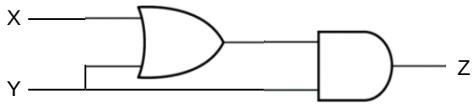
Practicing Boolean gates

Problem 1
Find the truth table for the following circuit:

X	Y	Z
0	0	1
0	1	0
1	0	1
1	1	1

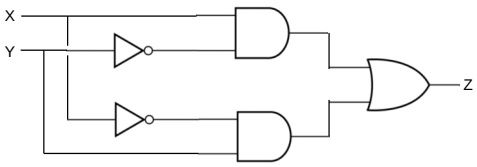
Problem 2
Find the truth table for the following circuit:



X	Y	Z
0	0	0
0	1	1
1	0	0
1	1	1

Note that Z is always equal to Y. Therefore, this circuit can be simplified by directly connecting Y to Z with a wire (ignoring X).

Problem 3
Find the truth table for the following circuit:



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

This behavior is known as the exclusive OR, XOR. If Z=X XOR Y, then Z is one if either X is 1 or Y is 1, but not both. In general, XOR produces a 1 if the number of 1's in the inputs is odd (that's why it is also called parity). The XOR logic is useful, so it was given a symbol:

Problem 4
Given a truth table, find a circuit that produces it.

X	Y	Z
0	0	1
0	1	1
1	0	0
1	1	1

We will illustrate the process step by step.

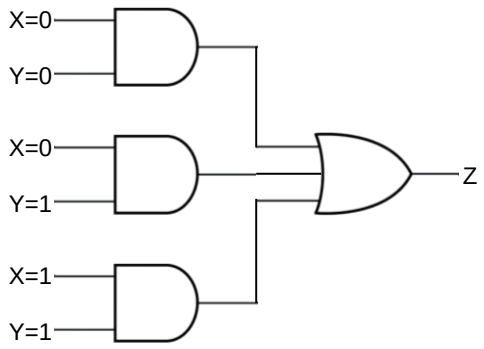
1. Identify all the rows in the truth table that have a 1 for the output (ignore the others):

X	Y	Z
0	0	1
0	1	1
1	1	1

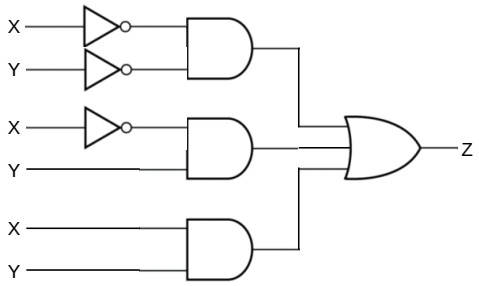
2. Say what you see in English using the every day logic:

   Z is 1 if (X=0 AND Y=0) OR (X=0 AND Y=1) OR (X=1 AND Y=1)

3. Transform the English sentence into a block diagram:

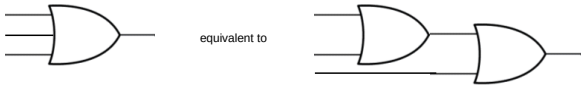


4. Replace all the 0's with inverters (NOT gates)



5. Voila!

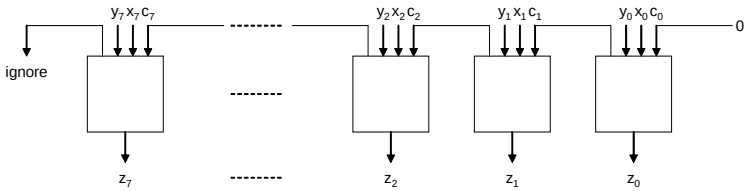
Note that the OR gate in this example is taking three inputs. This is fine because we can always create an OR gate with multiple inputs by cascading 2-input OR gates.



The same is true for AND and XOR gates.

Problem 5
Build a circuit to add two numbers in 2's complement (this means ignore the left-most carry).

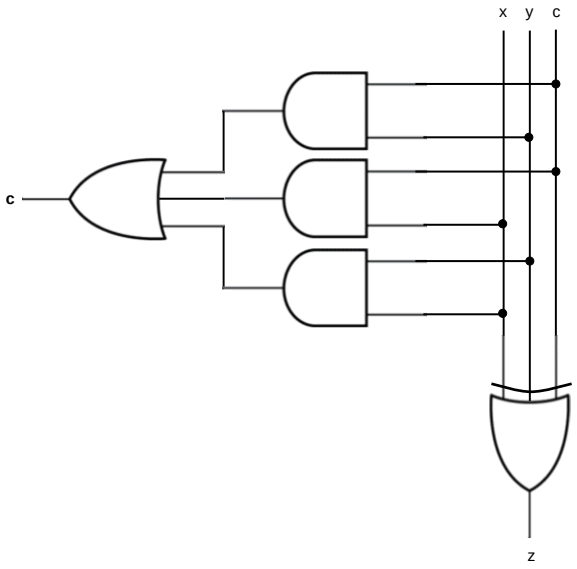
Let's assume that we are dealing with 8 bit numbers (each number is a byte). Then we can build this circuit by using 8 1-bit adders. Each adder takes two bits x and y and a carry and produces a bit result z, and a carry to the next adder (this is how we human beings perform addition, i.e. one column at a time). The carry received by the first adder is 0 (hard wired) and the carry produced by the last adder is ignored.



We only need to design one such adder. Let's look at the truth table of such adder:

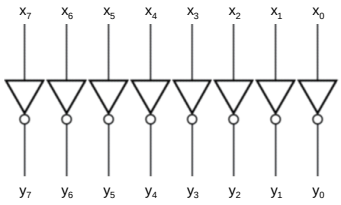
c	x	y	**c**	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

We can follow the technique of Problem 4 to build the curcuits for the above truth table (a circuit that produces **c** and a circuit that produces z). But we can immediately notice that z is 1 whenever the number of 1's in the input is odd. Therefore, z can be obtains by XORing c, x, and y. We can also notice that we have a carry whenever we have at least two ones in the inputs. This suggests the following circuit for the adder (recall that we can cascade gates for gates with multiple inputs).



Problem 6
Build a circuit to negate a number in 1's complement.

We simply need to flip all the bits (that's how negation works in 1's complement). If numbers are 8 bit numbers, we need 8 NOT gates for each bit. That's it. The following figure shows how to negate a given number $x_7 \dots x_0$.



Problem 7 Build a circuit to negate a number in 2's complement.

We simply need to flip all the bits and then add 1 (that's how negation works in 2's complement). Therefore, we can use the circuit of Problem 6 first to obtain the flipped pattern, then we can use the circuit of Problem 5 to add 1. That's it. The following figure shows how to add a 1 to a given 8 bit number $x_7 \dots x_0$.