Chapter 4: Storage

Computers Use Binary

Now you will finally understand the purpose of having to learn the basics of numbering systems. Like only knowing one language, computers only really know one numbering system: binary. This is because a computer's brain is simply made up of many switches. Being switches, they can only have one of two values. Those values are then thought of as either 'on' or 'off' which we equate to '0' and '1' in binary.

Actually the native machine language boils down to zeroes and ones as well. Everything in your computer is based on this numbering system. In fact, everything "digital" is made up of a sequence of *bits*. A bit is a single 'switch' or binary digit. In the binary number '0101' there are four *bits*. What makes up digital stuff is that it is very finite and specific. So as is the numbering system that it uses and the range of those numbers. Machines do not normally work with individual bits; they deal with larger storage units.

Storage Units

A storage unit is a single, set amount of one or more binary digits. Therefore a bit is a single binary digit and can only store one of two values. All storage units larger than bits are measured in bits, which simply equate to binary digits anyway. Larger storage units are also usually measured in their smaller counter-parts. So a *nibble* is four (4) bits, a *byte* is two (2) nibbles, a *word* is two (2) bytes, a *double-word* is two (2) words, and a *quad-word* is two (2) double-words:

Storage Unit	**Bits**
Nibble	4
Byte	8
Word	16
Double-Word	32
Quad-Word	64

You may think of the storage unit as just a number, but you have to remember it is a number whose range is very finite. These storage units that I speak of are *digital* in that they are located on a digital device such as your computer's hard drive. A digital device deals solely with finite data based on bits. Think of a storage unit as a cup that you only ever fill with water. It can store any amount of water that does not spill over the edge and it can also contain no water. Its range of water volume, therefore, is very finite and is not subject to change. However much water you put in that cup, or how you use it, is completely up to you. It is the same with this *digital data*.

Digital Data

Digital data refers to the values represented by the storage units kept on a digital device. This device is like a piece of paper when you write a phone number down. That phone number will be known for as long as the paper exists unless it is memorized (copied to your brain). Computers work in much the same way. The storage devices they use are either permanent or temporary (covered in Chapter 1). However, the values stored by a computer are very finite and very specific. The data stored is made up of one or more storage units which break down into individual bits.

A digital storage device is based on a specific storage unit. This unit is commonly the byte. Your computer's hard drive, for example, can store a specific number of bytes. The basic storage unit of a device is the smallest unit it will have any dealings with. So if you want to store a value that takes fewer bits than the smallest unit, you still have to use it. For example, if you want to store '1001b' on a hard drive that only deals with bytes you must store it as '0000 1001b'. This is the same thing number, but it uses more bits.

Hexadecimal Rules Storage

A storage unit's minimum value is always zero (duh), but its maximum value is determined by the number of bits that makes it up. Let us take a look at a nibble first. It is made up of four bits, so at its range is '0000b' to '1111b'. If we use our brains, and previously learned process of converting binary to decimal, we will find that the range is '0' to '15'. It's a little funky, but '15' in hexadecimal is simply '0xF'. Therefore a nibble represents a single hexadecimal digit just like a bit is a single binary digit.

The hexadecimal numbering system is extremely helpful when dealing with storage units. Two hex digits equate to a byte, four to a word, eight to a double word, and sixteen to a quad-word. This is a lot easier than dealing with the massive amounts of bits associated with each. An added plus is the ease at which you can convert binary to hex and vice versa. Most of the time when dealing with raw storage you will see and use hexadecimal rather than decimal or even binary. It lends itself much to that purpose.

Vast Amounts

Where vast amounts of data are concerned, specifically bits or bytes, the amount is referred to with a smaller value whose name is prefixed. These prefixes I am sure you have already heard of: kilo, mega, giga, and tera. Each of these represents a particular weight which, when multiplied by the value included, represents an approximate number of bits or bytes. The prefix values start with 'kilo' whose weight is one thousand. The prefix values increase from there. Each following prefix weight is equal to the last weight squared. So 'mega' is one thousand squared or one million or one thousand thousand. ☺

Now let's try these prefixes with data. A kilobyte is one thousand bytes. Ten kilobytes is ten thousand bytes. Thirteen megabits is thirteen million bits. Your hard drive is measured like this. If you have a twenty (20) gigabyte hard drive then it has twenty billion bytes available for storage. Just think; each one of those bytes can contain one of 256 different values. This means that, with your twenty gigabytes, you could have up to 1,310,720,000,000,000 different sets of data. That's a lot of possibilities!

Sometimes numbers will be trailed by an abbreviation rather than the whole word. The abbreviation is typically the first letter of the prefix and is sometimes followed by a 'b'. So instead of seeing '10 kilobytes', you'll probably see '10k' or '10kb'. The 'b', unfortunately, is extremely ambiguous and can mean *either* bytes or bits. The context of what it's used in is your only clue as to what it means.

Unfortunately there are two ways of seeing the prefix values and I have only just shown you one, which is used in the world of hard drives mostly. The other way is the same except the weight of each prefix is a multiple of 1024, not 1000. This number is based on binary restrictions rather than simple metrics. So a 'kilo' is actually 1024, 'mega' is 1024 kilo, 'giga' is 1024 mega, and 'tera' is 1024 giga. Therefore ten kilobytes (10k) in this system is ten thousand two-hundred forty (10,240) bytes. This second system is used every where else in computers that I have seen.

You won't have to deal much with the first because even operating systems will tell you hard drive information based on the 1024 variant. The only time you will need to know the first variant is when you actually want to buy a hard drive. A twenty gigabyte hard drive is twenty billion bytes not 21,474,836,480 ($20*1024^3$).

N-Bit CPU

But computers don't deal with bits *or* nibbles directly. The smallest storage unit directly utilized is a byte which is eight (8) bits[1]. And even then a CPU must *upscale* a single byte of storage to be used in any instructions. When you bought your computer, whatever it is, you may have noticed that the one of its features was the 'bits' of its processor. Most computers these days are running 32-bit CPU's, but we are all on the verge of making the leap to 64. What does this mean?

The count of bits associated with a CPU is actually the size of the storage unit used by the CPU for any of its calculations. The machine language of most CPU's includes three things to make up a single instruction: a command and two parameters. Each of these is actually a number whose storage unit is the same as the bit-count of the CPU. I am using the term 'bit-count' a bit (ha ha, pun pun) inappropriately, but I can't think of any other way of saying it without being confusing.

When the CPU is made to handle bigger numbers *natively*, it will be able to theoretically access that much more information faster. For example, you're computer's memory is

[1] Not all machines use eight (8) bits for a byte; but they are strange indeed and you encounter them rarely, if at all.

limited to about 4 gigabytes on a 32-bit processor. If you calculate out the maximum value of 32-bits you will see why.