**bitsandbytes**

**Bits and Bytes**

**A Data Communication Historical Series**

**By Bob Pollard**

**Bits and Bytes:**

**A very brief overview:**

The words bits and bytes are continually used when discussing a computer and both are an important part of design and operational functions. Both RAM and the hard disk capacities are measured in bytes, as are file sizes when they are stored or examined in a file viewer.

For instance: The statement "This computer has a 32-bit Pentium processor with 512 megabytes of RAM and 20 gigabytes of hard disk space” would be used to describe the capability and storage capacity of a particular computer.

**Decimal Numbers:**   
The easiest way to understand bits is to compare them to something familiar: Digits.

A digit is a single place in the common numbering system that can hold numerical values between 0 and 9. Digits are normally combined together to create larger numbers. For example, the number 6,453 are composed of four digits. It is understood that in the number 6,453, the 3 is filling the ‘1s position’, while the 5 is filling the 10s position, the 4 is filling the 100s position and the 6 is filling the 1,000s position. This would be referred to as a base 10 numbering system.

The number could be expressed in the following manner:

(6 x 1000) + (4 x 100) + (5 x 10) + (3 x 1) = 6000 + 400 + 50 + 3 = 6453

Numbering systems do not require the use of 10 different values in a digit. A number system can be any base number structure. In fact, there are many good reasons to use different number bases in different situations.

**Bits:**   
Computers operate using the base-2 number system, also known as the binary number system. The reason computers use the base-2 system is because it makes it a lot easier to design and implement them within the current electronic technology.

Computers use binary numbers, therefore they use binary digits in place of decimal digits. The word bit is a shortening of the words ‘Binary Digit’. Where decimal digits have 10 possible values ranging from 0 to 9, binary bits have only two possible values: 0 and 1. Therefore, a binary number is composed of only 0s and 1s, example the number ‘1011.

The first right digit position of 1011 has the power of one (0 or 1). The next position to the left has an increasing total power of three when added to the first digit. The next position to the left has the total power of seven when added to the previous two digits; and the next position to the left has a total power of fifteen (16 counting 0) when added to the previous three digits.

In the binary system, if a bit is 1, and a 1 is added to it, the first bit (right) becomes 0 and the next position (carry over) bit becomes 1. Counting from 15 to 16 the carry over through 4 bits causes 1111 to increase to 10000.

Starting at zero and counting through 20, decimal (left column) and binary (right column) would appear as shown in the table (listing) below.

**Decimal Binary**

**0 = 0**

**1 = 1**

**2 = 10**

**3 = 11**

**4 = 100**

**5 = 101**

**6 = 110**

**7 = 111**

**8 = 1000**

**9 = 1001**

**10 = 1010**

**11 = 1011**

**12 = 1100**

**13 = 1101**

**14 = 1110**

**15 = 1111**

**16 = 10000**

**17 = 10001**

**18 = 10010**

**19 = 10011**

**20 = 10100**

**Bytes:**   
Bits are usually bundled together into 8-bit groups, and these groups are called bytes. The question may arise: Why 8 bits in a byte? The 8-bit byte was found to be the most viable size for use in computer systems.

With 8 bits in a byte, a total value of 256 can be represented, which would provide a count of 0 to 255, as shown below:

0 = 00000000

1 = 00000001

2 = 00000010

3 = 00000011

**\***

254 = 11111110

255 = 11111111

A Compact Disk (CD) or other device may use 2 bytes, or 16 bits for addressing and storage purposes, which will give a bit count ranging from 0 to 65,535, illustrated below:

0 = 0000000000000000

1 = 0000000000000001

2 = 0000000000000010

3 = 0000000000000011

**\***

65534 = 1111111111111110

65535 = 1111111111111111

Computer systems typically use 32 or 64 bits for addressing purposes; i.e. 4 bytes or 8 bytes.

[**ASCII**](https://sites.google.com/site/mdprcp/ascii-1) (American Standard Code for Information Interchange):   
Bytes are frequently used to hold individual characters in a text document. In the ASCII character set (7-character bits); each binary value between 0 and 127 is given a specific character. Many computers extend the ASCII character set to use the full range of 256 characters available in a byte. This is possible when the 8th bit is not used for parity purposes. The extended character set uses the upper (additional) 128 characters to handle special characters such as accented characters from common foreign languages.

As the previous illustrations indicate in binary numbers when a bit is added to the existing bits the total count or possible value of the bits doubles. Example:

Value or total (including 0) = 2 4 8 16 32 64 128 256

Number of binary bits (including 0) = 1 2 3 4 5 6 7 8

When many bytes are referenced the prefixes, such as: kilo, mega, giga are used. These prefixes are used in the following manner: kilobyte, megabyte and gigabyte or K, M and G, as in Kbytes, Mbytes and G-bytes, or KB, MB and GB may be used.

**Example of values:**

**Name Abbr. Size (bits)**

**Kilo K 1,024**

**Mega M 1,048,576**

**Giga G 1,073,741,824**

**Tera T 1,099,511,627,776**

**Peta P 1,125,899,906,842,624**

As illustrated above kilo is about a thousand, mega is about a million, giga is about a billion, and so on. So when someone says, "This computer has a 2 gig hard drive”; implies the hard drive stores 2 gigabytes, or approximately 2 billion bytes, or exactly 2,147,483,648 bytes. A 2-gigabyte hard drive is not very large when one considers that a single CD will hold 650 megabytes, therefore 3 CDs would probably fill (use) the entire hard drive. Also Terabyte databases are fairly common these days, which would require a large capacity hard drive.

Notice the count (total value) does not come out exact, i.e. Kilo does not equal exactly 1000. This happens because the total values are calculated using binary bits (base 2).

**Binary Math:**

Binary math works just like decimal math, except that the value of each bit can be only 0 or 1. For example: add the following binary numbers.

**010**

**+ 111**

**1001**

Starting with the digits to the right, add 0 + 1 = 1 for the first total digit. No carrying required.

Add the middle digits and the result will be, 1 + 1 = 10 for the second total digit, so save the 0 and carry the 1. For the third digit (left), 0 + 1 + 1 = 10, so save the zero and carry the 1.The last 1 is carried forward and is added to the total (0 + 0 + 1 = 1) in the forth position giving a total of 1001. If this is translated over to a decimal number the result should be as follows: 2 (010) + 7 (111) = 9 (1001).