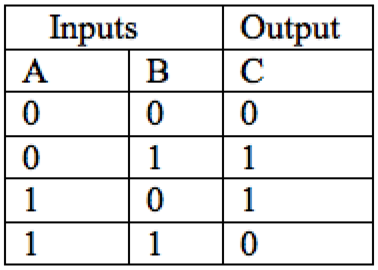
Name: Mike Xie

Computer Science 605.611

Problem Set 3

Module 3 defines and describes basic logic gates such as AND, OR, XOR, NOT, etc.

1. The following truth table defines the output C as a function of the two inputs A and B:



Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

a) (5) Write down the expression for the logical product of the zero maxterms.
Maxterms were defined in module 3.

Zero Maxterms are sum terms that equal to 0.
Then to get the logical product of the zero maxterms, we need to do a product of all the sum terms that equal to 0.

Here is a truth table of the Max Terms

Inputs		Outputs	Maxterms			
A	B	C	M0	M1	M2	M3
0	0	0	0	1	1	1
0	1	1	1	0	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	0

We will have two maxterms
M0 = (A + B)
M3 = (A' + B')

So, the product of the zero maxterms will be the following
C (A, B) = AND (M0, M3)
C (A, B) = (A + B) (A' + B')

b) (5) Use only Boolean identities (not truth tables) to show that the logical sum of the non-zero minterms for this function is equivalent to the logical product of the zero maxterms. Minterms and maxterms were defined in module 3.

In this answer, I will show that the logical sum of non-zero minterms is equivalent to the logical product of zero maxterms.

Remember that the product of maxterms was the following
MaxtermFunction (A, B) = (A + B) (A’ + B’)

The sum of minterms will be the following
MintermFunction (A, B) = (A’B) + (AB’)

From the Maxterm, I will use Boolean identities to get the Minterm to show that they are equivalent.
MaxtermFunction = (A + B) (A’ + B’)

Function	Property
(A + B) (A’ + B’)	Current Maxterm
((A + B) A’) + ((A + B) B’)	Distributive Law OR Form
(A’ (A + B)) + (B’ (A + B))	Commutative Law AND Form
(A’A + A’B) + (B’A + B’B)	Distributive Law OR Form
(0 + A’B) + (B’A + 0)	Inverse Law AND Form
(A’B) + (B’A)	Identity Law OR Form
(A’B) + (AB’)	Commutative Law AND Form

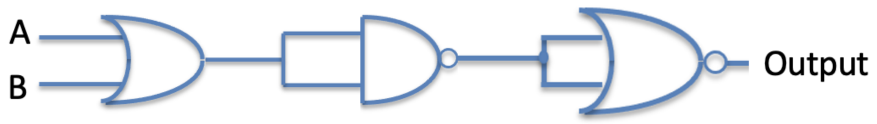
The final result of applying Boolean identities to the logical product of zero Maxterm is the following function.

(A’B) + (AB’)

We can see that this is clearly the logical sum of the non-zero Minterm function.
Therefore, I have shown that the Maxterm is equivalent to the Minterm, using Boolean identities.

2. (5) Which one of the following individual logic gates is equivalent to the circuit shown below? That is, given the same two single-bit data inputs A and B, which one of the 6 logic gates listed below generates the same output as the circuit?

a) AND	d) NAND
b) OR	e) XNOR
c) NOR	f) XOR



First, we have two variables A and B.
We apply an OR operation to get an output.
We use that output as input for the AND operation.
Then we apply a NOT operation to the output.
Then taking that output, we apply an OR operation.

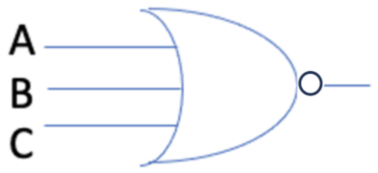
Lastly, we apply a NOT operation and that would be our final output.

Let's build a truth table

Input		Intermediate Steps				Output
A	B	A + B	(A + B) (A + B)	((A + B) (A + B))'	((A + B) (A + B))' + ((A + B) (A + B))'	(((A + B) (A + B))' + ((A + B) (A + B))')'
0	0	0	0	1	1	0
0	1	1	1	0	0	1
1	0	1	1	0	0	1
1	1	1	1	0	0	1

We can see that this is clearly equivalent to an OR gate.
So, the Answer must be **B) OR**

3 a) (5) Show how the following 3-input NOR gate can be implemented using a minimum number of 2-input NOR gates. Your solution should not be a logic expression; it should contain only NOR gates with two inputs.



Remember that the NOR gate for 3 inputs has the following truth table

Inputs			Intermediate Step	Output
A	B	C	OR	NOR
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

If we can first get the OR of 2 variables and get a single intermediate output, we can NOR the single intermediate output with a third variable to get the NOR of all 3 variables.

We can get the OR of 2 variables but getting the inverse of a NOR of 2 variables.

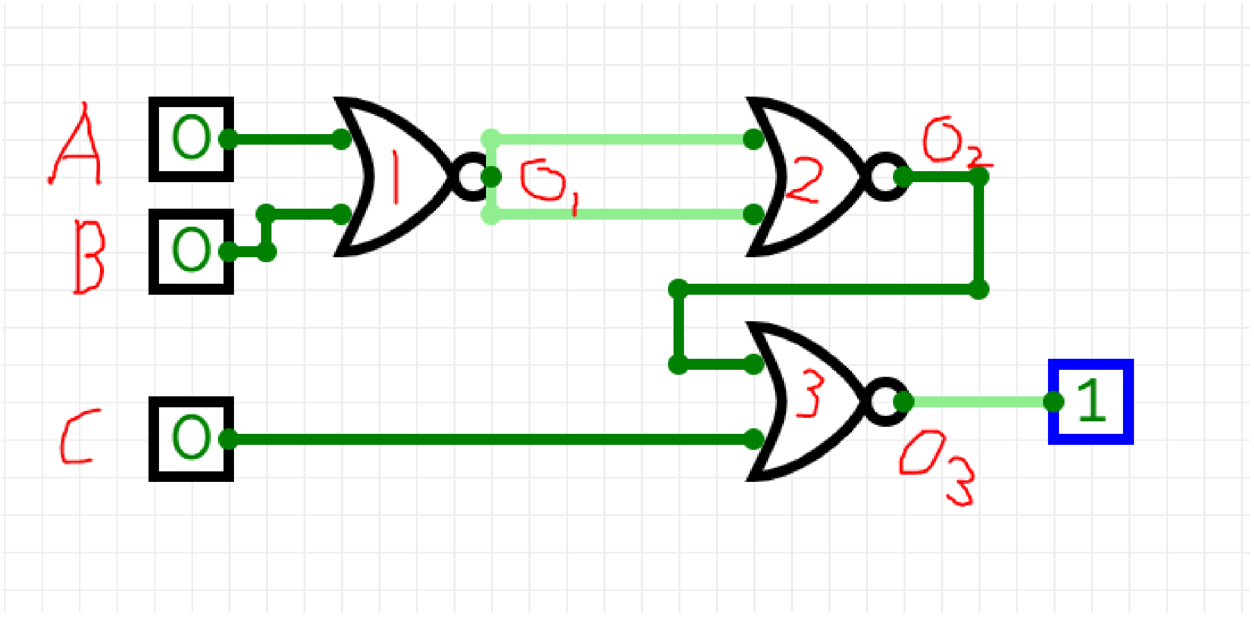
We can get the inverse of a NOR of 2 variables by applying a NOR of the output of a NOR of two variables.

Here is the truth table but with only 2-input NOR gates, to show that it works.

Inputs			Intermediate Step			Output
			NOR	Inverse of the output from a NOR, which is equal to an OR	NOR of an OR of 2 variables (as the first input) with a third variable (as the second input)	
A	B	C	NOR (A, B)	NOR (NOR (A, B), NOR (A, B))	NOR (NOR (NOR (A, B), NOR (A, B)), C)	Final output
0	0	0	1	0	1	1
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	1	0	1	0	0
1	0	0	0	1	0	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	0	1	0	0

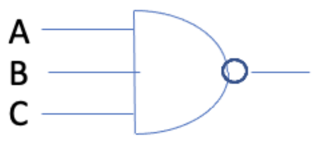
So, you can see that we only need 3 2-input NOR gates to equal 1 3-input NOR gate.

Here is a picture of the 3 NOR gates at a glance.



So therefore, we only need 3 2-input NOR gates to make 1 3-input NOR gate..
In the picture, Gate 1 and Gate 2 will give us an OR of variables A and B. Let's call this output variable O2.
Gate 3 will give us a NOR of the output of variable O2, and C. Which will, in the end, give us a NOR of 3 variables.

b) (5) Show how the following 3-input NAND gate can be implemented using a minimum number of 2-input NAND gates. Your solution should not be a logic expression; it should contain only NAND gates with two inputs.



Remember that the NAND gate for 3 inputs has the following truth table

Inputs			Intermediate Step	Output
A	B	C	AND	NAND
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

If we can get the AND of 2 variables, to get one output.
We can later use the NAND gate of the output and the third variable to get the final answer.

We can get the AND of 2 variables by using a NAND and then doing the inverse of that output using another NAND.

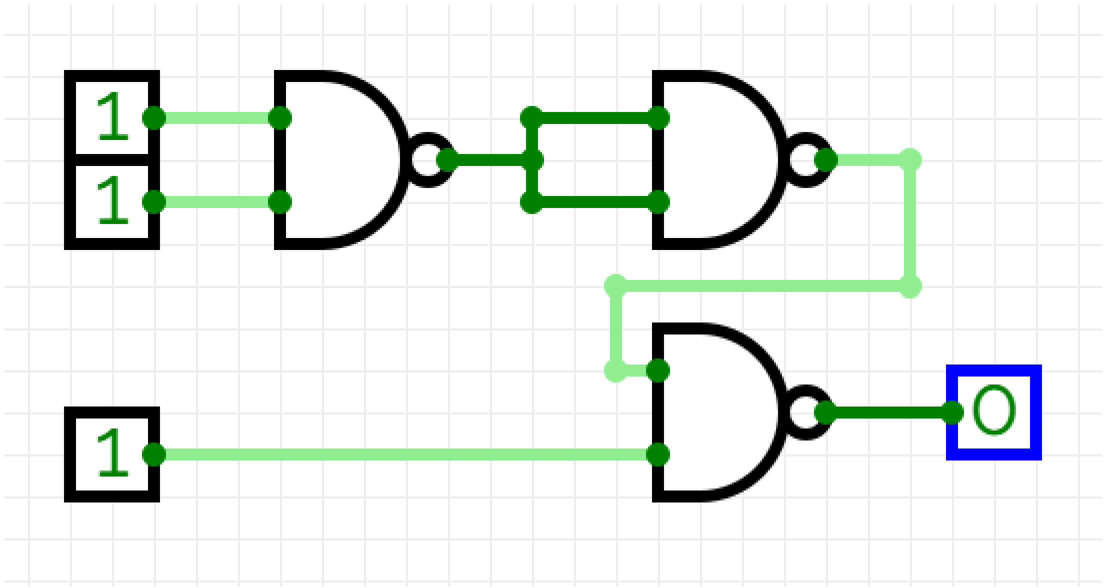
Here is the truth table but with only 2-input NAND gates to show that it works.

Inputs			Intermediate Step			Output
			NAND	Inverse of the output from a NAND, which is equal to an AND	NAND of a NAND of 2 variables (as the first input) with a third variable (as the second input)	
A	B	C	NAND (A, B)	NAND (NAND (A, B), NAND (A, B))	NAND (NAND (NAND (A, B), NAND (A, B)), C)	Final output
0	0	0	1	0	1	1
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	0	1	1	0	1	1
1	1	0	0	1	1	1

1	1	1	0	1	0	0

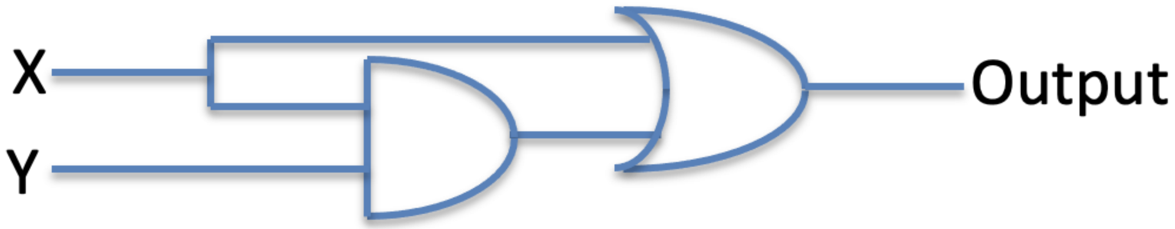
In the final output, you can see that it is equivalent to a NAND gate for all 3 variables.
So, you can see that we only need 3 2-input NAND gates to equal 1 3-input NAND gate.

Here is a picture of the 3 NAND gates at a glance.



So therefore, we only need 3 2-input NAND gates to make 1 3-input NAND gate.

4. (5) Consider the logic circuit shown below. Write down a <u>simplified</u> logic expression for Output as a function of the inputs X and Y. Show how you obtained your answer.
Output = ____X________



Lets make a truth table

Input		Intermediate Steps		Output
X	Y	XY	XY + X	XY + X
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1

1	1	1	1	1

From the truth table we can see that the Output Function should be XY +X

We can try to simplify it down by using Boolean identities.

Function	Property
XY + X	Current Function
X (Y + 1)	Factor out X
X (1)	Null Law OR Form
X	Identity Law AND Form

So, the output function in simplest form, is just X

5. a) (5) Register $8 contains a 32-bit machine instruction that computes the sum of its immediate operand and register $6. Show a series of MIPS true-op assembly language instructions that places the 32-bit sign-extended version of the machine instruction's immediate operand into register $5. No register other than $5 should be modified.

So, we want to extend the sign. An intermediate operand has 16 bits. A register has 32 bits.

To extend the sign, we would need to eheck the most significant bit of the 16 bit intermediate operand, and extend that sign bit all the way to the left. This is how we can extend the sign bit of 16 bits to 32 bits.

The sra true-op instruction has a way of doing this. The sra instruction is a right arithmetic shift and automatically extends the sign.

We shift left 16 times.
Then do arithmetic shift right 16 times.
That is how we will extend the sign bit.

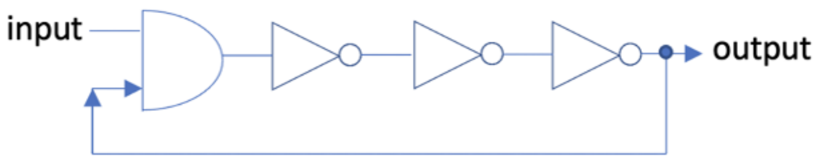
The true-op machine instructions is as follows.
```
sll $5, $8, 16
sra $5, $5, 16
```

b) (5) Write down one or more integer arithmetic MIPS true-op instructions that will generate the integer quotient generated by dividing register $2 into register $3. The quotient should be placed into register $7. No CPU register other than $7 should be modified.

div saves the quotient to LO register
mflo moves the data from the LO register to another register.

So, the MIPS true-op instructions are the following.
```
div   $3, $2
mflo  $7
```

6. a) (5) Is the circuit shown below a combinational circuit or is it a sequential circuit? Explain your answer.



Because this circuit loops back into itself, it must be a sequential circuit.

The looping back of the output back into the input allows the circuit to remember the output. Making this a sequential circuit,

b) (5) Assume each gate in the circuit show above in part a) takes 1 ns to produce the result on its output line in response to a value on its input line. That is, the propagation delay for each gate is 1 ns. As shown in the table below, the values for both Input and Output are 0 at time 0.
The table shows the value of Input for the next 12 time intervals. Fill in the column for Output to show its resulting value for each of the time intervals.

Time	Input	Output	Current Gate
0	0	0	Input
1	0	0	AND
2	0	1	NOT
3	0	0	NOT
4	0	1	NOT
5	0	0	AND
6	0	1	NOT
7	0	0	NOT
8	1	1	NOT
9	1	1	AND
10	0	0	NOT
11	0	1	NOT
12	0	0	NOT

7. (5) The truth table on the left below corresponds to a half adder where A and B are the two input data bits, X is the sum and Y is the carry. Complete the truth table on the right by filling in the entries to make the table correspond instead to a half subtractor. For the half subtractor, X is the difference (A - B) and Y is the borrow.

A	B	X	Y

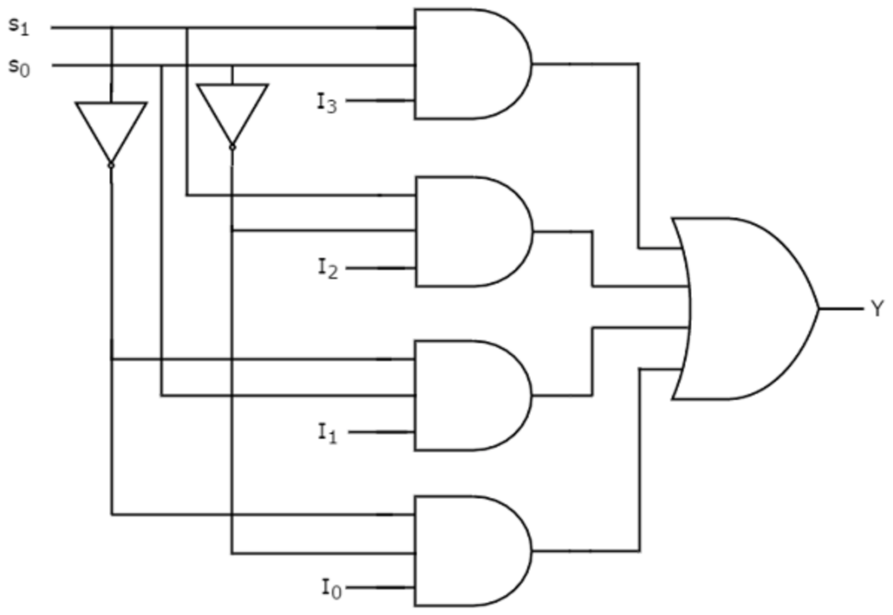
A	B	X	Y

0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

8. (5) For the circuit shown below, explain how the output Y is related to the inputs. Which one of the following devices does the circuit implement?

- a decoder
- an encoder
- a multiplexer



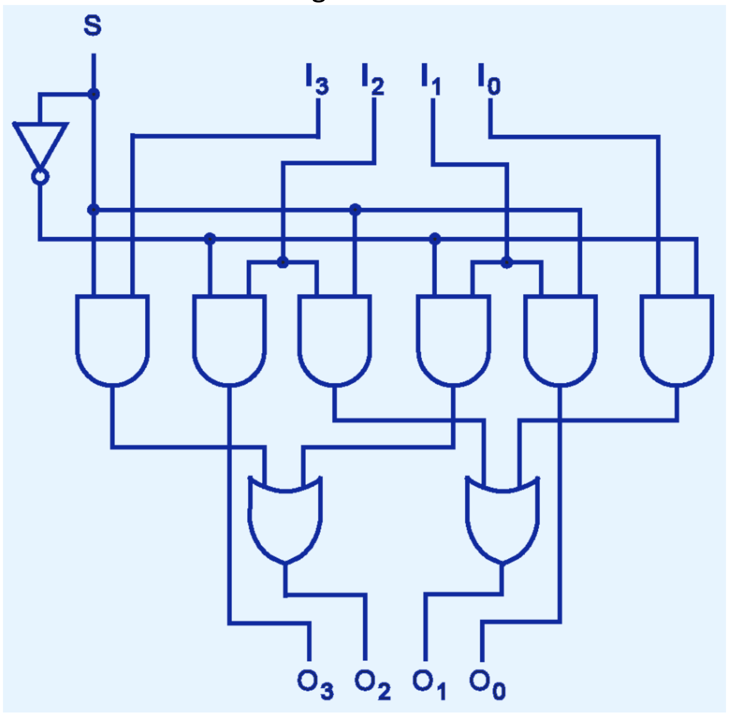
This seems like a 4-to-1 multiplexer with 2 select inputs.

We are given 4 inputs and 2 select inputs, and we output one Y bit.

Depending on the selector inputs, we will select one of the inputs as the output.

If s1s0 = 00, we output i0 for y
If s1s0 = 01, we output i1 for y
If s1s0 = 10, we output i2 for y
If s1s0 = 11, we output i3 for y

9. Consider the following circuit:



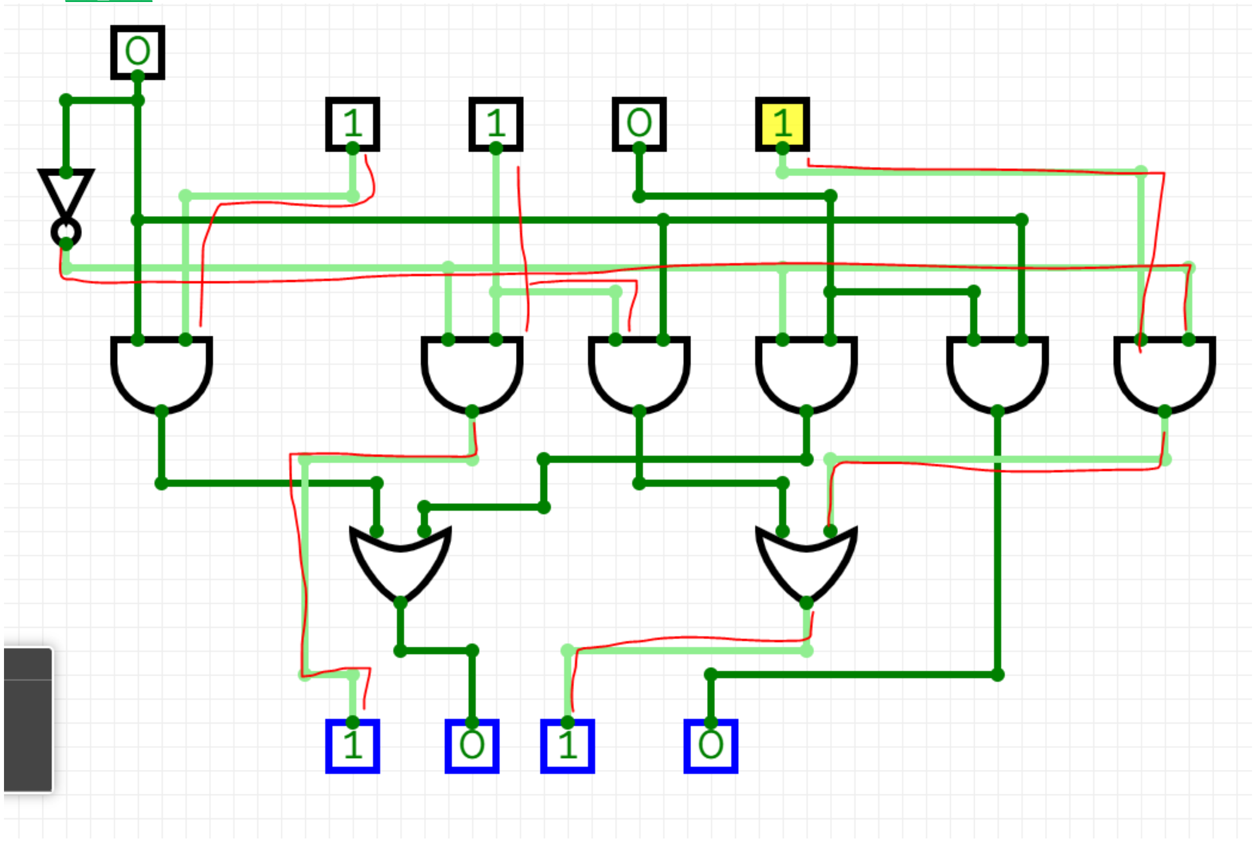
a) (5) Is this a combinational circuit or is it a sequential circuit? Explain your answer.

This is a combinational circuit. There are no feedback loops, and no flip flops to store memory. Because there are only logic gates, it must be a combinational circuit.

b) (5) If S=0, show the value for each of the 4 outputs if initially
I_3, I_2 , I_1 I_0 = 1101.

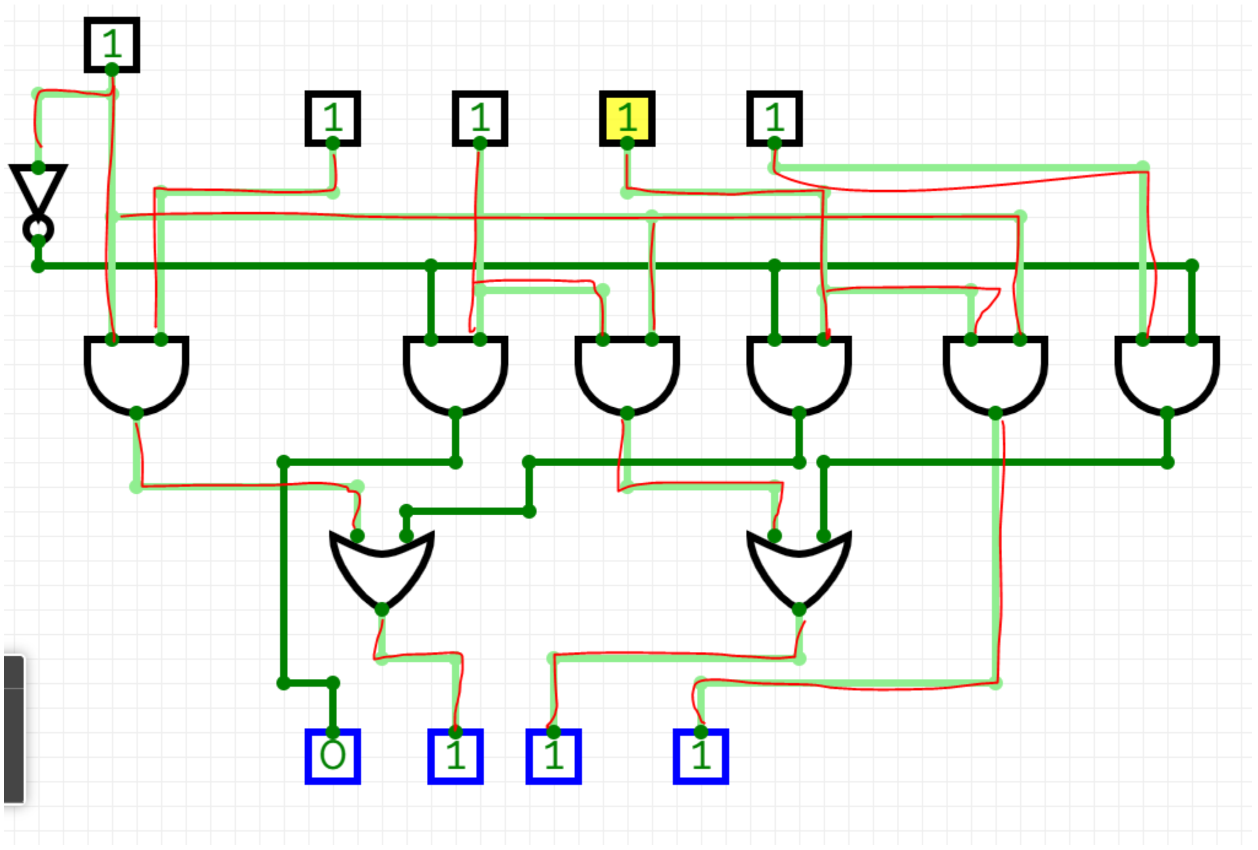
O_3 = __1____

O_2 = ___0___
O_1 = ___1___
O_0 = ___0___

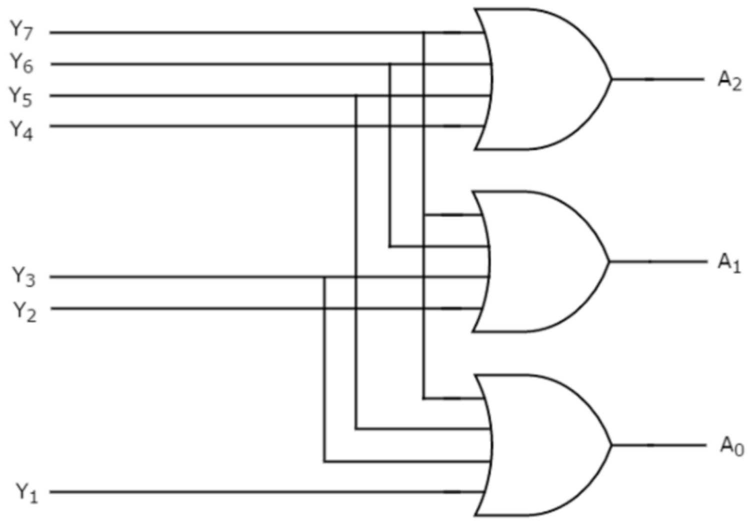


c) (5) If S=1, show the value for each of the 4 outputs if initially I_3, I_2, I_1 I_0 = 1111.

O_3 = ___0___
O_2 = ___1___
O_1 = ___1___
O_0 = ___1___


1
1
1
1
1
0
1
1
1

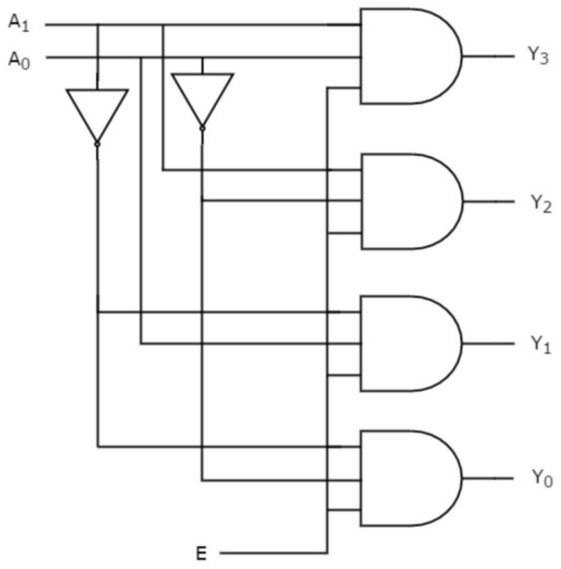
10. (5) Complete the table below to show the outputs A2 A1 A0 as a function of the inputs Y7, Y6, Y5, Y4, Y3, Y2 and Y1. Of the choices: decoder, encoder or multiplexer, which one does the circuit implement.?



Y7	Y6	Y5	Y4	Y3	Y2	Y1	A2	A1	A0
0	1	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	1	0	0	1	0

This circuit implements an encoder. More inputs than outputs.
A 7 to 3 encoder.

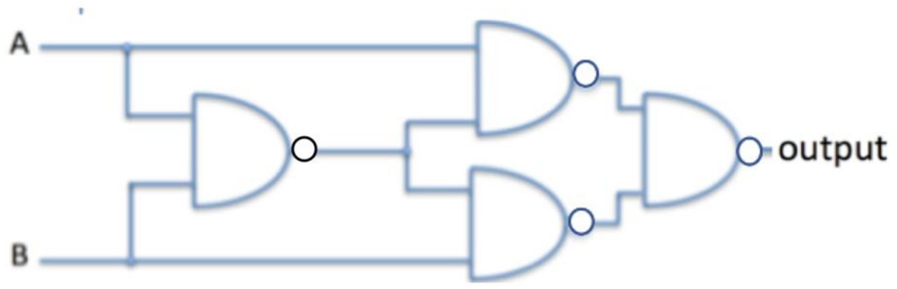
11. (5) In the circuit shown below, A1 & A0 are the data inputs and E is a control. Complete the table to show the outputs Y3, Y2, Y1, and Y0 for each combination of the inputs A1, A0 and E. Which of the following devices does this circuit implement? Explain your answer.



A1	A0	E	Y3	Y2	Y1	Y0
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0

This is a decoder circuit. It has more outputs than inputs. It also has an enable bit.
A 2 to 4 decoder.
If the enable bit is 0, it forces all outputs to equal 0.

12. (5) Consider the logic circuit shown below:



Which one of the following individual logic gates is equivalent to this circuit? That is, which one of the 6 logic gates listed below generates the same output as the circuit above with A and B as its two inputs?

a) AND	d) NAND
b) OR	e) XNOR
c) NOR	f) XOR

The truth table of the circuit is the following.

Inputs		Intermediate Steps				Output
A	B	NAND (A, B)	NAND (A, NAND (A, B))	NAND (B, NAND (A, B))	NAND (NAND (A, NAND (A, B)), NAND (B, NAND (A, B)))	Final Output
0	0	1	1	1	0	0
0	1	1	1	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0

Obviously from the truth table, we see that the circuit is equivalent to XOR
So, the answer is f) XOR

13. (5) Write down a single MIPS instruction to place into register $7, the quotient of the signed integer in register $4 divided by the constant 256. Your solution should not use any of the MIPS division instructions.

Because 2^8 = 256

Dividing a signed 32-bit integer by 256 would be the same as performing an arithmetic right shift by 8 bits.

The following would be the MIPS instruction.
sra $7, $4, 8

14. (5) Let b31, b30, b29, . . . b1, b0 correspond to the 32 bits within a CPU register. The unsigned integer represented by this 32-bit pattern corresponds to the following polynomial in powers of two:

P_unsigned = b31*(2^31) + b30*(2^30) + b29*(2^29) + . . . + b1*(2^1) + b0*(2^0)
where, of course, each bn is either 0 or 1

The two's complement signed integer represented by the 32-bit pattern corresponds to the following polynomial in powers of two where the leftmost bit (b31) is multiplied by –(2^31):

P_signed = b31*[-(2^31)] + b30*(2^30) + b29*(2^29) + . . . + b1*(2^1) + b0*(2^0)

Evaluate P_unsigned and P_signed for the 32-bit pattern:
11000000000000110100001000111010
and express your answers in decimal.

P_unsigned = ________3,221,439,034________

P_signed = __________-1,073,528,262_________

1100 0000 0000 0011 0100 0010 0011 1010
P_unsigned = 2^31 + 2^30 + 2^17 + 2^16 + 2^14 + 2^9 + 2^5 + 2^4 + 2^3 + 2^1
P_unsigned = 3,221,439,034

-1073528262

P_signed = -(2^31) + 2^30 + 2^17 + 2^16 + 2^14 + 2^9 + 2^5 + 2^4 + 2^3 + 2^1
P_signed = -1,073,528,262