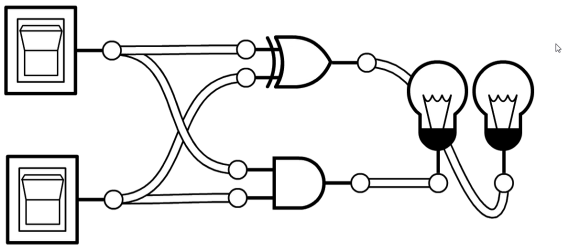
Now that we know how to add and multiply using binary numbers we can apply this knowledge to our previous understanding of circuits.

Our aim will to be have our inputs as the numbers that we will add or multiply on and our outputs as the product or sum.

Half adder

Let's start with the most basic example:



Half adder circuit

This circuit has the following possible range of outputs, where A and B are the input switches and X and Y are the output signal. The logic gates (an `XOR` and an `AND`) are equivalent to the add function.

```
A  B  X  Y
-  -  -  -

0  0  0  0
0  1  0  1
1  0  0  1
1  1  1  1
```

We can see that if we treat A and B as single binary digits that could correspond to 2^1 (either `0` or `1`) , then the X and Y outputs can be viewed collectively to constitute the sum of A and B (we have put the denary equivalent in brackets)

```
A  B  X  Y
-  -  -  -

0  0  0  0      0 + 0 = 00 [0]
0  1  0  1      0 + 1 = 01 [1]
```

```
1  0  0  1      1 + 0 = 01 [0]
1  1  1  1      1 + 1 = 10 [2]
```

This is called a half adder because it cannot go higher than 2^1.

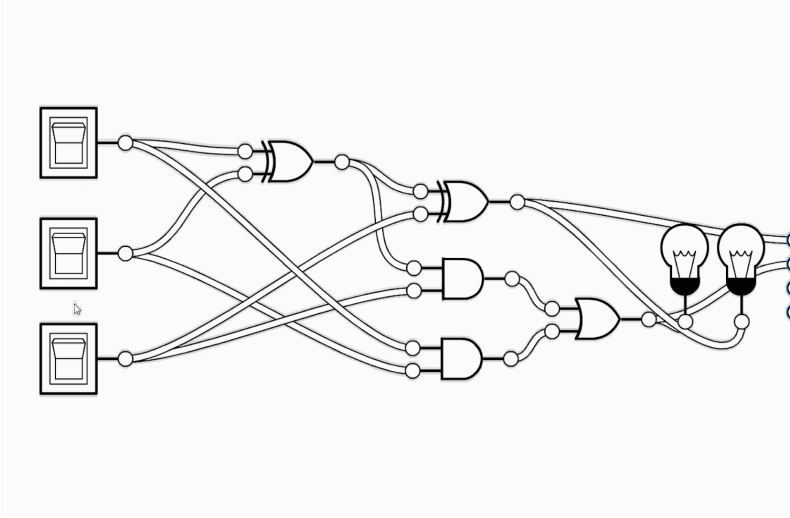
Representing binary output as denary values

There are special output components that can represent the combination of binary inputs and logic gates as denary values. Here is an example using a **seven-segment display** :

maths_with_logic_gates_5.gif.crdownload

Full adder

To represent numbers higher than the denary 2, we would need a carrying function so that we could represent numbers up to denary 3 and 4. The limit of a half adder is 2^1.



We do this by adding another switch input: