Title:
How Computers Add - A Logical Approach

Word Count:
847

Summary:
We looked at Number Systems and counting (see It's a Binary World - How Computers Count) last time. As a quick refresher, we saw that computers are made up of many units of 0 and 1, the binary system. 1 is the highest digit possible so numbers in the computer are stored as for example 1010 or 10 in decimal. We also saw that these binary numbers can be seen as octal (8) or hexadecimal (16) numbers - in this case 1010 becomes 15 octal, or A hex.

You probably realise that the...

Keywords:

Article Body:
We looked at Number Systems and counting (see It's a Binary World - How Computers Count) last time. As a quick refresher, we saw that computers are made up of many units of 0 and 1, the binary system. 1 is the highest digit possible so numbers in the computer are stored as for example 1010 or 10 in decimal. We also saw that these binary numbers can be seen as octal (8) or hexadecimal (16) numbers - in this case 1010 becomes 15 octal, or A hex.

You probably realise that the 'standard' PC code is in 8 bit bytes taking the hex system a stage further. You may also know that processors, and Windows software that runs on them, have progressed from 8 bits to 16 bits to 32 bits to 64 bits. Basically this means the computer can work on 1,2, 4 or 8 bytes at once. Don't worry if this is all Gobbledegook, you don't need it to understand how computers add!

OK now to the Math - cringe time! It's a little more complicated than last time, but if you think logically, like a computer, realising they are really dumb, you will sail through it!

We take a break here to look at a bit of math you may not have heard of - Boolean Algebra. Once again it's really simple, but it shows you how a computer works, and why it is so pedantic!

Boolean Algebra is named after George Boole, an English Mathematician in the 19th Century. He devised the logic system used in digital computers more than a century before there was a computer to use it!

In Boolean Algebra, instead of + and - etc. we use AND and OR to form our logic steps.
For example:-

x OR y = z means if x or y is present, we get z.
However,
x AND y = z means that both x and y need to be present to get z.
We can also consider an XOR (eXclusive OR).
x XOR y=z means that x or y BUT NOT BOTH must be present to get z.

That's it! That's all the math you need to understand how a computer counts. Told you it was simple!

How do we use this logic in the computer? We make up a little electronic circuit called a Gate with transistors and things, so we can work on our binary numbers stored in a register - just a bit of memory. (And that's the last electronics you'll hear about!). We make an AND gate, an OR gate, and an XOR gate

When we add in decimal, for example 9+3 we get 2 'units' and carry one to the 10s, giving 10+2=12

Remember the binary bit values in Decimal 1,2,4,8 etc? We start at 0 then 1 in the first bit position, the 1 bit. If we add 1 + 1 binary we have to end up with 10, which has a 1 bit in the second bit position, and a 0 in the first, giving Decimal 2+0=2. This second bit position is formed by a CARRY from the first bit.

To make an adder we must duplicate with a logic circuit the way we add in binary. To add 1+1 we need 3 inputs, one for each bit, and a carry in, and 2 outputs, one for the result (1 or 0), and a carry out, (1 or 0). In this case the carry input is not used. We use 2 XOR gates, 2 AND gates and an OR gate to make up the adder for 1 bit.

Now we go another step, and forget about gates, because now we have a Logic Block, an ADDER. Our computer is designed by using various combinations of logic blocks. As well as the adder we might have a multiplier (a series of adders) and other components.

Our ADDER block takes one bit (0 or 1) from each number to be added, plus the

Carry bit (0 or 1) and produces an output of 0 or 1, and a carry of 0 or 1. A table of the input A, B and Carry, and output O and Carry, looks like this:-

With no Carry in:

A	B	c	O	C
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1

With Carry in:

A	B	c	O	C
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

This is known as a Truth Table, it shows output state for any given input state.

Let's add 2+3 decimal. That is 010 plus 011 binary. We will need 3 ADDER blocks for decimal bit values of 1, 2 and 4)

The first ADDER takes the Least Significant Bit (decimal bit value 1) from each number. Input A will be 0, input B will be 1 with no carry - 0.

From the truth table this gives an output of 1 and a carry of 0 (3rd row). BIT 1 RESULT = 1

At the same time the next ADDER (decimal bit value 2) has inputs of 1, 1 and a carry of 0, giving an output of 0 with a carry bit of 1 (4th row). BIT 2 RESULT = 0

The next ADDER (decimal bit value 4) has inputs of 0, 0 and a carry of 1, giving an output of 1 with no carry - 0 (5th row). BIT 4 RESULT = 1.

So we have bits 4,2,1 as 101 or 4+1=5.

It seems like a laborious way to do it, but our computer can have 64 adders or more, adding simultaneously two large numbers billions of times a second. This is where the computer scores.

Next time we will get to how a computer performs more complcated operations, and

it's simple!