HKN ECE 120 Midterm 3 Worksheet

Serialized Design (Review)

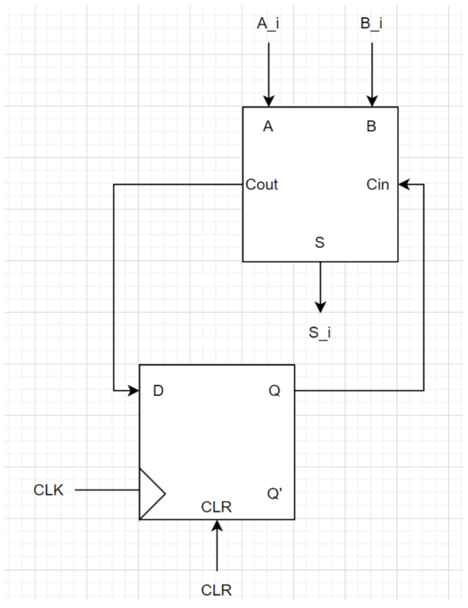
Problem 1

Consider one bit-slice of a full adder.

a. How many bits need to be passed between each bit slice? What are they?
1 bit. We need to pass the carry bit between each bit slice.

b. If we wanted to serialize this design and add 1 bit at a time, how many D flip-flops would we need to store these signals?
1. We need 1 DFF for each signal we need to store.

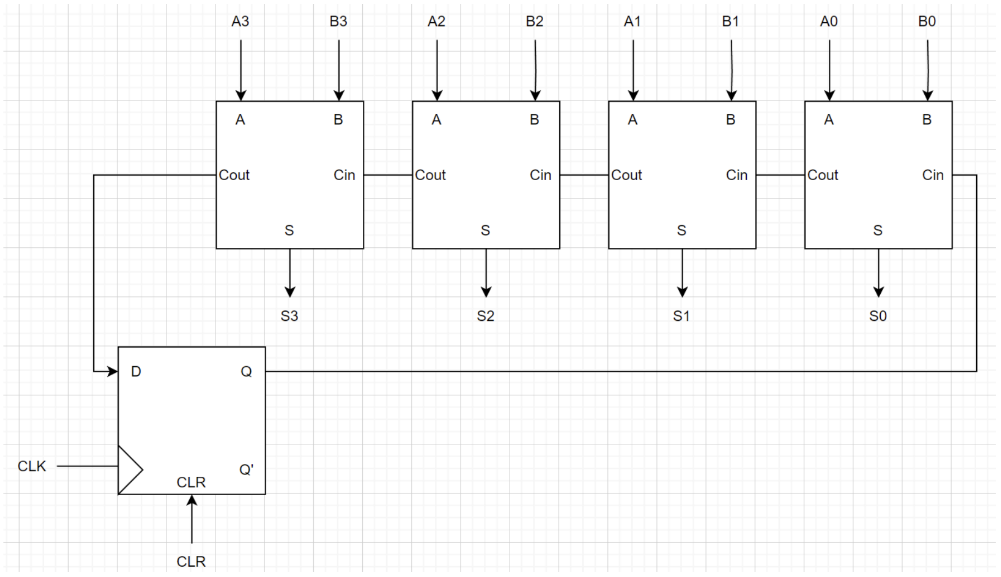
c. Implement a serialized binary adder, adding 1 bit at a time.



Note: the CLR (clear) signal is used when we want to clear the carry bit between adding different sets of numbers.

d. Assume we instead wanted to add 4 bits (1 hexadecimal) at a time. Would any signals be different?
No, we still only need to pass one carry bit, the carry bit of the MSB, to the next clock cycle's LSB.

e. Implement a serialized binary adder, adding 4 bits at a time.

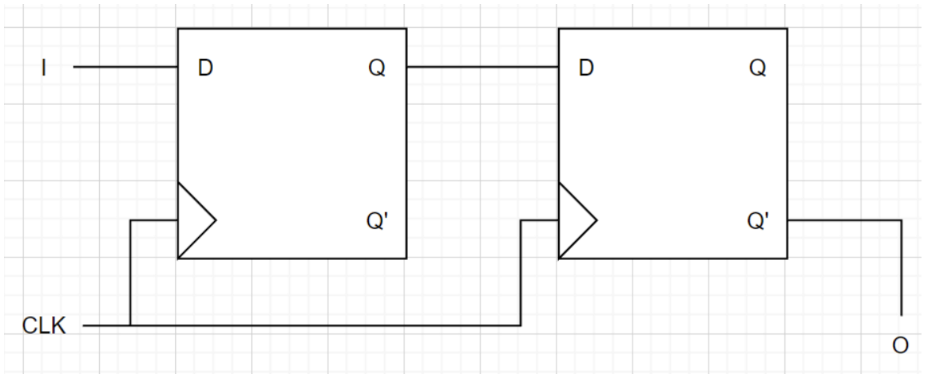


Problem 2

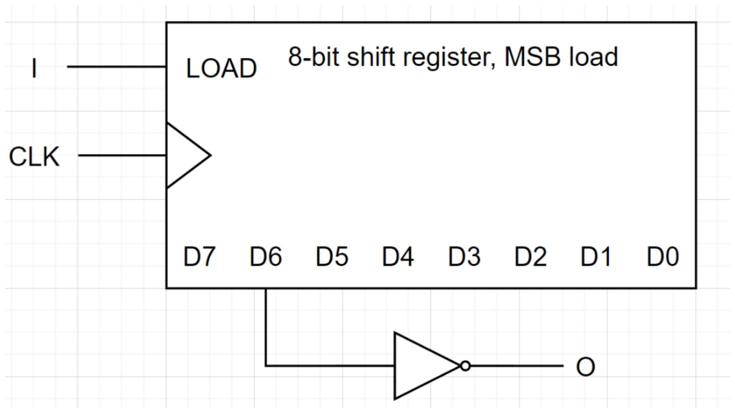
Consider a serialized circuit that takes a sequence of bits, 1 bit at a time, and outputs the same sequence of bits but is delayed by 2 bits and inverted. For example, assuming inputs and outputs are taken at the start of each clock cycle, if the input is 100101101100, then the output is XX0110100100.

a. How many bits do we need to store between each clock cycle?
2 bits.

b. Implement this circuit using D flip-flops.



c. Can we also implement this circuit using a shift register?
Yes:



Problem 3

In 50 words or less, what are the advantages and disadvantages of serialization over bit-slicing?

For larger sequences, serialized designs take much less area than bit-sliced designs. However, serialized designs are much slower, as they depend on the speed of a clock signal, while bit-sliced design is only dependent on gate delay.

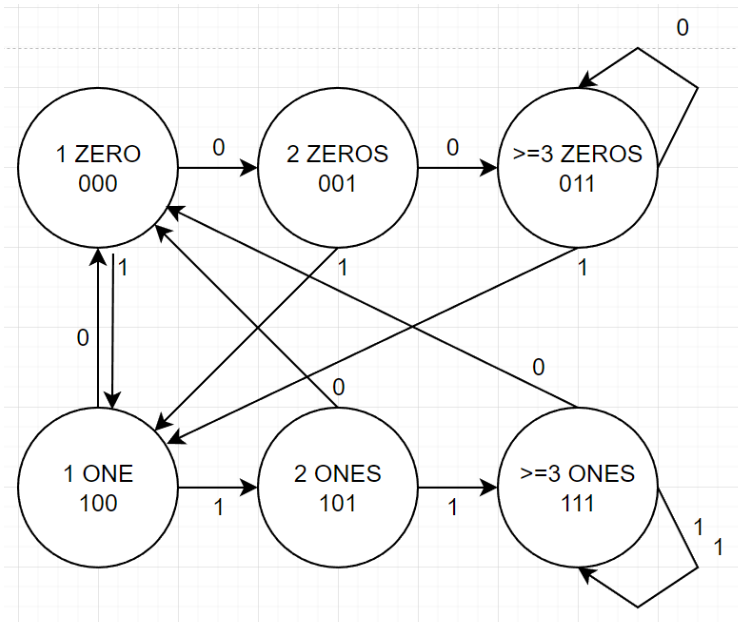
Finite State Machines

Problem 1

Suppose we wanted to create a Moore FSM that has two outputs Z and O. Z is 1 if and only if the input contains three or more consecutive zeroes, and O is 1 if and only if the input contains three or more consecutive ones.

a. How many unique states do we need for this finite state machine? How many bits do we need to represent all states? **6 states. For both 0s and 1s, we want to keep track of how many we have seen consecutively. We have one state for seeing one, two, and three or more for both 0 and 1, so 6 states. We can represent it with 3 bits, allowing for up to 8 states.**

b. Draw a state diagram for this FSM, showing all states used. **This is one possible enumeration.**

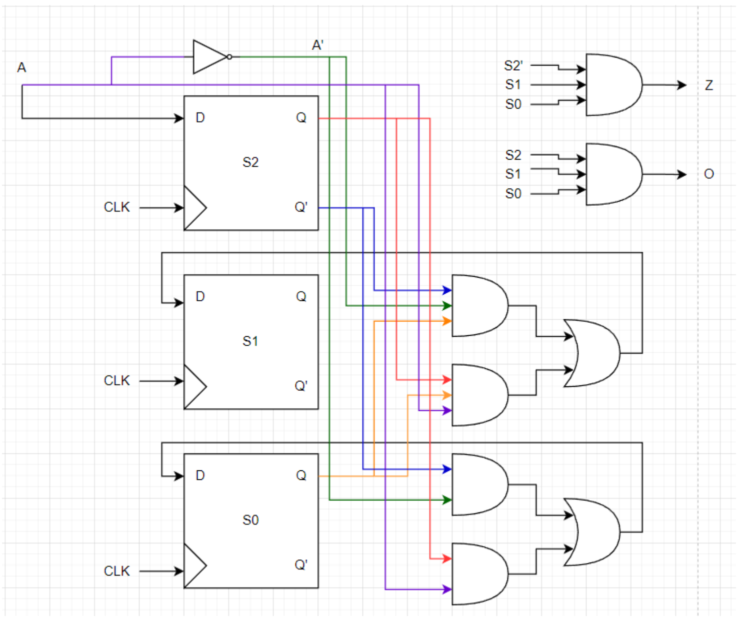


c. Create a truth table for this FSM. For unused states, write X.

S_2	S_1	S_0	A (input)	S_2^+	S_1^+	S_0^+
0	0	0	0	0	0	1
0	0	0	1	1	0	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	X	X	X
0	1	0	1	X	X	X
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	0	1	1	1	1	1
1	1	0	0	X	X	X
1	1	0	1	X	X	X
1	1	1	0	0	0	0
1	1	1	1	1	1	1

d. Use K-maps or another method to derive next-state expressions for this FSM.
$S_2^+ = A$
$S_1^+ = S_2' S_0 A' + S_2 S_0 A$
$S_0^+ = S_2' A' + S_2 A$

e. Implement this FSM using D flip-flops.

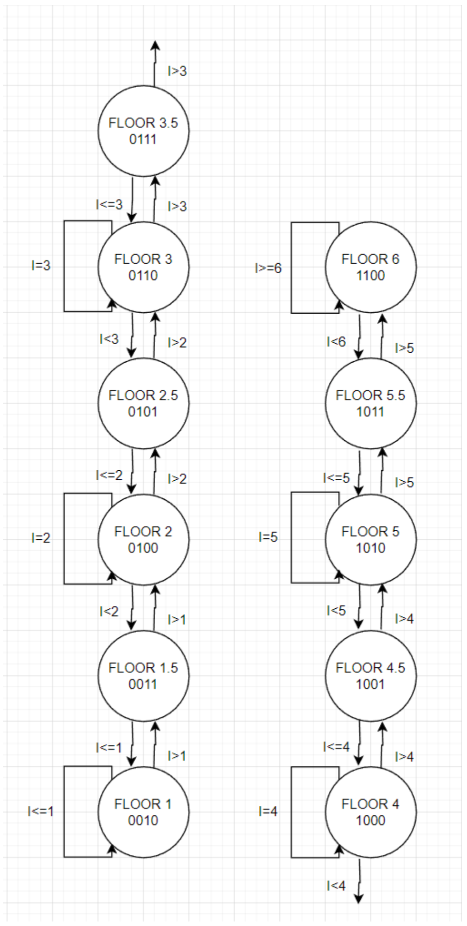


Problem 2

Suppose we wanted to create a Mealy FSM that represents an elevator in a building with 6 floors, 1 through 6. It takes two clock cycles for the elevator to move between floors. The input is a 3-bit unsigned integer $I_2I_1I_0$ and the output is a single bit F that is 1 when the elevator has reached the correct floor and 0 when it has not. You may assume that the input is always between 1 and 6 (inclusive).

a. How many unique states do we need for this finite state machine? How many bits do we need to represent all states? **We need a state for each of the floors and another state for moving between floors. Since there are 6 floors, we need 6 floors, plus 5 in-between states, so we need 11 states. We can represent this with 4 bits, allowing for up to 16 states.**

b. Draw a state diagram for this FSM, showing all states used. You may also use comparisons ($=, <, >$). (Hint: it might be helpful to enumerate the states in a way that preserves the 3-bit floor)
This is probably the easiest enumeration, but others are possible.

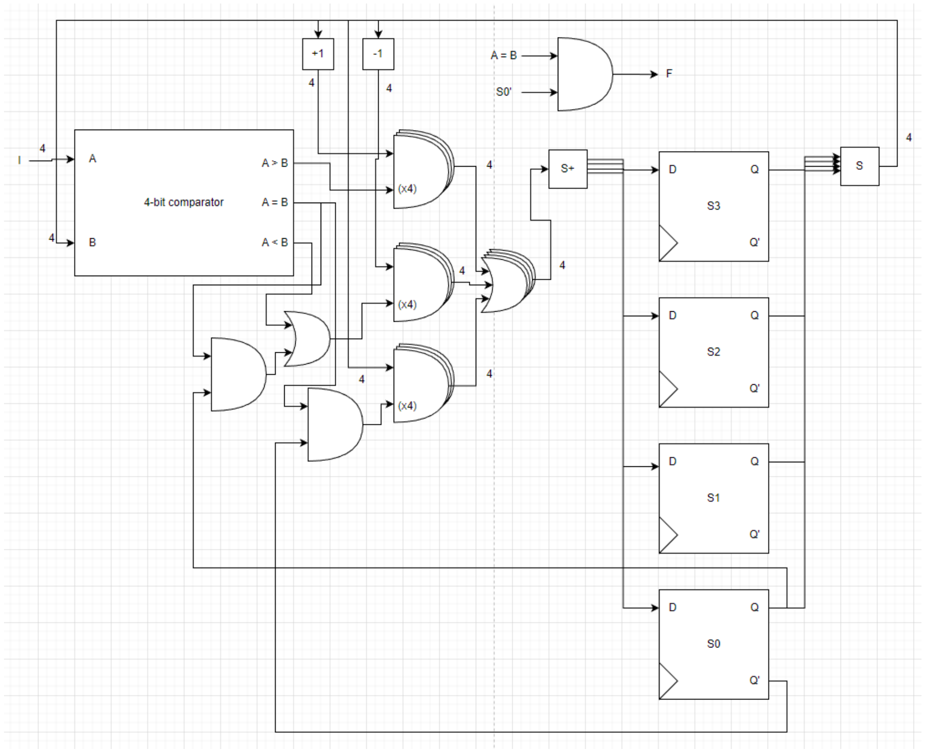


c. Create a truth table for this FSM, defining new intermediate variables for your comparisons. For unused states, write X.
$E = (I = S_3S_2S_1)$
$L = (I < S_3S_2S_1)$
$G = (I > S_3S_2S_1)$

L	E	G	S_0	S^+
0	0	0	0	XXXX
0	0	0	1	XXXX
0	0	1	0	S + 1
0	0	1	1	S + 1
0	1	0	0	S
0	1	0	1	S - 1
0	1	1	0	XXXX
0	1	1	1	XXXX
1	0	0	0	S - 1
1	0	0	1	S - 1
1	0	1	0	XXXX
1	0	1	1	XXXX
1	1	0	0	XXXX
1	1	0	1	XXXX
1	1	1	0	XXXX
1	1	1	1	XXXX

d. Use K-maps or another method to derive next-state expressions for this FSM.
$S^+ = (S+1)G + (S-1)(L + ES_0) + (S)(ES_0')$

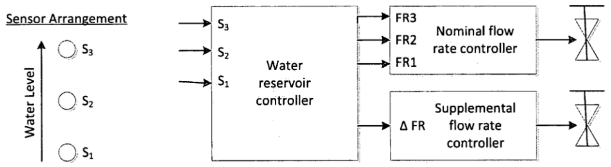
e. Implement this FSM using D flip-flops and a 3-bit digital comparator, with outputs for $A > B$, $A = B$, and $A < B$. Feel free to use adders, MUXes, buses, or any other component covered earlier in the course.



Problem 3

Taken from HDLBits "Design a Moore FSM"

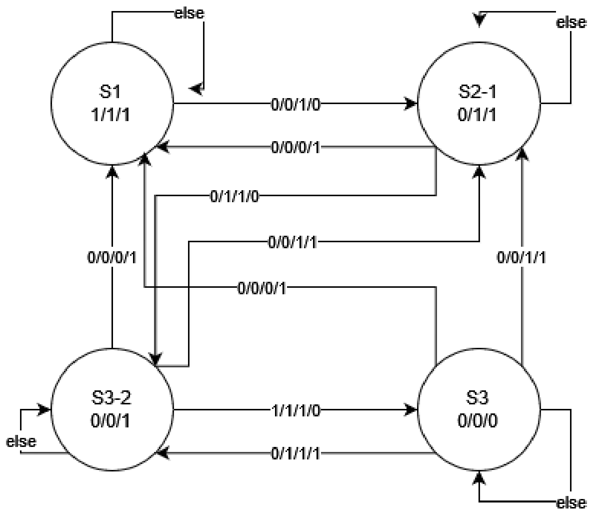
Q4. [10] A large reservoir of water serves several users. In order to keep the level of water sufficiently high, three sensors are placed vertically at 5-inch intervals. When the water level is above the highest sensor (S_3), the input flow rate should be zero. When the level is below the lowest sensor (S_1), the flow rate should be at maximum (both Nominal flow valve and Supplemental flow valve opened). The flow rate when the level is between the upper and lower sensors is determined by two factors: the water level and the level previous to the last sensor change. Each water level has a nominal flow rate associated with it, as shown in the table below. If the sensor change indicates that the previous level was lower than the current level, the nominal flow rate should take place. If the previous level was higher than the current level, the flow rate should be increased by opening the Supplemental flow valve (controlled by ΔFR). Draw the Moore model state diagram for the water reservoir controller. Clearly indicate all state transitions and outputs for each state. The inputs to your FSM are S_1, S_2 and S_3; the outputs are FR1, FR2, FR3 and ΔFR.



Sensor Arrangement
Water Level
S_3
S_2
S_1
Water reservoir controller
FR3
FR2
FR1
Nominal flow rate controller
Δ FR
Supplemental flow rate controller

Water Level	Sensors Asserted	Nominal Flow Rate Inputs to Be Asserted
Above S_3	S_1, S_2, S_3	None
Between S_3 and S_2	S_1, S_2	FR1
Between S_2 and S_1	S_1	FR1, FR2
Below S_1	None	FR1, FR2, FR3

SOLUTION



Outputs within state given in format:
FR3/FR2/FR1

Inputs/Output given in format:
S3/S2/S1/DeltaFR

Memory

Problem 1

a. What is memory addressability? What is the difference between memory addressability and a memory address?
The number of addresses, or address space, is the number of memory locations in a memory module. The addressibility is the number of bits of memory that can be accessed at once at a memory location, corresponding to a memory address.

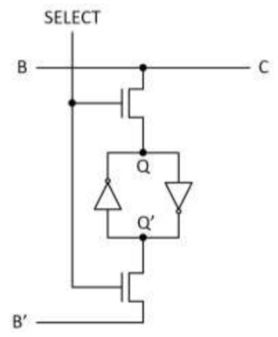
b. What is the purpose of the chip select signal in a RAM block?
The chip select signal allows for our control logic to control which memory modules are enabled, allowing us to share connections between memory modules.

c. What is the purpose of the write enable signal in a RAM block?
The write enable signal allows us to select between reading and writing on a memory module.

d. Describe the process to read from a RAM block. Detail the signal values:
Chip select should be set to high (active), and write enable should be set to low (inactive). Note that some modules show $\overline{WE}$, which indicate active low, so write enable should be set to high in that case.

e. Describe the process to write to a RAM block. Detail the signal values:
Chip select should be set to high (active), and write enable should be set to high (active). Again, if write enable is active low, it should be set to low.

Problem 2



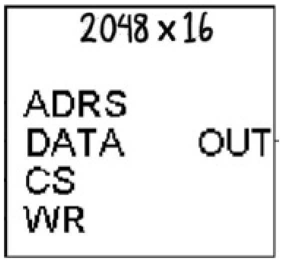
a. What should we apply to Select, B and B' in order to read the value Q to C?
SELECT = 1, B = High-Z, B' = High-Z

b. What should we apply to Select, B and B' in order to hold a value Q?
SELECT = 0, B = High-Z, B' = High-Z

c. What should we apply to Select, B and B' in order to write a 0 to Q?
SELECT = 1, B = 0, B' = 1

d. What should we apply to Select, B and B' in order to write a 1 to Q?
SELECT = 1, B = 1, B' = 0

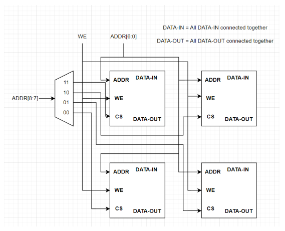
Problem 3



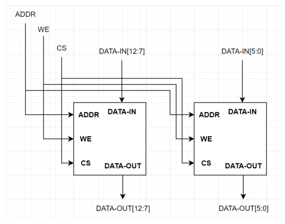
For each of the combinations below, select and draw those that can be used to build a 2048 x 12 RAM using only the parts given. If not possible, explain why. Assume that logic 0 and 1 signals are available and that the chip select and write enable are active-high.

a. Two 1024 x 6 RAMs and one 4:1 multiplexer
   Not possible. The largest we can build is a 2048 x 6 or 1024 x 12 RAM.

b. Four 512 x 12 RAMs and a 2:4 priority encoder


DATA-IN = All DATA-IN connected together
DATA-OUT = All DATA-OUT connected together
WE
ADDR[6:0]
ADDR[8:7]
11
10
01
00
ADDR
DATA-IN
WE
CS
DATA-OUT

c. Two 2048 x 6 RAMs



d. Four 512 x 3 RAMs and two 4:1 multiplexers
   Not possible. The largest we can build is a 2048 x 3 or 512 x 12 RAM.

LC-3 ISA

Problem 1

Review Questions. Answer for the LC-3 ISA.

a. How many bits are used to address memory? What is the memory address space?
The LC-3 has 16 bit memory addresses. Its address space is 2^{16}, or 65536 locations.

b. How many bits of data are stored at each memory space/what is the memory addressibility?
The LC-3 has a 16 bit memory bus. Its memory addressibility is 16 bits.

c. What is the bus width?
The LC-3 has a system bus that is also 16 bits wide. By making all bus sizes be the same, we don't have to do conversions between different sized buses.

d. How many registers are in the register file? How wide are they?
8 registers, R0 through R7. They are each 16 bits wide.

e. How many other registers are there? How wide are they?
At least 5. We have the Program Counter (PC), the Instruction Register (IR), the Memory Address Register (MAR), the Memory Data Register (MDR), control codes (NZP), and more registers for I/O (not taught in ECE 120).

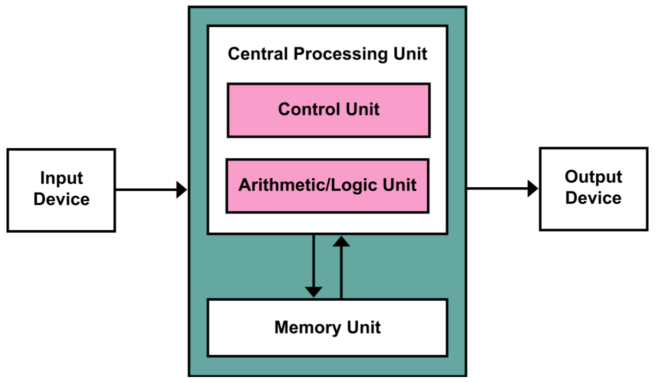
f. How do we control which signals are sent to the system bus?
Tri-state buffers are added wherever there is a connection into the system bus, allowing for control logic to ensure only one is enabled at any one time.

Problem 2

Consider the Von Neumann Architecture.

a In 50 words or less, explain the role of the control unit.
The control unit tells all other units what to do depending on the current state in control logic, which is determined by the current instruction. It holds the program counter and instruction register.

b In 50 words or less, explain the role of the ALU.
The ALU performs all of the actual computation in a von Neumann machine. Data is loaded to and from memory, and processed through the ALU.



c For the LC-3 ISA, what does the control unit correspond to?
The PC, IR, and control logic.

d For the LC-3 ISA, what are the inputs and outputs?
Keyboard input and display (character display) output.

LC-3 Assembly

Problem 1

Registers		Memory
R0: x0004	R4: x0002	x3000: x50F0
R1: x000C	R5: x000A	x3001: x0401
R1: x0005	R6: x0007	x3002: x5633
R3: x0003	R7: x0001	x3003: x12BF
PC: x3000	CC: b010	x3004: ...

1. Starting at address x3000, the LC-3 processor will continue until the instruction at x3003 finishes executing. What are the values in each of the registers below?

R0: **x0000**

R1: **x0004**

R2: **x0005**

R3: **x0003**

PC: **x3004**

CC: **b001**

2. Write a complete list of the sequence of values taken by the MAR register as the LC-3 processes these instructions. Use only as many lines as are necessary.

#1: x3000 (initial value)	#5:
#2: **x3001**	#6:
#3: **x3002**	#7:
#4: **x3003**	#8:

No memory loading operations (LD, ST, LDR, STR, etc), thus MAR only sees the PC value

Problem 2

Write an LC-3 Assembly program that compares two numbers in R2 and R3 and puts the larger number into R1. If the numbers are equal, then R1 is set equal to 0. Use as many lines as necessary.

Location	**Assembly**	**Comments**
x3000	NOT R2, R2	Set R2 to -R2 (step one)
x3001	ADD R2, R2, #1	Set R2 to -R2 (step two)
x3002	ADD R2, R3, R2	Set R2 to R3 - R2
x3003	BRn #3	If the result is negative, R2 >R3
x3004	BRp #4	If the result is positive, R3 >R2
x3005	AND R1, R1, #0	Else, set R1 to 0
x3006	HALT	
x3007	ADD R1, R2, #0	If R2 >R3, Set R1 to R2
x3008	HALT	
x3009	ADD R1, R3, #0	If R3 >R2, Set R1 to R3
x300A	HALT	

Problem 3

Write two LC-3 Assembly programs that executes a bitwise OR operation and a bitwise XOR operation respectively on R4 and R5 and puts the result in R1. How might you extend these to support NOR and XNOR?

OR

Location	**Assembly**	**Comments**
x3000	NOT R4, R4	R4 <= ~R4
x3001	NOT R5, R5	R5 <= ~R5
x3002	AND R1, R4, R5	R1 <= R4'R5'
x3003	NOT R1, R1	R1 <= (R4'R5')' = R4 + R5

XOR

Remember that $p \bigoplus q = (p+q)(pq)'$

Location	**Assembly**	**Comments**
x3000	AND R3, R4, R5	R3 <= R4R5
x3001	NOT R4, R4	R4 <= R4'
x3002	NOT R5, R5	R5 <= R5'
x3003	AND R2, R4, R5	R2 <= R4'R5'
x3004	NOT R2, R2	R2 <= (R4'R5')' = R4 + R5
x3005	NOT R3, R3	R3 <= (R4R5)'
x3006	AND R1, R2, R3	R1 <= (R4 + R5)(R4R5)'

NOR and XNOR would be adding an additional instruction NOT R1, R1 to negate the result.

Problem 4

a. Assuming LC-3 now has 32 registers, we want to increase the number of registers that we can specify in the LC-3 ADD instruction to 32. Is there a problem with this? Explain.
Yes, specifying 32 registers would require 5 bits per field, meaning 5 bits for SR1, 5 for SR2 and 5 for DR in the case of the 2 source version. This leaves only 1 bit for the opcode, which is not possible. If we were to extend the immediate version instead, we can fit the opcode, SR and DR into 14 bits, but this leaves only 2 bits for the immediate, which is extremely inconvenient at best.

b. A memory's addressibility is 64 bits. What does that tell you about the size of the MAR and MDR, given a 64-bit ISA and 2^{20} memory locations?
64-bit addressibility means MDR must be 64 bits, as one address stores 64 bits of data. MAR can stay 20 bits as that is all that is needed to access one location.

c. Say we have a memory consisting of 256 locations, and each location contains 16 bits. How many bits are required for the address for a byte-addressable system? Explain.
$256 = 2^8$**, so 8 bits for the address.**
1 byte is 8 bits, $16/8 =$ **2 bits to offset into one location.**
So $8 + 2 =$ **10 bits are needed in total.**