toe? (See p. 73–74 in your textbook if you do not.) We can use bit vectors to represent the state of any tic-tac-toe game. Consider the following game state (we use "*" instead of "O" to avoid confusion with zero).



We can represent this state using two 9-bit vectors **AX** = (101 000 000) and **A*** = (010 010 000). The vector **AX** indicates whether each cell contains an **X** (denoted with a 1) or a non-**X** (denoted with a 0). The first three bits represent the top three cells from left to right; the next three bits represent the middle three cells; and the last three bits in the vector represent the bottom three cells. Vector **A*** is analogously defined.

(a) Give the bit-vector representation (both **AX** and **A***) for the following game state.



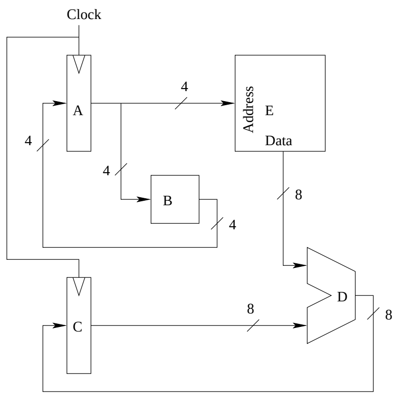
AX :

A* :

(b) Give the bit-vector representations (both **AX** and **A***) of all possible next states for the above game, assuming it is $*$'s turn next.

(c) Construct a gate-level logic circuit to determine if the $*$ player has won (*i.e.*, has three in a row). The input to this circuit is the 9-bit **A*** vector (**A*** [0-8]). The output (W$*$) should be 1 if only if the $*$ player has won. You may assume the game state was arrived at through a succession of legal moves (*i.e.*, both players will not have 3 in a row).

3. [8 Points] **Sequential Logic Circuits.** Consider the following circuit.



Components are labeled according to the following key: (A) A 4-bit register representing an unsigned binary number (initially set to 0). (B) A 4-bit incrementer. (C) An 8-bit register (initially set to 0). (D) An 8-bit adder. (E) A 2^4-by-8-bit memory. Assume the memory has the following contents.

Address	Contents	Address	Contents
0	2	8	5
1	1	9	2
2	1	10	1
3	4	11	1
4	2	12	3
5	3	13	4
6	2	14	1
7	2	15	1

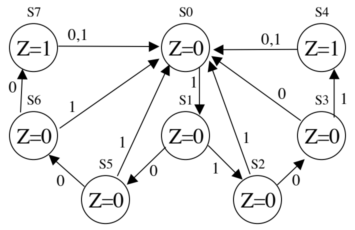
(a) Determine the values stored in the A and C registers at the end of each clock cycle for 8 cycles. Complete the following table with this information.

Clock Cycle	A	C
0	0	0
1		
2		
3		
4		
5		
6		
7		
8		

(b) In a sentence, what does this circuit do?

4. [12 Points] **Finite State Machines.** It's your first day at the Pretty Good Lock Co. Your boss knows you're a whiz so she asks you to build a combination lock circuit. This lock takes a combination (*i.e.*, a sequence of 0s and 1s on the 1-bit input X) and generates a 1-bit output Z, indicating whether the lock is closed (0) or open (1). The lock must accept two combinations: 1, 1, 0, 1 and 1, 0, 0, 0.

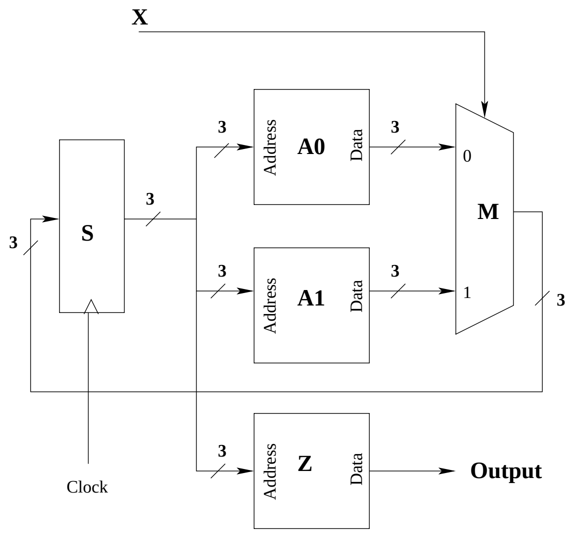
   (a) Where do you begin? First, you examine the state diagram (below) created by your predecessor (he got fired; don't ask why). Assume S0 is the initial state. Edges are labeled with the input that causes a state transition. Nodes are labeled with the output (*e.g.*, "Z=1") of that state.



   Experiment with the finite state machine defined by this state diagram to determine if it actually accepts 1, 1, 0, 1 and not, for example, 1, 1, 0, 1, 0 by completing the following table. (S-next represents the state reached when a transition is made from state S on input X. Z is the output in state S-next.)

S	X	S-next	Z
0	1		
	1		
	0		
	1		
	0		

(b) So far so good. You start to build a circuit for the state diagram, but you suddenly realize that once you build the circuit, you will be unable to change the combination without building a new circuit. Maybe that's why your predecessor got fired! You resolve to do better and build the following programmable combination lock circuit.



Components are labeled according to the following key: (A0) A 2^3-by-3-bit memory. (A1) A 2^3-by-3-bit memory. (Z) A 2^3-by-1-bit memory. (S) A 3-bit register (initially containing 0). (M) A 3-bit 2-to-1 mux. Suppose the contents of memories A0, A1 and Z are initialized as follows.

Address	A0	A1	Z
0	1	0	0
1	0	2	0
2	3	0	0
3	0	4	0
4	5	0	0
5	0	6	0
6	0	0	1
7	0	0	0

Voila! You've built a programmable combination lock, and it's currently programmed to accept the combination 0, 1, 0, 1, 0, 1. Try it out on 0, 1, 0, 1, 0, 1, 1 by completing the following table.

Clock Cycle	S	X	S-next	Z
0	0	0	1	0
1	1	1		
2		0		
3		1		
4		0		
5		1		
6		1		

(c) Okay, cool. Program this circuit (*i.e.*, provide the contents of the memories) to accept the two combinations from part (a). Complete the following table.

Address	A0	A1	Z
0			
1			
2			
3			
4			
5			
6			
7			

(d) In fact, the above circuit can be used to implement any finite state machine with 1 input, 1 output, and up to 8 states. (i) How would you modify your circuit to support more than 8 states? (ii) How would you modify your circuit to support 2-bit inputs? Don't actually build the circuit; instead, describe what changes you would make to the existing circuit.

5. [No Points] **Last and Most Important Question!** Please complete this question, and give us your feedback!
   (a) How many hours did you spend on this assignment?

   (b) On a scale of 1-5, how difficult did you find this assignment? (1-easiest, 5- most difficult)

   (c) Do you have any other comments on your experience completing this assignment? What are they?