Q-Module: CT Segment; Reading 2a (Counting in Binary)

Excerpts from
How To Count (Programming For Mere Mortals, Vol 1)
Steven Frank

Binary

What if instead of 10 symbols, we only had 2? The good news is we'd still be able to represent any numeric value, just as we can in decimal.

It turns out there is indeed such a thing as base 2. It's called "binary", and it's very important to computers and programming.

The rules work exactly as they do in base 10, except our only two available symbols are:

0, 1

So, let's count in binary, starting again from the value zero:

0

1

Uh oh. We haven't even made it to the value two, and we've run out of symbols. What would we do in decimal? Think back to the odometer example [used earlier to explain how decimal counting works. You have a series of cylinders with 0-9 inscribed on them.] How would an odometer work, if each digit on the cylinders only went from 0 to 1 instead of 0 to 9? Well, when we needed to go past 1, we'd add 1 the digit to the left and wrap the rightmost digit around back to 0:

10

It looks like a ten, because we're used to decimal. But that's the decimal value two expressed in binary. Let's keep counting:

11

There's the decimal value three, but we've run out of symbols again. Same old odometer story: increase the digit to the left by 1, and wrap the 1s back around to 0:

100

101

110

111

Can you guess the next number?

1000

The binary number 1000 is the same value as the decimal number 8. It looks like "one thousand" to us, because we want to think in decimal, and it just so happens that both decimal and binary use two of the same symbols: 0 and 1. But you can plainly see that we've only counted 8 numbers, not a thousand.

Why is binary so important to computers? Because a single binary digit — or "bit" for short — is the smallest individual unit of data that a computer can deal with.

Bits are the "atoms" of everything digital. Every piece of information stored or processed by a computer is made up of some number of bits. Sometimes just a few, sometimes thousands or millions. But everything the computer understands is, at its lowest level, made up of bits. Text, graphics, sounds, program instructions — all made of bits in one way or another …

Why do computers want to speak in binary? Because bit values can be represented very easily using electricity within the computer's circuitry. The presence of electricity on a wire might indicate a "1". The absence of electricity, a "0". Just like a light switch, you can represent one of two states, on or off, by whether or not electricity is coming down a wire.

Now imagine you had a few million light switches, each attached to a light. How many possible combinations of lights off and on could you have? Quite a few. In fact, for each additional light switch you add, you double the number of possible combinations. In just a moment, I'll show you how that works.

You may have heard the term "8-bit" in reference to very early personal computers. To what does this refer? It means the processors in those vintage machines were able to most efficiently work with units of data 8 bits in size. Here's an arbitrary 8 bit binary number:

0001 1011

I've split it into two groups of four digits to make it easier to read, but you should consider it eight contiguous bits. How could we figure out what the decimal value of that binary number is?

You might not have noticed, but as we were doing our binary counting exercise, each time we had to add another digit to the left, we'd gone up by a power of two in decimal:

Binary	Decimal	Power of 2
1	1	2^0
10	2	2^1
100	4	2^2
1000	8	2^3
1 0000	16	2^4
10 0000	32	2^5
100 0000	64	2^6
1000 0000	128	2^7

(What does “power of two” mean? It’s a math term, meaning to multiply the number 2 by itself a certain number of times. In other words, 2^3 – read as “two to the power of three” or “two to the third power” – means “multiply two by itself three times”, or put another way: 2 × 2 × 2. There’s a special exception for 2^0: two to the power of 0 equals 1.)

If you look at the previous chart you can easily see that every time you add another bit to a binary number, you’ve doubled the range of possible numeric values that can be represented. With just 8 bits, you can represent 256 unique numeric values, starting with 0 and counting up to 255. In modern-day computers, a group of 8 bits is also called a “byte”.

Also notice in the chart that when a power of two is represented in binary, it is simply a “1” followed by that many “0”s. For example, the decimal number 8 is equal to 23, and indeed the binary representation is a “1” bit followed by three “0” bits.

If you’re visually inclined, like me, imagine our mystery 8 bit number as a series of 8 boxes, each box having a corresponding decimal value. Then, for each box that contains a “1” bit, add up the corresponding decimal numbers:

Decimal	128	64	32	16	8	4	2	1
Binary	0	0	0	1	1	0	1	1

Our binary number has “1”s at the decimal 16, 8, 2, and 1 positions. Add those together:

16 + 8 + 2 + 1

= 27

The 8-bit binary number 00011011 is equal to the decimal number 27.

---- END ----