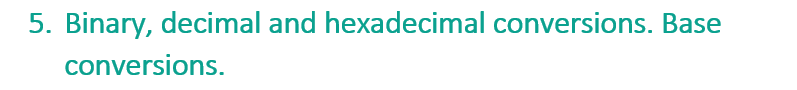


Subscribe to DeepL Pro to edit this document.  
Visit [www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document) for more information.



**Any base to decimal**

The formula applied to convert a number in any base to the decimal system is to **multiply the value of each digit by the base raised to the position**, with the digit before the decimal point (corresponding to the units) occupying the position 0.

|  |
| --- |
| 100011 = 1 - 25 + 0 - 24 + 0 - 23 + 0 - 22 + 1 - 21 + 1 - 20  A816 = A - 161 + 8 - 160  2678 = 2 - 82 + 6 - 81 + 7 - 80 |

When they are **decimal numbers**, the procedure is the same, taking into account that from the decimal point to the right the exponents are negative.

|  |
| --- |
| 10,101 = 1 - 21 + 0 - 20 + 1 - 2-1 + 0 - 2-2 + 1-2-3 |

Remember that a number raised to a negative power is equal to the inverse of the number raised to the positive power:

2-n = 1 / 2n

An alternative when converting **from binary to decimal** is to write the weights of each digit on top of it and add only those whose digit has a value of 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **64** | 32 | 16 | 8 | **4** | **2** | **1** |  |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |  |

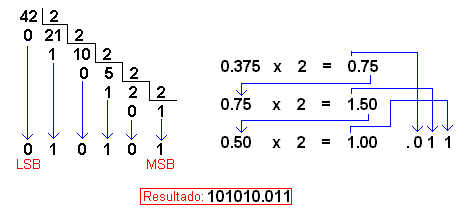
The above number would be 64 + 4 + 2 + 1 = 71

**Decimal base to binary**

To convert a number in base 10 to binary we have to make successive divisions by 2 until the result is 1. **The number in binary will be the last result plus all the remainders we have obtained**. Remember that you have to start with the last result, that is, from right to left.

If the number has a decimal point, the integer part is calculated according to the method we have seen above. The decimal part is calculated by successive multiplications by 2: multiply the decimal part by 2, keep the integer part of the result (which has to be 0 or 1) and multiply the decimal part by 2 again.

The result will be formed by each of the digits of the integer part that we have been obtaining.



**Hexadecimal to binary base**

The conversion between base 16 and binary is very simple because each hexadecimal digit can have exactly 16 values, which are precisely the values that we can represent with 4 binary digits (from 0000 to 1111).

Therefore, to convert a base 16 number to binary we only have to **replace each hexadecimal digit by its corresponding sequence of binary digits** (always groups of 4).

|  |
| --- |
| 5A9 = 0101 1010 1001 |

**Octal base to binary**

It is analogous to the previous step. In this case each octal digit can have 8 different values, which are the values that can be represented by 3 binary digits (000 to 111).

Therefore, to convert a base 8 number to binary, **each octal digit must be replaced by the corresponding sequence of 3 binary digits**.

|  |
| --- |
| 4370 = 100 011 111 000 |

Logotipo

Descripción generada automáticamente con confianza baja

**OPERATIONS IN BINARY**

In the binary system, the following operations can be performed:

* Mathematical operations: addition, subtraction, multiplication, division, etc.
* Logical operations, which use logical operators such as NOT, AND, OR, XOR, etc.

**Arithmetic operations**

* **Sum**

Tabla

Descripción generada automáticamente

* **Subtract**

Tabla

Descripción generada automáticamente con confianza media

**Logical operations**

They are based on Boolean algebra, named after the English mathematician George Boole, who laid the foundations of binary logic operations. The internal circuits of the computer contain logic gates that perform all the operations, but actually perform the logic operations.

* **NOT (negation)**

Texto

Descripción generada automáticamente con confianza baja

* **OR (or)**

Tabla

Descripción generada automáticamente con confianza media

* **AND (and)**

Tabla

Descripción generada automáticamente con confianza media

* **XOR**

Tabla

Descripción generada automáticamente

* **NAND**

Tabla

Descripción generada automáticamente

* **NOR**

Tabla

Descripción generada automáticamente

**COMPLEMENTS**

Internally, the computer uses complements both to represent negative numbers and to subtract, because it is easier to perform the physical circuit of complement and addition than that of subtraction.

To subtract another number from a number, you would add the first number to the complement of the number you want to subtract, and the result obtained will be identical to the subtraction.

Let's take a look at the different types of complements.

**Complement to 1**

The 1's complement of a binary number is obtained by changing each 0 by 1 and vice versa. In other words, each bit of the number is changed by its complement.

Diagrama

Descripción generada automáticamente con confianza baja

**Complement to 2**

It arises to solve the problem that when representing negative numbers with the complement a1, there are two possibilities to represent the number zero.

The **2's complement** of a binary number can be obtained using several methods:

***FIRST METHOD***

We add 1 to the 1's complement of the number we are looking for.

For example:

C2(1100)=C1(1100)+1=0011+0001=0100C2(1100)=C1(1100)+1=0011+0001=0100

***SECOND METHOD***

We go through all the zeros of the given number from the LSB (Least Significant Bit) to the left, until we find the first one, leaving the numbers as they are. From there, we invert zeros and ones.

*Let's look at the example above.*

*1100:* we go through the number from the LSB until we reach the first 1. From there, we invert the remaining 1, and we are left with, as before, 0100.