

Binary encoding

Introduction to binary

In the late 1930s, Claude Shannon showed that by using switches which were closed for "true" and open for "false," it was possible to carry out logical operations by assigning the number 1 to "true" and 0 for "false."

This information encoding system is called **binary**. It's the form of encoding that allows computers to run. Binary uses two states (represented by the digits 0 and 1) to encode information.

Since 2000 BCE, humans have counted using 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). This is called "decimal base" (or base 10). However, older civilizations, and even some current applications, used and still use other number bases:

- Sexagesimal (60), used by the Sumerians. This base is also used in our current system of timekeeping, for minutes and seconds;
- Vicesimal (20), used by the Mayans;
- Duodecimal (12), used in the monetary systems of the United Kingdom and Ireland until 1971: A "pound" was worth twenty "shillings," and a "shilling" was worth twelve "pence". The current system of timekeeping is also based around twelve hours (particularly in American usage);
- Quinary (5), used by the Mayans;
- Binary (2), used by all digital technology.

The bit

The term **bit** (shortened to a lowercase *b*) means "**binary digit**", meaning 0 or 1 in binary numbering. It is the smallest unit of information which can be manipulated by a digital machine. It is possible to represent this binary information:

- with an electrical or magnetic signal, which, beyond a certain threshold, stands for 1;
- By the roughness of bumps in a surface;
- using flip-flops, electrical components which have two stable states (one standing for 1, the other 0).

Therefore, a bit can be set to one of two states: either 1 or 0. With two bits, you can have four different states (2^2):

| | |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

With 3 bits, you can have eight different states ($2 \times 2 \times 2$):

| 3-bit binary value | Decimal value |
|--------------------|---------------|
| 000 | 0 |
| 001 | 1 |
| 010 | 2 |
| 011 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | 7 |

For a group of n bits, it is possible to represent 2^n values

Bit values

In a binary number, a bit's **value**, depends on its position, starting from the right. Like tens, hundreds, and thousands in a decimal number, a bit's value grows by a power of two as it goes from right to left, as shown in the following chart:

| | | | | | | | | |
|----------------------|-------------|------------|------------|------------|-----------|-----------|-----------|-----------|
| Binary number | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| value | $2^7 = 128$ | $2^6 = 64$ | $2^5 = 32$ | $2^4 = 16$ | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |

Conversion

To convert a binary string into a decimal number, multiply the value of each bit by its weight, then add together the products. Therefore, the binary string 0101, in decimal, comes to:

$$\begin{aligned} &2^3 \times 0 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 \\ &= 8 \times 0 + 4 \times 1 + 2 \times 0 + 1 \times 1 \\ &= 5 \end{aligned}$$

The byte

The **byte** (shortened to uppercase *B*) is a unit of information composed of 8 bits. It can be used to store, among other things, a character, such as a letter or number.

Grouping numbers in clusters of 8 makes them easier to read, much as grouping numbers in threes helps to make thousands clearer when working in base-10. For example, the number "1,256,245" is easier to read than "1256245".

A 16-bit unit of information is usually called a **word**.

A 32-bit unit of information is called a **double word** (sometimes called a *dword*).

For a byte, the smallest number possible is 0 (represented by eight zeroes, 00000000), and the largest is 255 (represented by eight ones, 11111111), making for 256 different possible values.

align="center" align="center" align="center"


| $2^7 = 128$ | $2^6 = 64$ | $2^5 = 32$ | $2^4 = 16$ | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |
|-------------|------------|------------|------------|-----------|-----------|-----------|-----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Kilobytes and Megabytes

For a long time, computer science was unusual in that it used different values for its units than metric system (also called the International System). Computer users would often learn that 1 kilobyte was made up of 1024 bytes. For this reason, in December 1998, the International Electrotechnical Commission weighed in on the issue (<http://physics.nist.gov/cuu/Units/binary.html>). Here are the IEC's standardized units:

- One kilobyte (kB) = 1000 bytes
- One megabyte (MB) = 1000 kB = 1,000,000 bytes

- One gigabyte (GB) = 1000 MB = 1,000,000,000 bytes
- One terabyte (TB) = 1000 GB = 1,000,000,000,000 bytes

| | |
|---|--|
|  | <p>Warning! Some software (and even some operating systems) still use the pre-1998 notation, which is as follows:</p> <p>One kilobyte (kB) = 2^{10} bytes = 1024 bytes</p> <p>One megabyte (MB) = 2^{20} bytes = 1024 kB = 1,048,576 bytes</p> <p>One gigabyte (GB) = 2^{30} bytes = 1024 MB = 1,073,741,824 bytes</p> <p>One terabyte (TB) = 2^{40} bytes = 1024 GB = 1,099,511,627,776 bytes</p> |
|---|--|


The IEC has also defined binary kilo (kibi), binary mega (mebi), binary giga (gibi), and binary tera (tebi).

They are defined as:

- One kibibyte (kiB) is worth $2^{10} = 1024$ bytes
- One mebibyte (MiB) is worth $2^{20} = 1,048,576$ bytes
- One gibibyte (GiB) is worth $2^{30} = 1,073,741,824$ bytes
- One tebibyte (TiB) is worth $2^{40} = 1,099,511,627,776$ bytes

In some languages, such as French and Finnish, the word for byte does not begin with "b", but the international community, on the whole, generally prefers the English term "byte." This gives the following notations for kilobyte, megabyte, gigabyte, and terabyte:

kB, MB, GB, TB

| | |
|---|--|
|  | <p>Note the use of an uppercase <i>B</i> to distinguish <i>Byte</i> from <i>bit</i>.</p> |
|---|--|

This is a screenshot from the software *HTTrack*, the most popular offline web browser, showing how this notation is used:

| Informations | | | |
|-----------------|--------------|----------------------|----------------|
| Octets écrits: | 168,56MiB | Liens parcourus: | 420/675 (+100) |
| Temps: | 20s | Fichiers écrits: | 533 |
| Taux transfert: | 0B/s (85B/s) | Fichiers mis à jour: | 1 |
| Connexions: | 1 | Erreurs: | 96 |

Binary operations

Simple arithmetic operations such as addition, subtraction, and multiplication are easily performed in binary.

Addition in binary

Addition in binary follows the same rules as in decimal: Start by adding the lowest-valued bits (those on the right) and carry the value over to the next place when the sum of two bits in the same position is greater than the largest value of the unit (in binary: 1). This value is then carried over to the bit in the next position.

For example:

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 1 | 0 | 1 |
| + | 0 | 1 | 1 | 1 | 0 |
| - | - | - | - | - | - |
| | 1 | 1 | 0 | 1 | 1 |

Multiplication in binary

The multiplication table in binary is simple:

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

Multiplication is performed by calculating a partial product for each multiplier (only the non-zero bits will give a non-zero result) When the bit of the multiplier is zero, the

partial product is null; when it is equal to one, the partial product is formed from the multiplicand, shifted X places, where X is equal to the weight of the multiplier bit.

For example:

| | | | | | | |
|---|---|---|---|---|---|--------------|
| | | 0 | 1 | 0 | 1 | multiplicand |
| x | | 0 | 0 | 1 | 0 | multiplier |
| - | - | - | - | - | - | |
| | | 0 | 0 | 0 | 0 | |
| | 0 | 1 | 0 | 1 | | |
| 0 | 0 | 0 | 0 | | | |
| - | - | - | - | - | - | |
| | 0 | 1 | 0 | 1 | 0 | |

Source: <http://en.kioskea.net/contents/57-binary-encoding>