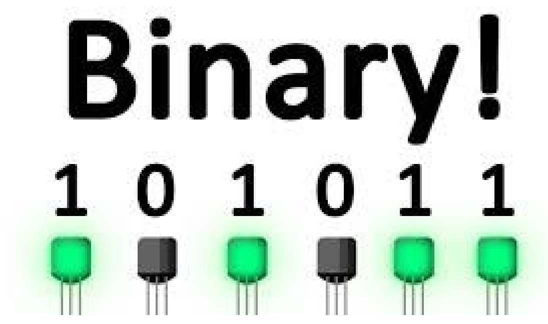
Chapter II :

Number systems

Information coding



Introduction :

Number systems and digital coding form the basis of representing information in the digital world. Whether for scientific calculations, computer applications, or the operation of electronic devices, mastering these concepts is essential to understanding how data is processed, stored, and transmitted.

A **numbering system** is a method of representing numbers using a defined set of symbols. Some of the most common include **decimal** , **binary** , **octal ,** and **hexadecimal** , each of which has significance in different contexts. For example, binary is the basic language of computers, while hexadecimal makes it easier to read binary data.

Digital coding , on the other hand, involves converting information, such as letters or symbols, into machine-readable digital formats. Codes such as ASCII **, Unicode , or BCD** codes are used to represent text, special characters, or instructions.

Understanding these systems not only provides a better understanding of the fundamentals of computer science, but also practical tools for manipulating and interpreting digital data in various applications. This introduction leads us to explore the principles, conversions between systems, and rules that govern these essential representations of the digital world.

- Vocabulary

Whatever the nature of the information processed by a computer (image, sound, text, video), it is always in the form of a set of numbers written in base 2, for example 01001011. The term bit (lowercase b in notation) means " binary digit", i.e. 0 or 1 in binary numbering. It is the smallest unit of information that can be manipulated by a digital machine. It is possible to physically represent this binary information by an electrical or magnetic signal, which, beyond a certain threshold, corresponds to the value 1. The *byte* or capital B in notation) is a unit of information composed of 8 bits. It is used, for example, to store a character such as a letter or a number. A unit of information composed of 16 bits is generally called a word (in English *word*). A unit of information of 32 bits in length is called a double word (in English *double word* , *dword*). Many computer scientists were taught that 1 kilobyte was worth 1024 bytes, but in December 1998, the international organization *IEC* ruled on the question1. Here are the standardized units:

✓ **One kilobyte (KB) = 10^3 bytes**
✓ **One megabyte (MB) = 10^6 bytes**
✓ **One gigabyte (GB) = 10^9 bytes**
✓ **One terabyte (TB) = 10^{12} bytes**
✓ **One petabyte (PB) = 10^{15} bytes**
✓ **One exabyte (Eo) = 10^{18} bytes**
✓ **One zettabyte (Zo) = 10^{21} bytes**
✓ **One yottabyte (Yo) = 10^{24} bytes**
✓ **One ronnabyte2 (Ro) = 10^{27} bytes**
✓ **One quettabyte (Qo) = 10^{30} bytes**

II.1 Binary system :

This is the system that is used by the machine (the computer), it is composed only by binary sequences **(1 and 0)** . The base of this system is 2.

- Definition:

The **binary system** is a numbering system that uses only two digits: **0** and **1.** Unlike the decimal system that we commonly use and which is based on **10 digits** (from 0 to 9), the binary system is based on the base **2** .

- **The Decimal System and Human Thought**

Humans use the decimal system to count because it is more intuitive and based on the fact that we have **ten fingers** . Each digit in a decimal number represents a power of 10. For example, the number 345 is broken down into:

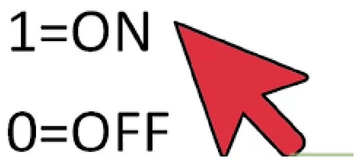
$3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 = 300 + 40 + 5.$

We are used to manipulating numbers this way, and it is very efficient for everyday human needs. However, machines **,** like computers, do not work in the same way.

- **Machines and Binary: The Language of 0s and 1s**

Machines, especially computers, work with **electrical circuits** . These circuits can only have two possible states:

- **On** (the current passes, represented by the number 1),
- **Off** (current does not pass, represented by the number 0).



This fits perfectly with the **binary system** , which is based on two states, 0 and 1. These two numbers are used to **encode all the information** that the computer processes, whether it is text, images, or programs.

- **How do computers understand numbers?**

Computers convert decimal numbers to binary numbers. For example, the decimal number 5 is written as 101 in binary:

$5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

So, each decimal number is broken down into powers of 2, and represented in binary. This simplified language (0 and 1) allows computers to perform very fast and precise calculations.

- **Conversion example**

Let's take the example of the decimal number 13:

- In binary, it is written **1101** , because:

$13 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

Each binary digit corresponds to an electrical state (**1 for "on" and 0 for "off"),** which allows electronic circuits to process and store this information.

- Example
  1001101: the decimal value of this sequence is:

$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 77$

Each binary digit (0 or 1) is called a bit, denoted by b, and any binary combination composed of eight bits is called a Byte .

II.2 The Different Coding Systems

In the world of computing, several numbering systems are used to represent and process digital data. The most common are the **binary system** , the **octal system , the hexadecimal** system , but there are others as well. Each system has a specific basis and is used for different applications, especially in programming, computing and electronics.

(a) Binary System (Base 2)

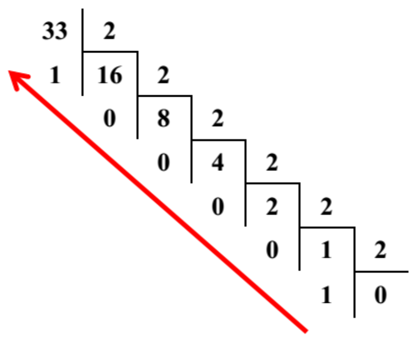
The binary system uses only two digits: 0 and 1. It is mainly used in computing, because computers work with circuits that can be in the "on" (1) or "off" (0) state.

- **Example** :
  The decimal number 10 is represented in binary as 1010.

- **Method :**

To convert a decimal number to a binary number, continuously divide the decimal number by the value 2, retaining the sequence of remainders from each division starting from the bottom to the top to obtain the equivalent number.

- Example: number 33 in binary



$$(33)_{10} = (100001)_2$$

(b) Octal System (Base 8)

The octal system uses eight digits (0 to 7). It is often used in electronics because it allows binary numbers to be represented by grouping three bits.

- **Example** : The decimal number 10 is represented in octal as 12.
- **Conversion** : To convert a decimal number to octal, divide the number by 8 and note the remainder.

(c) Hexadecimal System (Base 16)

The hexadecimal system uses sixteen digits: 0 to 9 for numbers 0 to 9 and A to F for numbers 10 to 15. It is widely used in programming, particularly to represent memory addresses or colors.

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
									10	11	12	13	14	15

- **Example** : The decimal number 255 is represented in hexadecimal as FF.

$$(\mathbf{FF})_{16} = 15 \times 16^{0} + 15 \times 16^{1}$$

Conversion : Converting a decimal number to hexadecimal is done by dividing the number by 16 and noting the remainder.

- **Other Coding Systems**

In addition to the systems mentioned above, there are other numbering systems used in specific contexts, such as the base **60 system** used in time (hours, minutes, seconds).

III.3 Conversion of a Number with Comma

To convert a decimal number with a comma, you must treat the integer part and the decimal part separately:

- For the whole part: we apply the traditional division method.
- For the decimal part: we multiply by the base and we keep the successive whole parts: this operation stops when we obtain a zero after the decimal point, otherwise it stops in the sixth multiplication'

Example :

Convert the number (23.625) from base 10 to base 2.

The integer part 23 is already determined, its result is: $(23)_{10} = (10111)_2$.

For the fractional part 0.625, we proceed as follows:

$$0.625 * 2 = 1.250$$

$$0.250 * 2 = 0.500$$

$$0.500 * 2 = 1.00$$

We see that the last fractional part is zero, so the result will be:

$\mathbf{(0.625)_{10} = (0.101)_2}$

The final result: $(23.625)_{10} = (10111.101)_2$

Let's take the infinite case for example $(23.56)_{10}$ For (23)10 it is always (10111): but for the fractional part $(0.56)_{10}$, we will have:

$$0.56 \times 2 = 1.12$$

$$0.12 \times 2 = 0.24$$

$$0.24 \times 2 = 0.48$$

$$0.48 \times 2 = 0.96$$

$$0.96 \times 2 = 1.92$$

$$0.92 \times 2 = 1.84$$

etc .,

We note that if we continue the operation of multiplying the fractional part by the value 2, we never reach the stopping criterion for this operation, this means that the binary sequence is infinite and we can truncate it by limiting ourselves to a certain number of bits (***6 bits after the decimal point***).

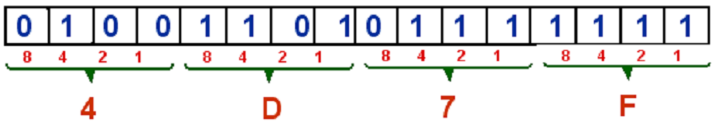
Hence the final result of: $(23.56)_{10}$ is $(10111.100011)_2$.

III.4 Operations in the binary system:

(a) Correspondence between binary and hexadecimal :

Converting from binary to hexadecimal is very simple, which is why we use this base. Simply match a four-bit word (nibble) to each hexadecimal digit.

Converting a 16-bit word between binary and hexadecimal



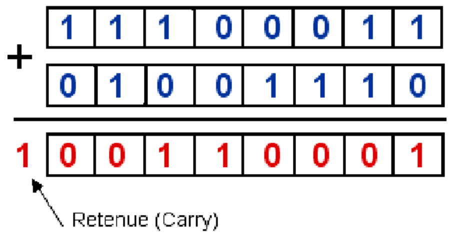
(b) Arithmetic and logical operations

- Addition in binary : The addition is performed bit by bit.

1 + 0 = 1

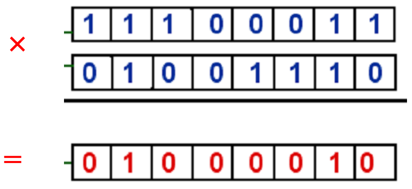
1 + 1 = 10

1 + 1 + 1 = 11



- **Logical product in binary**

The AND function is applied bit by bit



- **Converting Negative Numbers**

Negative numbers are often represented in computing using **two's complement** in binary systems. This involves reversing the bits (one's complement) and then adding 1 to the result.

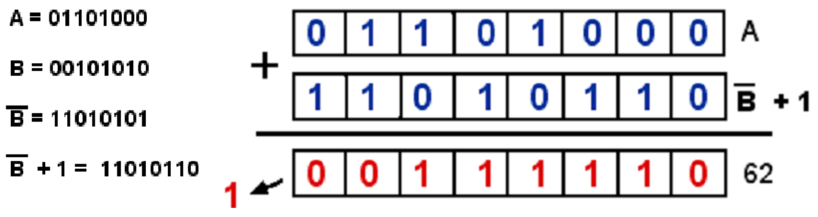
- **Example : 1**

$(-25)_{10} = (-B)_2 = \overline{B} + 1$

B		0	0	0	1	1	0	0	1
$\overline{B}$		1	1	1	0	0	1	1	0
$\overline{B}+1$		1	1	1	0	0	1	1	1

Example 2

In binary, the negative of a number is its 2's complement, that is, its complement + 1. Let two numbers A = 104 and B = 42. A - B = A + (- B)



A = 01101000
B = 00101010
$\overline{B}$ = 11010101
$\overline{B}$ + 1 = 11010110
0 1 1 0 1 0 0 0 A
+
1 1 0 1 0 1 1 0 $\overline{B}$ + 1
1
0 0 1 1 1 1 1 0 62

- The format is 8 bits, the left overflow bit should not be ignored.
- The first bit is 0 for negative numbers and 1 for positive numbers.
- The largest 8-bit signed number is +127 (01111111)
- The smallest 8-bit signed number is -128 (10000000) -128 to +127 => 256 combinations (2 to the power of 8)

II.5 Definition: Bits and Bytes

1. **Bit (b)** :
   - A bit is the smallest unit of data in computing, representing a binary state: 0 or 1 (off or on).
   - One bit alone cannot store much information, but when combined with others, it forms larger data units.
2. **Byte (B)** :
   - A byte consists of **8 bits** . It is the basic unit of information used to represent a character (like a letter, number, or symbol) in most computer systems.
   - For example, the letter 'A' is represented as **01000001** in binary, which is a byte (8 bits).

Conversion Table: Bits, Bytes, and Higher Units

No.	Unit	Abbreviation	Equivalent in Bytes	Notes
1	Bit	b	1/8 of a byte	Smallest unit of data
2	Byte	B	8 bits	Basic unit of digital information
3	Kilobyte	KB	1,024 bytes	1,024 bytes = 2^{10} bytes (Binary conversion)
4	Megabyte	MB	1,024 KB = 1,048,576 bytes	1,024 KB = 2^{20} bytes
5	Gigabyte	GB	1,024 MB = 1,073,741,824 bytes	1,024 MB = 2^{30} bytes
6	Terabyte	TB	1,024 GB = 1,099,511,627,776 bytes	1.024 GB = 2^{40} bytes
7	Petabyte	PB	1,024 TB = 1,125,899,906,842,624 bytes	1,024 TB = 2^{50} bytes
8	Exabyte	EB	1.024 PB = 1,152,921,504,606,846,976 bytes	1.024 PB = 2^{60} bytes
9	Zettabyte	ZB	1.024 EB = 1,180,591,620,717,411,303,424 bytes	1.024 EB = $^{2.70}$ bytes
10	Yottabyte	YB	1.024 ZB = 1.208,925,819,614,629,174,706,176 bytes	Largest unit in common use (2^{80} bytes)