Binary - Decimal - Hexadecimal

What are all these numbers anyway?

We commonly use terms like 'binary,' and 'bit,' and 'byte,' but maybe we haven't sufficiently explained what they are. So here's a new attempt.

Computers are basically just piles of switches, billions of them, driven by various clocks to make them switch back and forth in complex sequences. An electrical switch has two states - 'on' or 'off' - 'voltage' or 'no voltage' - ‘magnetized' or ‘not magnetized' - 'high' or 'low'. Logicians use the terms 'true' and ‘false' for the same distinction. Mathematicians and computer scientists call these '0' and '1'. These are known as bits and are generally represented in binary numbers. Digitizing is the process of converting information in the world into numeric form, and since computers, again, being piles of simple switches, represent this data using base-2 or binary numbers.

Imagine we use a lens to capture a landscape scene onto a flat surface, a camera obscura. If we divide that rectangle into a grid of tiny squares, and read the light levels from each square, we are converting the landscape information into numerical form, usually numbers from 0-255. Note, we’ve made tradeoffs: we have divided a continuous landscape into a series of squares, called pixels; we are reading total light levels thereby making a full color world into a greyscale; and we have taken a continuous range of possible light levels (analog information) and restricted it to a fixed range of numbers. If we now represent these in binary form, we can imagine encoding the same data as a series of switches.

Be sure to read through “binary numbers 1”. Since this is often confusing, here is a slightly different take on binary numbers.

Decimal
Think of our ordinary base-10 counting numbers. Our numbers go from 0 to 9 (10 different digits) at which point, when we add one, we find we have run out of digits. So we do this thing called moving over a place and creating a thing called a ‘ten’ (10). We continue, filling in numbers up to 19, at which point we create 20. These extra places, these shifts, are called powers of 10. That is, we take our digits and multiply them by the appropriate power of ten. So, 0-9 in the ones (rightmost) place are multiplied by no tens (10^0) so they are simply the digits themselves (called 'ones'). 0-9 in the tens (10^1) place are multiplied by 10 and added to whatever we had in the ones place. Likewise, for hundreds (10^2), thousands (10^3), and so on. Make sense?

Binary
We can do the same thing with binary numbers. However, now we only have two digits to work with - '0' and '1'. So let's count using the same idea as above, that is, when we run out of digits, we invent a new place. 0, 1, 10, 11, 100, 101, 110, 111. Congratulations! We've counted from 0 to 7 in binary. Here are the places:

2^0 = 1's place
2^1 = 2's place
2^3 = 4's place

Let's keep going a bit:

2^4 = 8's place
2^5 = 16's place
2^6 = 32's place
2^7 = 64's place
2^8 = 128's place

Each of these places, that is, each '0' or '1', is called a '**bit**'. And 8 of these bits is called a '**byte**'. No longer used but historically interesting, four bits is called a 'nibble'.

It's convenient to use leading zeros when writing binary numbers so that you don't confuse the places. So, we might write: 00000000, or 11111111, or 00110011 and so on. Computer programers often precede these with some designation to distinguish them from other kinds of numbers. After all, 11111111 in binary looks just like eleven million one hundred eleven thousand one hundred eleven in decimal. In C++ for micro-controllers, it's common to use '0b' (zero and small b) before the number, as in 0b11001100. Note that this isn't done for decimal numbers, which are just written as is.

How do we turn binary numbers into decimal numbers? Simple, if there is a '1' in a place, we add that power of 2 to the total otherwise we skip it. Example: what is 00100011 in decimal? We have 1's in the 1's, 2's, and 32's places, and 0's in the 4's, 8's, 16's, 64's, and 128's places. So we add 1+2+32=35.

How do we go the other way? Easy, use a calculator; selecting Programmer under the View Menu in the Apple Computer calculator allows you to switch between decimal and binary numbers Or do it the long way: take the original number, keep dividing by 2 and keep track of the remainders as we build the binary value from right to left. So,

35 / 2 = 17 R1 so we have a '1' in the 1's place. Keep going,
17 / 2 = 8 R1, so we put another '1' in the 2's place.
8 / 2 = 4 R0, so a '0' goes in the 3rd (or 4's place).
4 / 2 = 2 R0, so another '0' for the 8's place.
2 / 2 = 1 R0, so we have a '0' in the 16's place. And finally,
1 / 2 = 0 R1 so we have a '1' in the 32's place.
That's 100011, written by programmers as 0b00110011.

Hexadecimal
Four bits of binary are enough to count from 0 to 15 (0b1111 is 1 + 2 + 4 + 8 = 15). Programers wanted a quick way to specify these groups of 4 bits with a single number, but our decimal numbers only go up to 10. So they created a new system, base-16 (hexadecimal), and added 6 new digits, called 'a', 'b', 'c', 'd', 'e', 'f'. In other words, The ones place in hexadecimal consists of the digits: 0-9, a, b, c, d, e, and f where 'a' is commonly known as eleven, 'b' is twelve, and so on up to 'f' which is 15. After that, we create a new place, using the same digits, but this time, its a 16's place. For people who've gotten used to these numbers, it's a very quick way to express strings of binary numbers without so many zeros and ones to keep track of and mix up..

Going back to our example, 35, or 0b00100011. We split the binary number into two nibbles: 0010 and 0011. Now we figure out the hexadecimal equivalents of each of these: 0010 is 2, and 0011 is 3, so 35 decimal is the same as 23 hex. Looked at another way, 23 hex has two 16's and three 1's --> 32 + 3 = 35. (by the same token, 35 decimal has three 10's and five 1's --> 30 + 5 = 35) Programmers write hexadecimal numbers as: 0x23, to distinguish them from decimal or binary numbers. Those of you who make Web sites, should immediately recognize these numbers as the colors used in HTML.

ASCII
American Standard Code for Information interchange, or ascii, is used to map every possible typed character on a computer keyboard to numbers from 0 to 127. Note, this is 7 bits. More on this later.

http://www.asciitable.com
https://en.wikipedia.org/wiki/ASCII