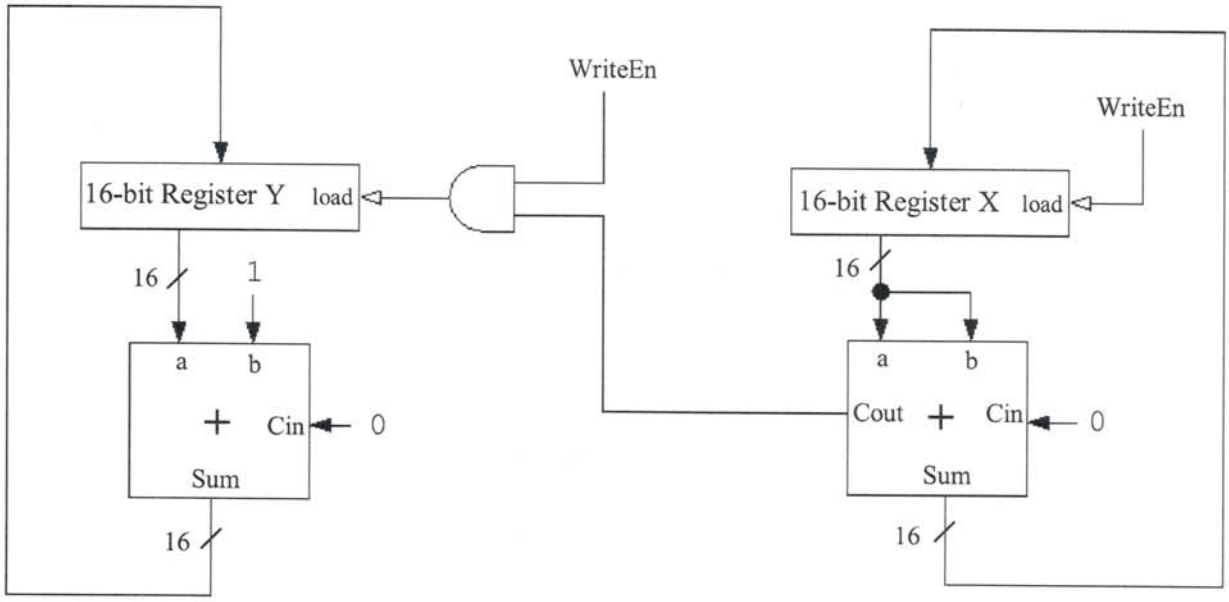
Name: _____

Problem 2 (20 Points): Sequential Circuits

Part A (10 points): For the diagram below, assume that Y initially contains a 0 while X contains some 2's complement number. In a single sentence, describe what the contents of Y represent after the WriteEn signal is raised to a logical '1' and lowered to '0' 16 times.

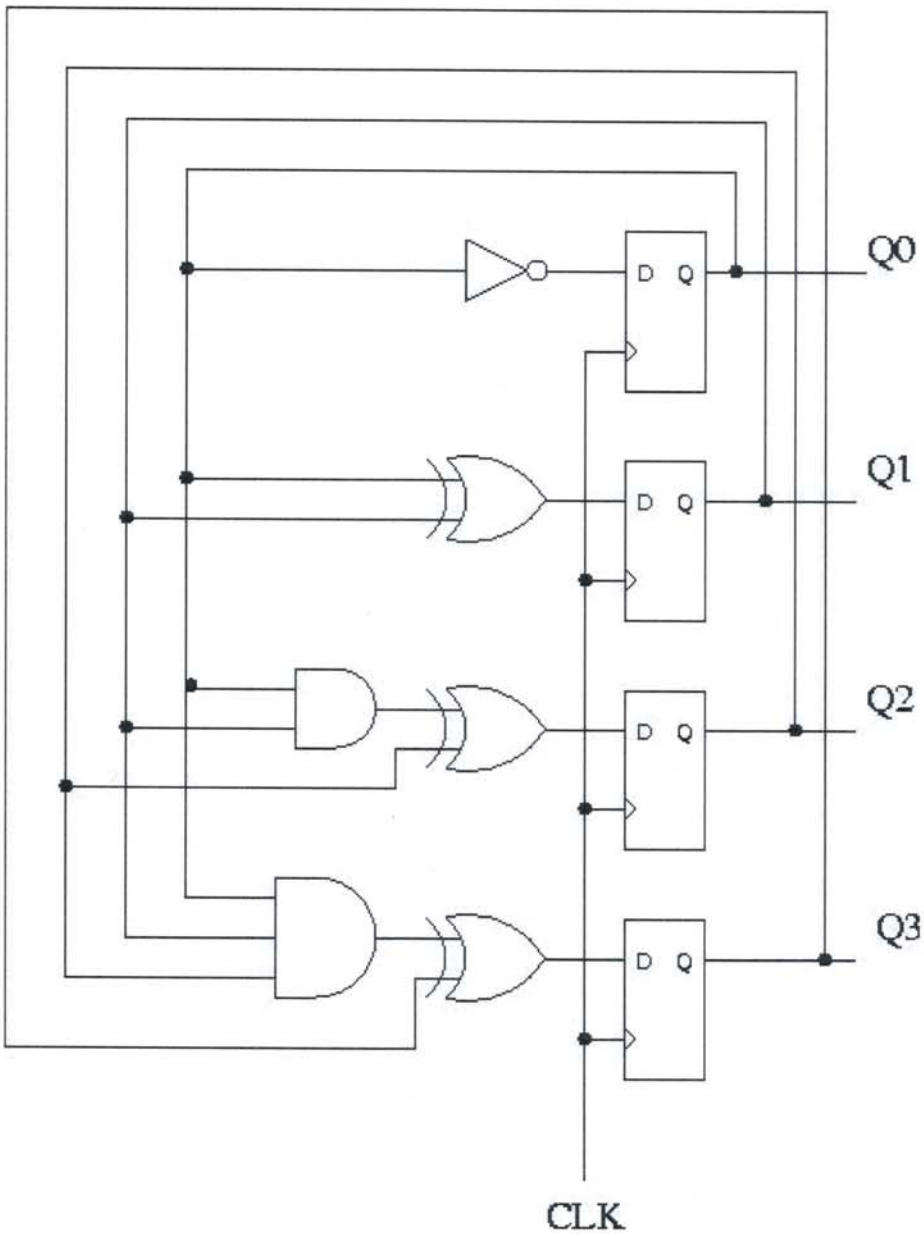


Y holds the number of 1 bits in the value initially stored in X.

Name: _____

Problem 2, continued:

Part B (10 points): Describe what the following sequential circuit does in a sentence. Assume that the output $Q_3Q_2Q_1Q_0$ is initially 0000.



4-bit binary counter (counts 0 to 15 and starts over)

Name: _____

Problem 3 (20 Points): Memories

Part A (4 points): A memory stores a total of 16 Mbits (1 Mbit is 2^{20} bits) and uses 19-bit addresses. What is the addressability of this memory in bits?

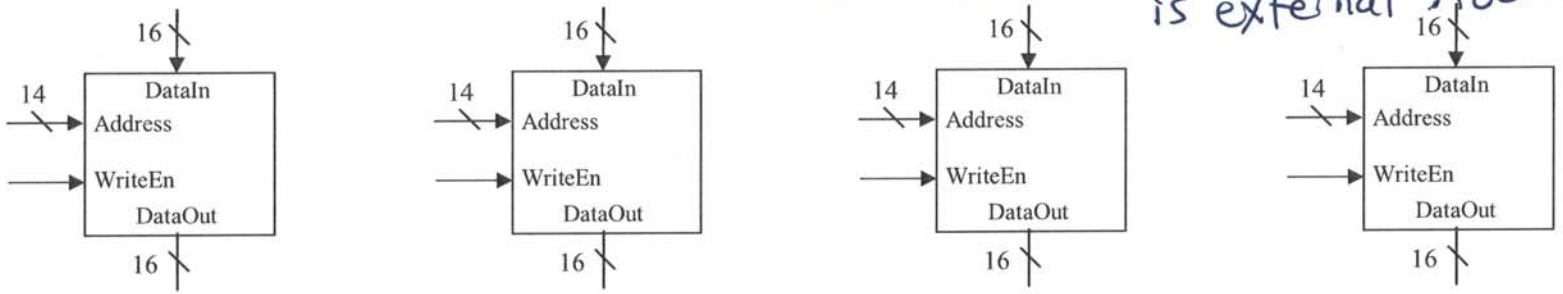
$$\frac{2^{24}}{2^{19}} = 2^5 = 32 \text{ bits}$$

Part B (4 points): Is it ever possible to have the PC, IR, MAR, and MDR all contain the same value during the FETCH cycle of the von Neumann instruction cycle? If so, explain how. If not, explain why not.

N/A (yet)

***Part C** (12 points): You must design a 2^{16} by 16-bit memory for the LC-3. However, you have only been given four 2^{14} by 16-bit memory blocks to work with. Draw your design on the next page, using any gates or combinational logic blocks we discussed in class in addition to the 4 memory blocks.

- External DataIn to all chips' DataIn
- External Address[13:0] to all chips' Address
- External WriteEn to enable of 2-to-4 decoder; External Address[15:14] to select of same decoder; one line from decoder to each chip's WriteEn (order doesn't matter)
- All chips' Data out to 16-wide 4-to-1 muxes. Input order must match address assignment used for decoder. Mux select is external Address[15:14].

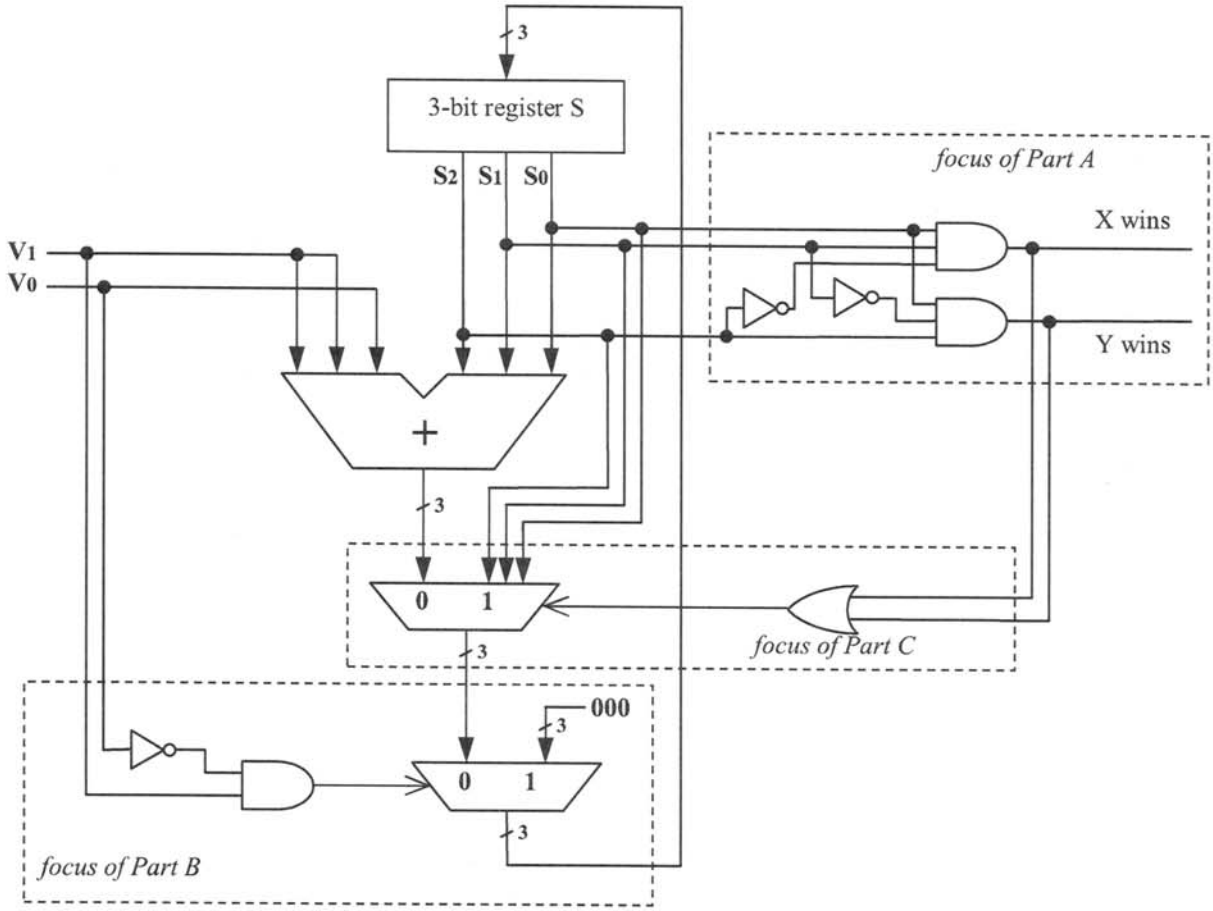


Name:

Name: _____

Problem 4 (20 Points): Finite State Machines

A certain finite state machine implements a match scoring system for a game played in rounds by two players (X and Y). Each round can result in a win for X, a win for Y, or a tie. When a player gets ahead by three rounds (has won three more times than the other player), that player wins the match. Answer all parts of this question based on the implementation shown below. The parts are designed to help you through the analysis.



Name: _____

Problem 4, continued:

Part A (2 points): For what value(s) of the register S does player X win? Player Y?

X wins on 011 (+3).

Y wins on 101 (-3).

Part B (3 points): Explain the function of the bottom mux and the logic controlling it.

Forces state to 000 when $V_1 V_0 = 10$.

Part C (3 points): Explain the function of the OR gate and the mux that it controls.

Locks state once either player has won.

Part D (4 points): Explain the meaning of the bits held in register S.

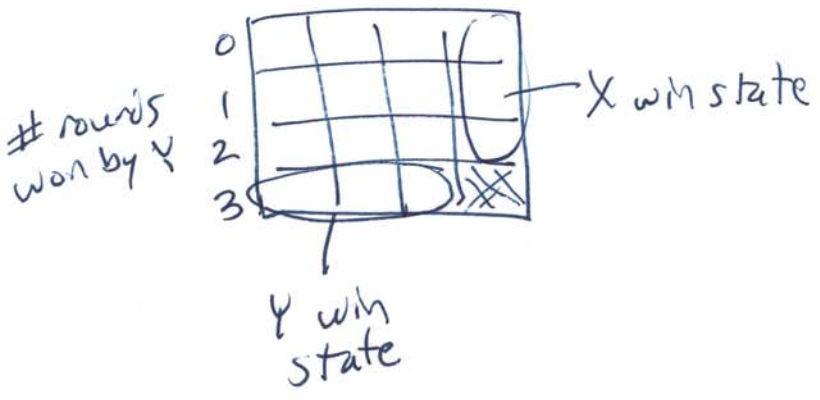
games X is ahead of Y (as 2's complement)

Part E (4 points): Fill in the table below of input values and their meanings; three of the meanings have been given already.

V_1	V_0	meaning
0	1	player X won a round
1	1	player Y won a round
0	0	round tied
1	0	(your answer to Part B)

***Part F** (4 points) How many bits of state are necessary if the game is instead played until one player has won three times? Justify your answer.

rounds won by X

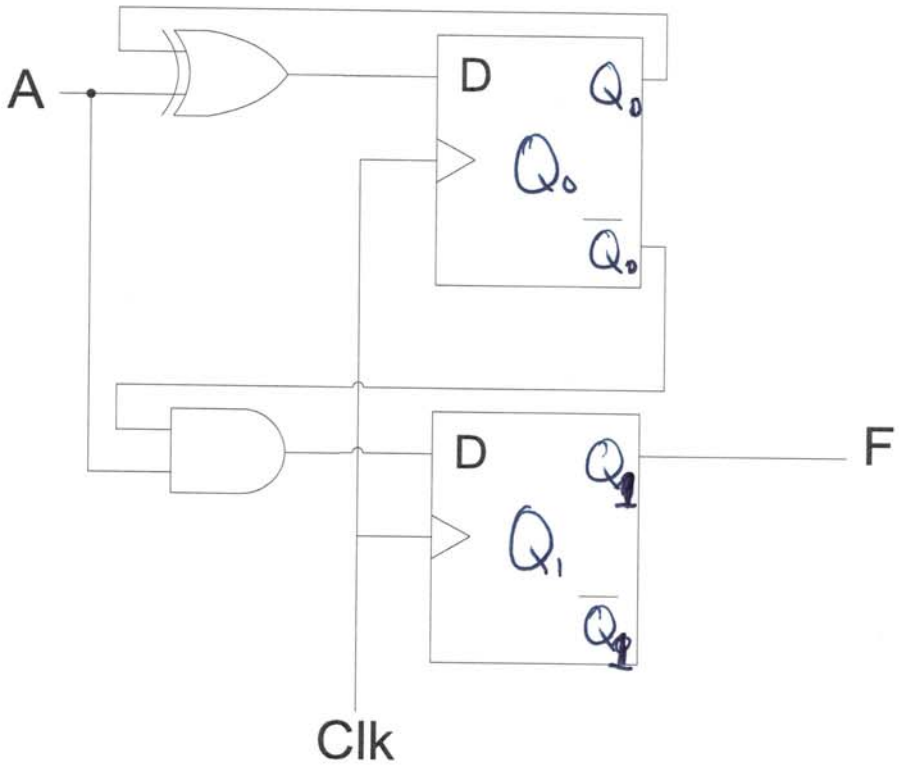


11 states
⇒ 4 bits

 Name: _____

Problem 3 (20 points): Finite State Machines

For the following question, refer to the circuit below.



$$Q_1^+ = A\overline{Q}_0$$

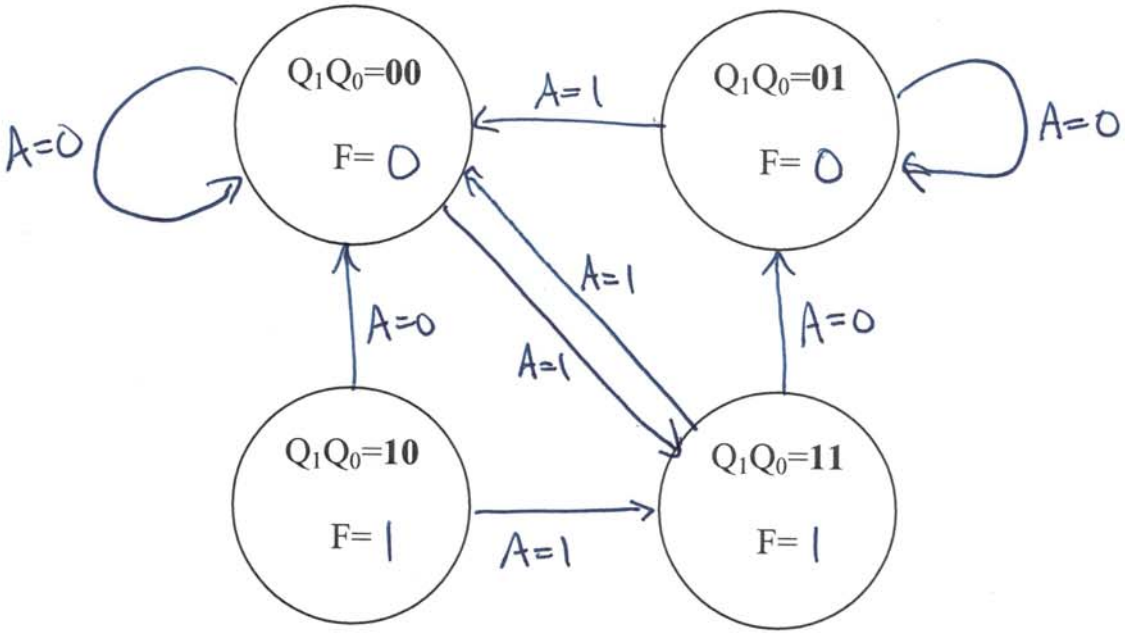
$$Q_0^+ = A \oplus Q_0$$

$$F = Q_1$$

$A=0$	$A=1$
$Q_1^+ = 0$	$Q_1^+ = \overline{Q_0}$
$Q_0^+ = Q_0$	$Q_0^+ = \overline{Q}_0$

 Name: _____

Part A (7 points): Draw the finite state machine transition diagram for the circuit. Be sure to fill in all input transition arcs and state outputs. The state diagram ($Q = Q_1Q_0$) has been partially filled in to help you get started. *Hint: For each value of A, write the next state in terms of the current state, then label the diagram below.*



Part B (3 points): Assuming that the circuit is started in state $Q_1Q_0 = 00$, are all states reachable? In other words, is it possible for the state of the circuit to take on any given combination of 0s and 1s through some sequence of inputs? Explain your answer.

No. The $Q_1Q_0 = 10$ state has no arcs entering it.

Part C (5 points): Fill in the table for the new state Q_{new} ($=Q_1Q_0$) and the output F after each input transition has completed. Assume that the circuit is initially in state $Q_1Q_0=00$.

A	0	1	1	0	0	1	0	1	1	0	1	1
$\mathbf{Q_{new}}$	00	11	00	00	00	11	01	00	11	01	00	11
$\mathbf{F_{new}}$	0	1	0	0	0	1	0	0	1	0	0	1

Part D (5 points): Summarize what this circuit does in one sentence.

Outputs a 1 after every other input 1.

Problem 1 (20 points): Short Answer

Part A (5 points): The IR (Instruction Register) holds the instruction that is to be executed. Given that the instruction bits can be held in the MDR, why is an IR necessary?

Some instructions (loads and stores) need to make use of MDR while being processed, so changing IR before they complete is difficult.

Part B (5 points): Consider the following LC-3 instruction (x3500 is the address at which the instruction is located):

x3500 LD R5, _____ ; we want to put the value x2BFF in register R5

Given the above instruction, what is the range of memory addresses at which the value x2BFF can be stored such that the above instruction can be executed successfully?

N/A (yet)

Part C (5 points): A certain memory chip has a total of 2^{32} bits and is 8-bit addressable. How many address bits must be specified when reading or writing a location on this chip?

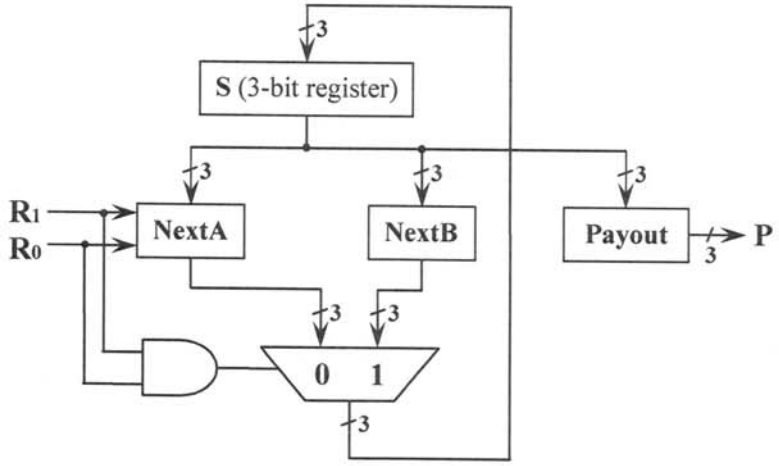
$\frac{2^{32}}{2^3} = 2^{29}$ addresses $\Rightarrow$ 29 bits

Part D (5 points): What fraction of the range of numbers that can be represented with an N-bit 2's complement data type can also be represented with an (N+1)-bit unsigned data type? (Justify your answer.)

N/A (done previously)

Problem 4 (20 points): Finite State Machines

As part of your job with the IRS, you are responsible for validating the accuracy of a certain casino's claims about the amount paid out by their new slot machine model. Each time a gambler inserts a coin, the machine randomly chooses one of four fruits—orange, peach, cherry, or lemon. A 3-bit FSM makes a transition based on the fruit chosen (represented with 2 bits), and the new FSM state specifies a payout in coins (a 3-bit unsigned value). A high-level diagram of the FSM appears below.



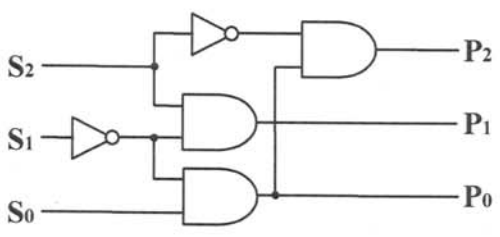
The inputs R_1 and R_0 encode the fruit selected as follows:
orange: R_1R_0=00 peach: R_1R_0=01 cherry: R_1R_0=10 lemon: R_1R_0=11

Part A (2 points): Which fruits use the NextA block to determine the FSM's next state?

orange, peach, and cherry

Part B (4 points): Based on the payout component shown below, fill in the table with the number of coins $P_2P_1P_0$ paid for each FSM state.

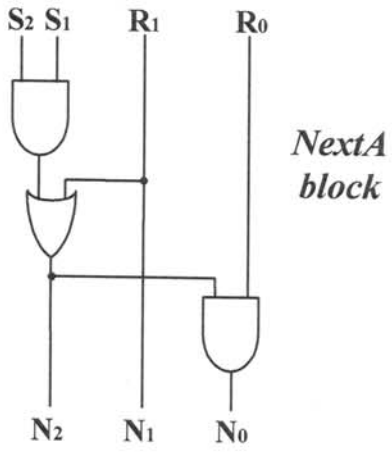
S_2	S_1	S_0	P_2	P_1	P_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	0	0	0

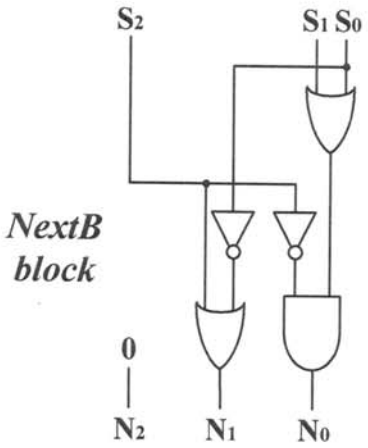


$\bar{S}_2\bar{S}_1S_0$

$S_2\bar{S}_1$

$\bar{S}_1S_0$





The NextA and NextB blocks are shown above. For both blocks, the S input is the current FSM state, the R input is the current fruit, and the N output is the next FSM state.

Part D (1 point): Find the next FSM state $N_2N_1N_0$ when a cherry is seen (R_1R_0=10).

(next A) 110

Part E (2 points): The next FSM state takes one of two values when an orange is seen (R_1R_0=00). Fill in the table below specifying the next states as a function of the current state, using X to represent irrelevant inputs.

(nextA)

S_2	S_1	S_0	N_2	N_1	N_0
1	1	X	1	0	0
else			0	0	0

Part F (2 points): The next FSM state takes one of two values when a peach is seen (R_1R_0=01). Fill in the table below specifying the next states as a function of the current state, using X to represent irrelevant inputs.

(next A)

S_2	S_1	S_0	N_2	N_1	N_0
1	1	X	1	0	1
else			0	0	0

Part G (4 points): Find the next FSM state when a lemon is seen (R_1R_0=11).

S_2	S_1	S_0	N_2	N_1	N_0
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	0

(next B)

$N_1 = S_2 + \overline{S_0}$

$N_0 = \overline{S_2}(S_1 + S_0)$

Part H (5 points): Identify all winning combinations for this state machine and the amount paid for each. A winning combination is a minimal sequence of input combinations that takes the FSM from an arbitrary state into a particular payout state. It is minimal in the sense that no suffix of the combination suffices to reach the same payout state. For example, if the combination "cherry lemon" were such a sequence, neither "lemon cherry lemon" nor "orange cherry lemon" could be, since "cherry lemon" is a suffix of both.

Hint: work backwards from each state with non-zero payout.

001 (payout 5)

lemon, lemon, lemon (Lunetta loves these!)

---

101 (payout 3)

cherry, peach

---

100 (payout 2)

cherry, orange

Problem 1 (10 points): Short Answer

Part A (2 points): Your friend is writing an LC-3 program in which the instruction at address 0x5000 must transfer control to another address. He's asked you to figure out which instructions he can use to accomplish this task. Below is a list of two target addresses. Your job is to figure out which instruction gets us there. We've provided partial solutions in binary for you. Provide the missing bits.

(i) Target address x5070 | 0000 1110 ____ ____ |

(ii) Target address x4FF0 | ____ ____ 1110 ____ |

N/A (yet)

Part B (3 points): Suppose we run an LC-3 program whose first instruction is at memory location 0x0010. The instruction at that location must put the value 0 into R1. List all of the individual instructions that can be used to accomplish this task. Be careful about making assumptions about the contents of memory or the register file!

N/A (yet)

Part C (2 points): In the von Neumann Model, what is the purpose of the program counter. Keep your answer brief!

PC points to the next instruction (holds the address of the next instruction).

Part D (3 points): Suppose that you are designing multiplier unit for a chip to perform addition and multiplication on 2's complement numbers. You want to design this unit such that overflow errors do not occur. If the inputs to the multiplier are two n-bit 2's complement numbers, how many bits (minimum) are needed on the output? Provide your answer in terms of n.

N/A (done previously)

 Name: _____

Problem 3 (20 points): Memory Organization

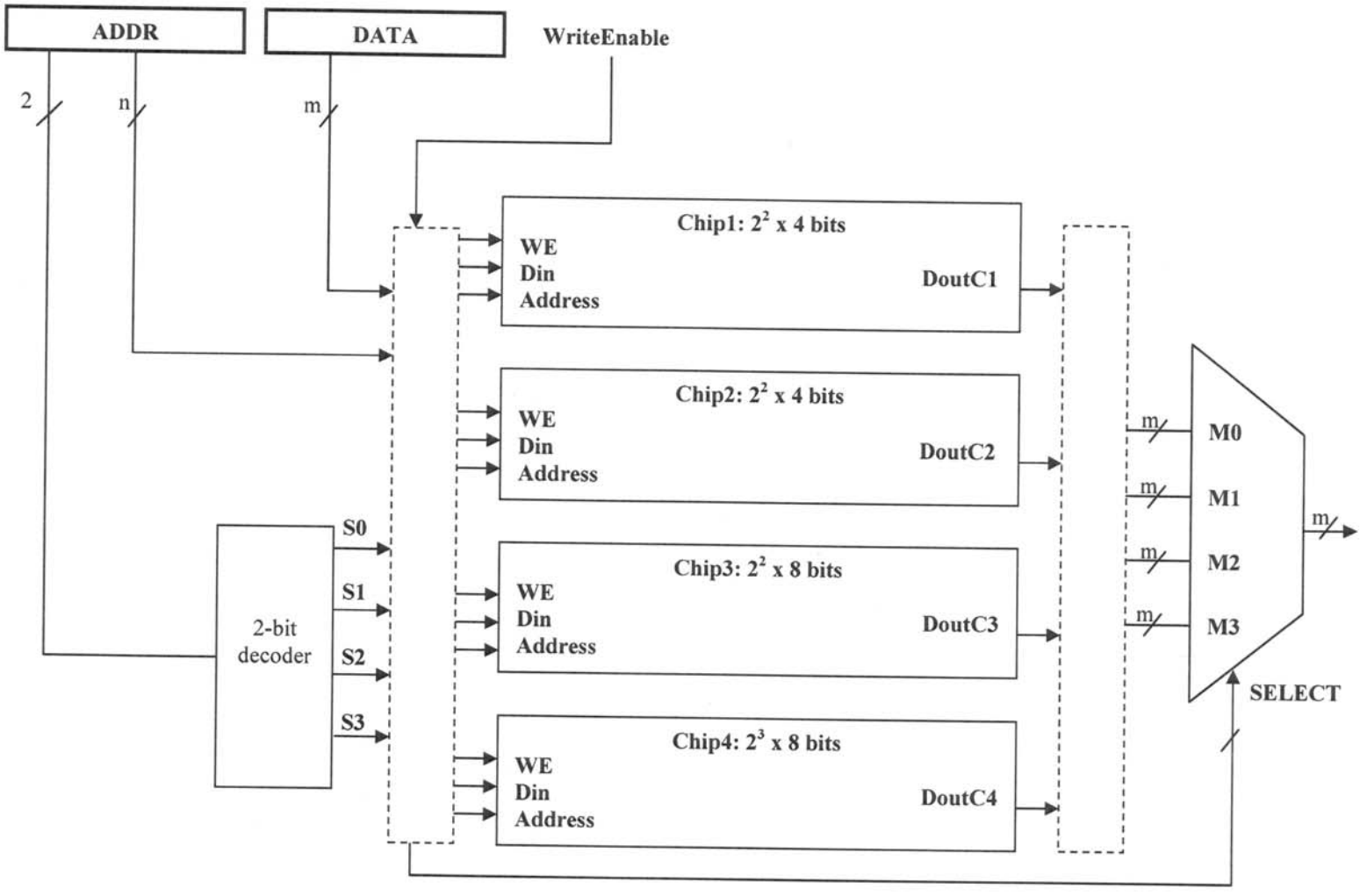
The ECE 190 TAs are trying to build a grading robot so they don't have to grade your exams. The robot will be controlled by a TC-2 processor (Tiny Computer) with an 8-bit word size. Unfortunately, when your TAs went to the Chip Store, memory chips large enough for the robot's computer were too expensive. To save money, the TAs decided to get a collection of smaller, cheaper chips and organize them into the larger memory required for the robot. They bought 4 chips altogether: two chips of size **2^2 x 4 bits**, one chip of size **2^2 x 8 bits**, and one chip of size **2^3 x 8 bits**. Use the following diagram to answer the questions about how the TAs designed the memory system of the TC-2.

Part A (10 points): If the robot requires 16 memory locations (all of the same size)…

What is the addressability of the combined memory? 8 bits

What is the address space? 16 addresses

How many address bits are required? 4 bits



 Name: _____

Part B (10 points): For this question, refer to the diagram on the previous page. You will be specifying the missing logic and wiring in the dotted boxes. You may use any of the signals provided or simple expressions containing these signals. If using ADDR, *you must specify which bit or bits you are using to receive credit* – for example, ADDR[2:0] for bits 2,1, and 0 or ADDR[1] for just bit 1. (Note that bit 0 of ADDR is the rightmost bit of the Address).

Chip Write Enable Signals

WE on Chip1	S0 · WriteEnable
WE on Chip2	S0 · WriteEnable
WE on Chip3	S1 · WriteEnable
WE on Chip4	(S2+S3) · WriteEnable

Mux Inputs and Select

M0	DoutC1 : DoutC2
M1	DoutC3
M2	DoutC4
M3	DoutC4
SELECT	ADDR[3:2]

Chip Address Lines

Address on Chip1	ADDR[1:0]
Address on Chip2	ADDR[1:0]
Address on Chip3	ADDR[1:0]
Address on Chip4	ADDR[2:0]

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): Using an 8-bit 2's complement data type, what is the smallest positive number that generates an overflow when added to itself?

N/A (done previously)

Part B (5 points): How many positive integers are represented in a 13-bit 2's complement data type?

N/A (done previously)

Part C (5 points): Using one or more LC-3 instructions, implement a branch if positive (BRp) to an address outside the range of the branch instruction. Assume that the address to which you want to branch is stored in R5. Write your instruction(s) in binary. *Note: you may not need all the lines provided below.*

Address	Instruction
x3000:	
x3001:	
x3002:	
x3003:	

N/A (yet)

Part D (5 points): The current LC-3 memory contains a total of 1 Megabit (2^{20} bits). If we were to make the memory byte-addressable (a byte is 8 bits) without changing its size, and without changing the size of the instructions (16 bits), how many bits would be necessary in the following registers?

MAR [17] IR [16]

Problem 3 (10 points): Memory

In a typical memory, setting the write enable input (WE) to 1 causes all bits of the selected memory location to accept new values. For a certain ECE445 (senior design) project, we need a 4-address, 3-bit addressable memory that allows us to write to each bit at a location separately. To do so, we hired a computer engineer from the University of Michigan.

The design is done by modifying the 4-address, 3-bit, memory that we discussed in class. Except for the WE input, all inputs and outputs remain the same. More specifically, ADDR is a 2-bit input for address selection, Din is a 3-bit input for providing bits when writing to a memory location.

As for the WE input, the modified memory structure no longer has a single WE input to all bits. Rather, when a location is selected, 3 WEout signals will independently determine if each bit in that location will be written. This modification has already been done by an Illinois student.

The Michigan engineer proposes to construct a combinational logic that provides a nice external interface for using the memory. The logic translates a 4-bit WE external input signal into the internal 3-bit WEout signals as follows:

WE is 4 bits wide and behaves as follows:

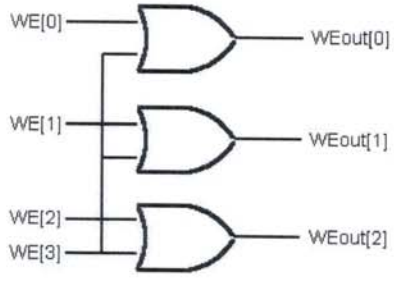
WE [3:0] = 0001	=>	write to bit 0 only
WE [3:0] = 0010	=>	write to bit 1 only
WE [3:0] = 0100	=>	write to bit 2 only
WE [3:0] = 1000	=>	write to all the bits
WE [3:0] = 0000	=>	do not write to any of the bits
WE [3:0] = other	=>	treat as a read cycle

Part A (6 points): For a first round of testing, we executed the six cycles shown on the left below. Assuming correct implementation and assuming that all the bits in the memory are initialized to 1, fill in the table on the right to reflect the final state of the memory (after the six cycles).

Cycle #	ADDR[1:0]	WE[3:0]	Din[2:0]
1	00	0100	000
2	10	1000	010
3	10	0000	110
4	01	0010	001
5	11	0001	011
6	00	1000	100

address	bit 2	bit 1	bit 0
00	~~0~~ 1	0	0
01	1	0	1
10	0	1	0
11	1	1	1

Part B (4 points): The memory designed by the Michigan alumnus passed the first test. However, the next round of tests revealed some undesired behavior. Examining his design, we found the following logic for generating the signals that specify whether or not each individual bit is written (WEout[2:0]).



In 25 words or less, explain the undesired behavior.

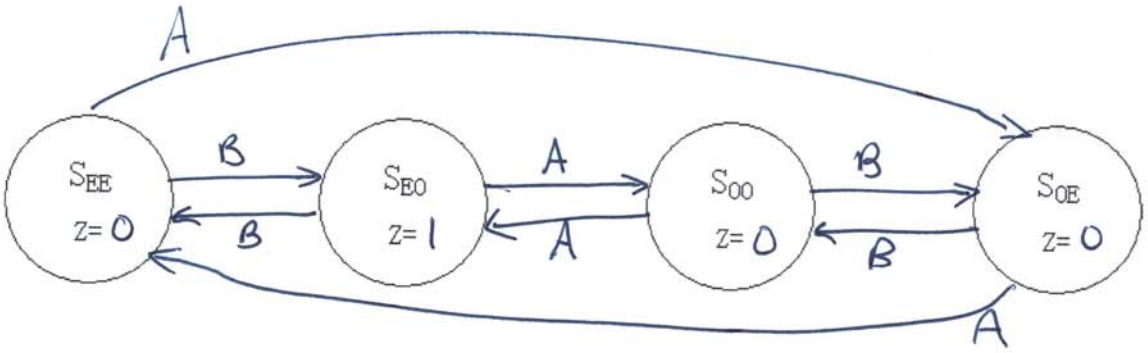
The "other" cases (read cycles) all write to some/all of the bits!

Problem 4 (25 points): Finite State Machines

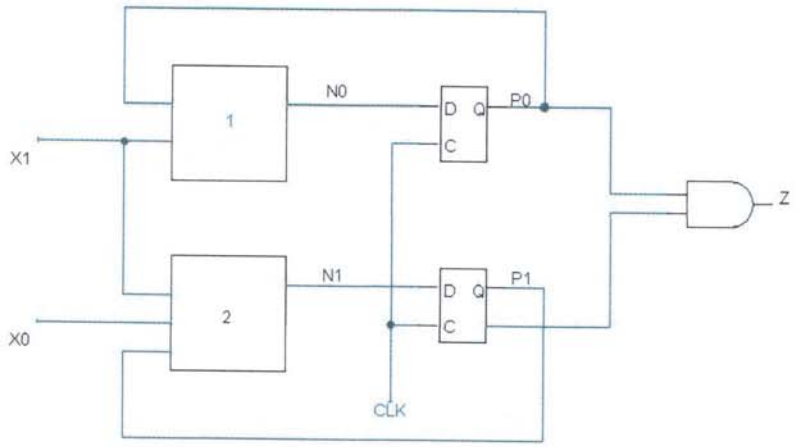
A certain finite state machine implements a pattern recognition system that recognizes specific sequences of inputs. Each input represents a letter, such as A or B. An input sequence can be written as a sequence of letters such as "ABAAABBBABA…". For this finite state machine, the output Z is 1 when an input sequence has an even number of A's and an odd number of B's. Otherwise, the output Z is 0. The states needed are named as follows:

S_{EE}	even number of A's and even number of B's
S_{EO}	even number of A's and odd number of B's
S_{OO}	odd number of A's and odd number of B's
S_{OE}	odd number of A's and even number of B's

Part A (8 points): Draw a high-level finite state machine transition diagram for the system. States and state names have been drawn for you. Add appropriate output values and all transition arcs. Label arcs as either A or B.

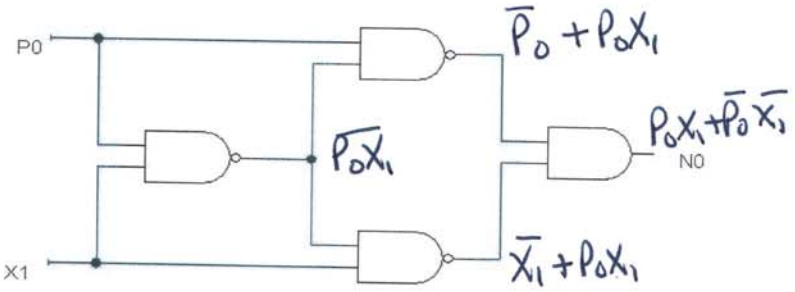


The circuit below shows an implementation of the finite state machine using two flip-flops and state representation (bits P_1P_0) given by $S_{EE} = 00$, $S_{EO} = 01$, $S_{OO} = 10$, and $S_{OE} = 11$. The blocks labeled 1 and 2 use the current state P_1P_0 and the inputs X_1X_0 to calculate the next state, N_1N_0. The letter A is represented as X_1X_0=00, and the letter B is represented as X_1X_0=01.



Problem 4, continued:

Part B (4 points): Based on the circuit below for **Block 1**, fill in the table for **N0**.



P0	X1	N0
0	0	1
0	1	0
1	0	0
1	1	1

Part C (8 points): Based on your answer to **Part B** and on the truth table for **Block 2** (shown to the left below) fill in the next state table on the right for the FSM implementation. Note that input X_1X_0=11 is not included in the table.

P1	X1	X0	N1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

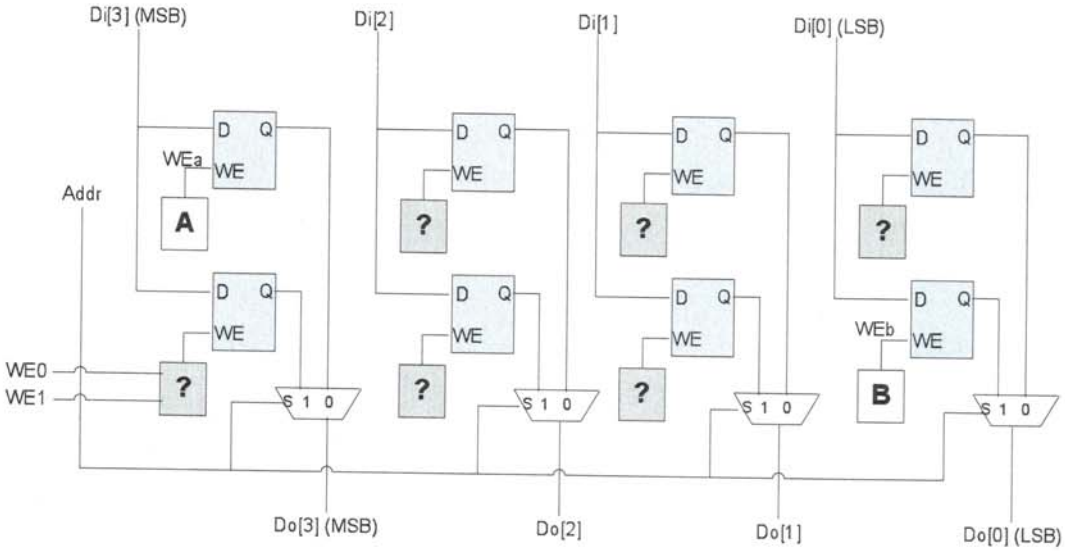
P1	P0	X1	X0	N1	N0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	1	0
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	1

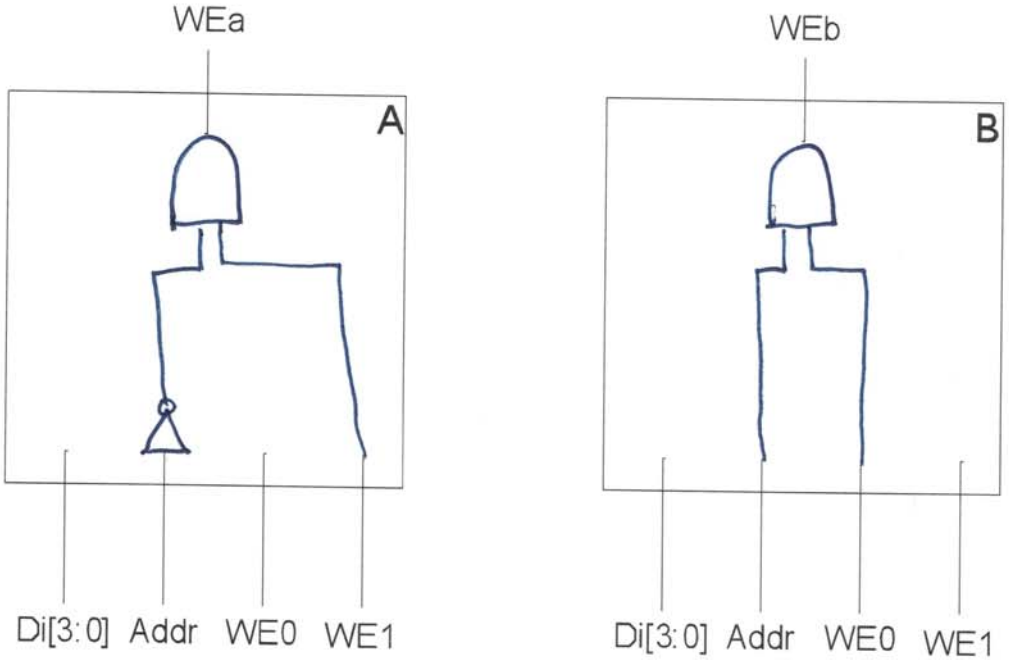
***Part D** (5 points): The input bits X_1X_0=10 represent the letter C. Explain in one sentence how you can use the next state table from **Part C** to verify that the letter C is handled correctly by the FSM implementation.

Check that $N_1N_0 = P_1P_0$ whenever $X_1X_0 = 10$
(there are four such lines in Part C,
and they satisfy the requirement).

Problem 4 (20 points): Memory

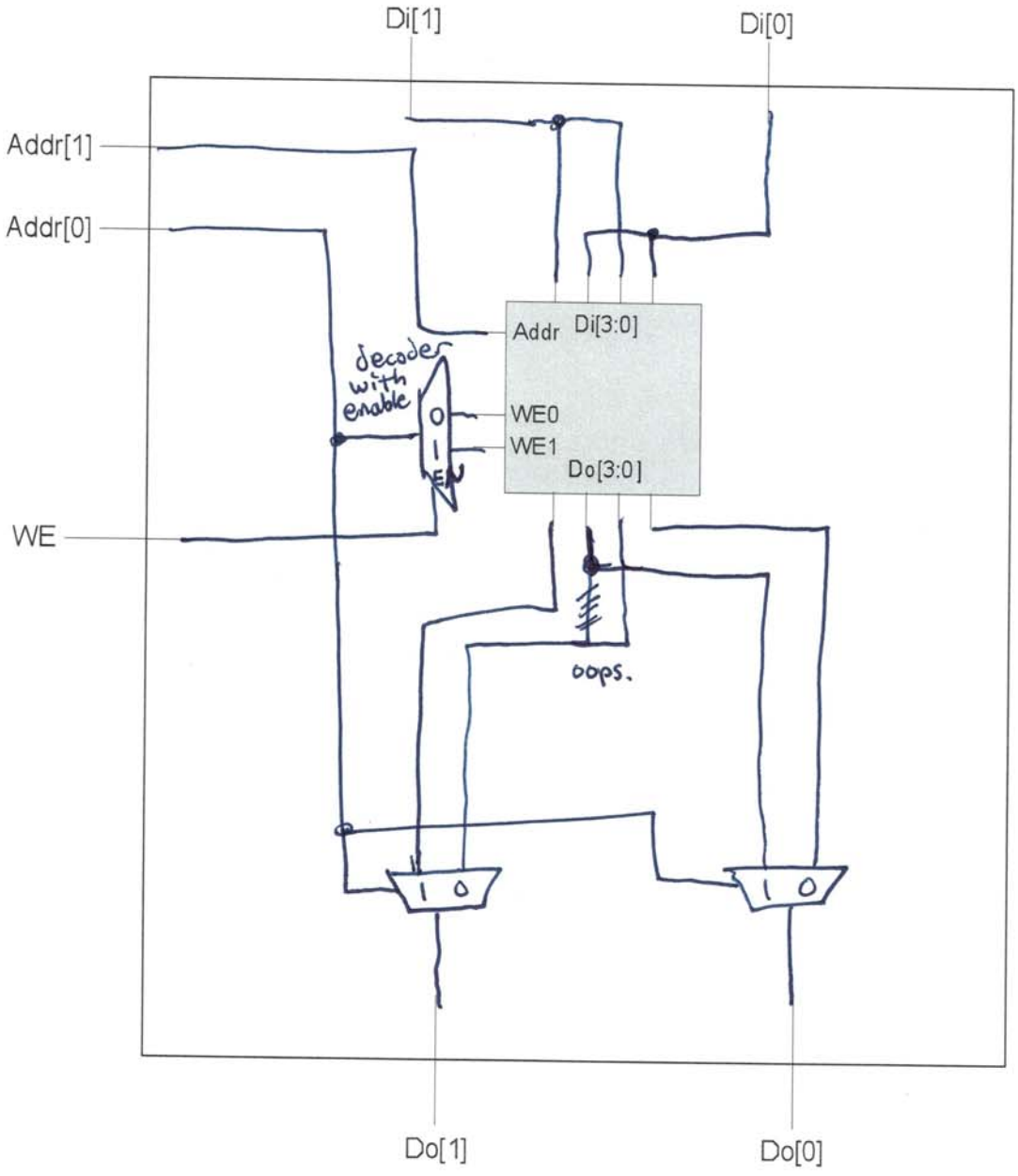
Part A (10 points): The figure below shows a $2^1 \times 4$ memory with two write-enable bits. WE0 controls writing to the two least significant bits, and WE1 controls writing to the two most significant bits. The logic for the boxes labeled "?" has been correctly implemented for you. Boxes A and B control the WE bit of the two indicated latches. Draw the combinational logic for boxes A and B in the provided space.





Problem 4, continued

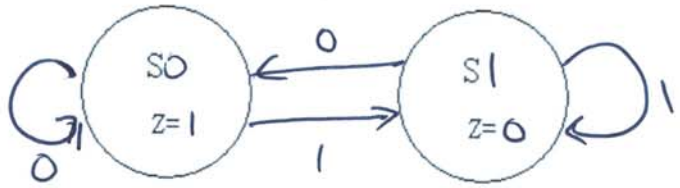
Part B (10 points): Implement a $2^2 \times 2$ memory using the $2^1 \times 4$ memory described in Part A. You may use any muxes, decoders, and gates that you deem necessary. Unnecessary complexity will result in point loss.



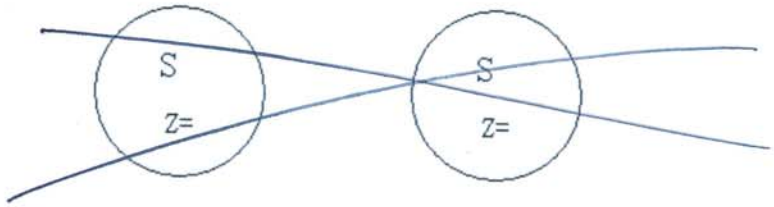
Problem 5 (18 points): Finite State Machines

For both parts of this problem, assume that the N-bit number is fed to the FSM serially, one bit at a time, with the least significant bit being fed first, and the most significant bit being fed last.

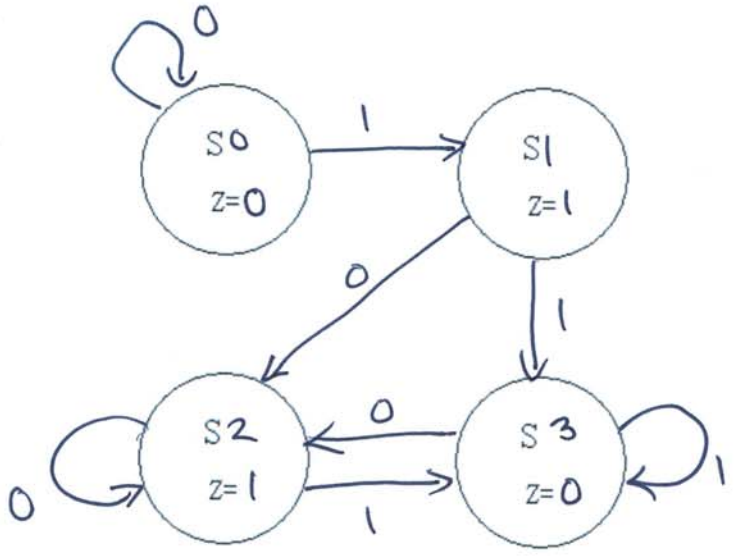
Part A (8 points): Draw a high-level finite state machine transition diagram that outputs the 1's complement of an N-bit binary number input. Please label every circle with a state number (S0, S1, etc.) and the output (Z) at that state. You may or may not need all the circles shown below. Label the arcs as either 1 or 0.



Start in either state



Part B (10 points): Draw a high-level finite state machine transition diagram that outputs the 2's complement of an N-bit binary number input. Please label every circle with a state number (S0, S1, etc.) and the output (Z) at that state. You may or may not need all the circles shown below. Label the arcs as either 1 or 0.



Start in S0

top states have "carry in" from +1

bottom states as above (PartA), after carry stops propagating

 Name: _____

Problem 3 (10 points): Memory

In this problem you will be working with 2x1-bit memory cells. Each cell stores 2 bits which can only be accessed one at a time. Which bit is accessed is determined by the address input to the memory cell. Given four 2x1-bit memory cells, build a 4x2-bit memory unit using only AND, OR, NOT gates and MUXes.