UNIT 1
INTRODUCTION TO COMPUTERS

1. INTRODUCTION

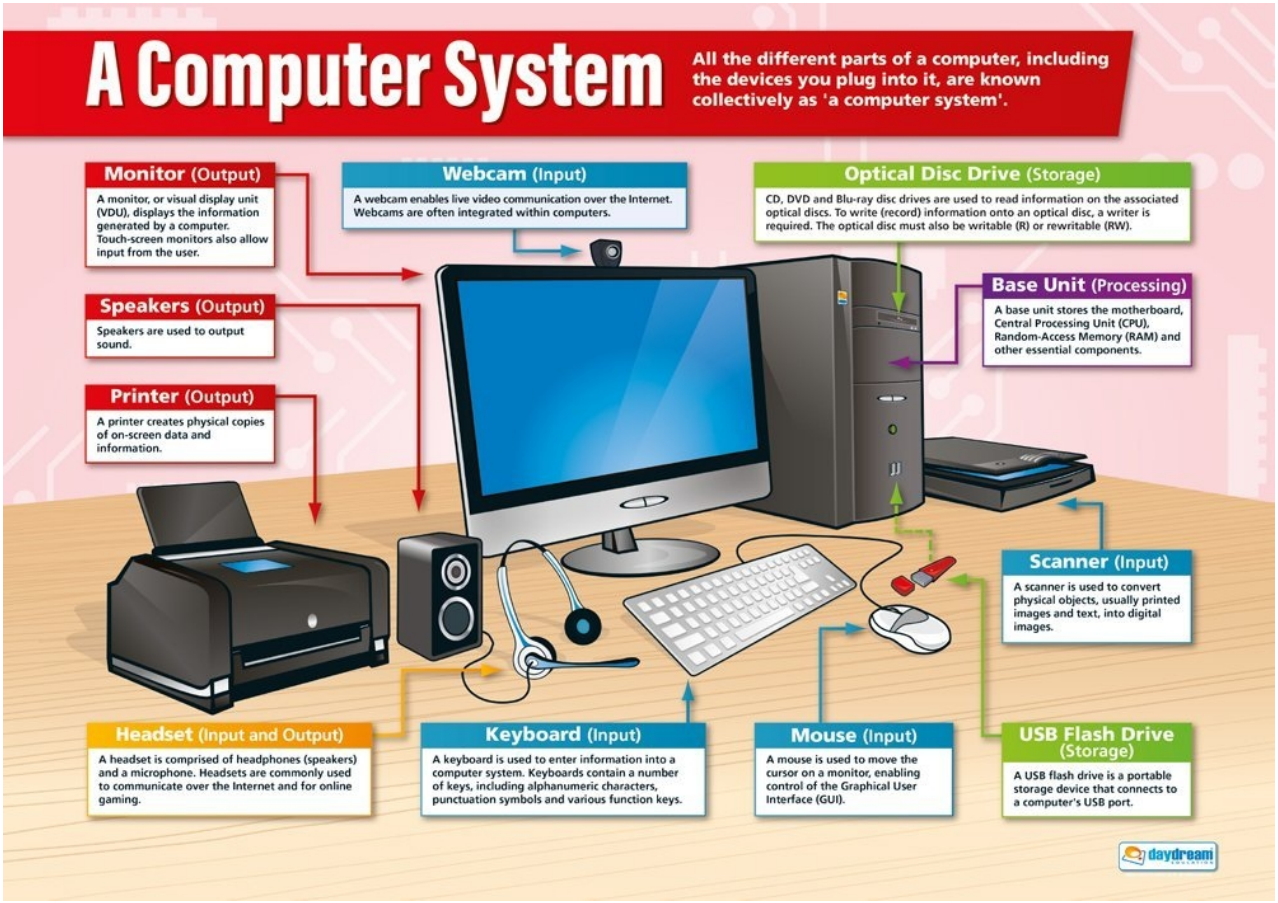
1.1. Basic concepts

Computer science or computing

It is the science that deals with the theory and methods of processing information in digital computers, the design of computer hardware and software, and the applications of computers.

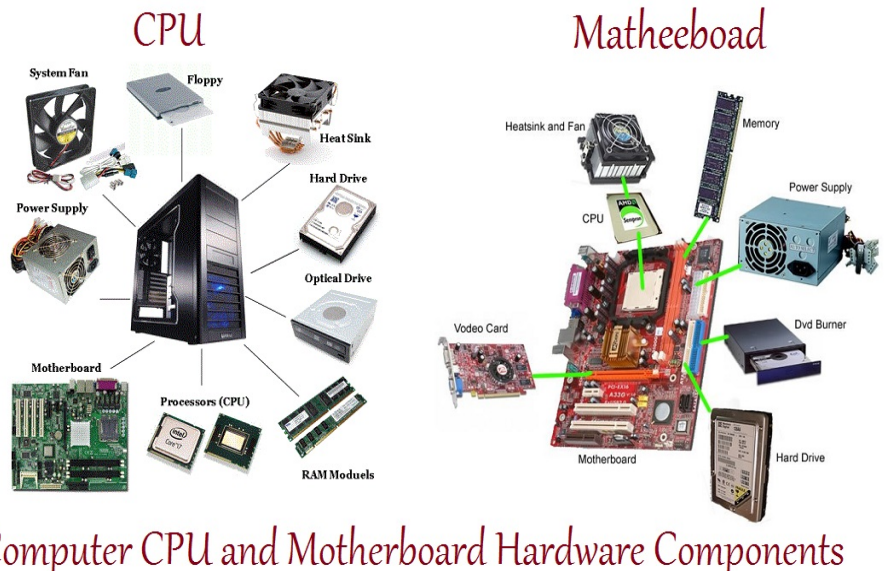
Computer system

A computer system is the set of **hardware** and **software** elements that are part of the processing and management of the information: computers, peripherals, wires, networking elements, operating system, applications, drivers, etc. It also includes the **users** that work with them and its **documentation** (user's manuals).



Hardware

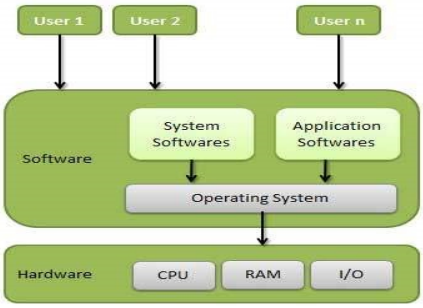
They are the physical parts of the computer system (case, keyboard, monitor, …).

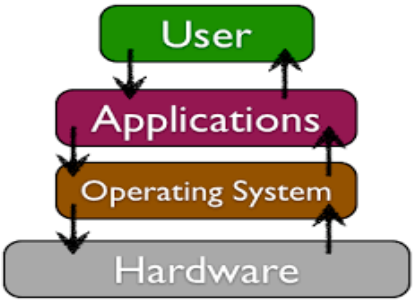


Computer CPU and Motherboard Hardware Components

Software

They are the logical components of the computer system. The software is usually divided into **System software** (operating system, drivers, …) and **Applications** (games, browsers, text documents, spreadsheets, databases, ...).





Operating system

It is the set of programs that manage the resources of the computer (CPU, memory, peripherals, …) and act as interface between the user or the applications and the hardware. It is an abstraction layer than let us work with a computer knowing nothing about its hardware characteristics.

Computer

A computer is an electronic device that accepts information using an input device. The information is processed by a central processing unit or stored in a storage unit and then processed. The result is supplied by an output device.

A computer is a device that can be instructed to carry out sequences of arithmetic or logical operations automatically via computer programming. Modern computers have the ability to follow generalized sets of operations, called programs. These programs enable computers to perform an extremely wide range of tasks.

Program

A program is a sequential set of instructions written in a programming language. These programming languages can be low level ones (machine language ans assembler) or high level ones (C, C++, Java, PHP, …).

1.2. Functions of a computer

- **Input**. The computer system receives information from the users. A user enters information using input devices like keyboard, mouse, webcam, punch card, magnetic tape, joystick, magnetic disk, etc. The Input unit accepts data, converts it to a readable form and sends it to Central Processing Unit (CPU) or a storage unit.
- **Storage**. The given data may be stored in a computer using different storage devices i.e. primary or secondary memory. The primary memory consists of Registers, Main Memory and Cache Memory, and is volatile, fast and expensive. The secondary or external memory is implemented with devices as Floppy Disk Drives (FDD), Hard Disk Drives (HDD), Compact Disks (CD), Solid State Drives (SSD), USB flash drives (Pendrives), etc. and is permanent, slow and cheap.
- **Processing**. It is considered the basic computing operation. It implies the execution of the instructions of the programs with the data stored in memory.
- **Output**. The computer system represents the results of the processed data through the output devices like monitor, printer, speakers, etc.

1.3. Antecedents

Throughout history, the human being has tried to simplify the process of information management, especially when this involved repetitive tasks or complex calculations that caused its duration to be greatly dilated. They always try to avoid and accelerate manual and repetitive tasks using machines and tools, especially in calculation operations.

This has led to the invention of devices that facilitate the processing of data. In their origins, these devices were simple calculation machines that were evolving and perfecting to the current computers.

2. HISTORY OF COMPUTERS

Within the history of computers we can differentiate two stages:

- **The mechanical age** or **Generation 0**. At this stage the first mechanical calculating machines emerged. The automatic process machines of this generation were slow, unreliable and unwieldy.
- **The electronic age**. In this stage the electronic elements replace the mechanical elements. The devices invented in this era go from being mere calculators to real computers. Then begins a technological career in search of greater reliability and speed in a smaller space that continues to this day.

2.1. The mechanical age.

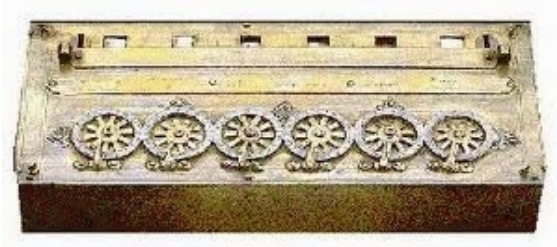
The first step in this process, more than 3.500 years ago, was the use of the abacus, which is an instrument used to manually perform arithmetic calculations. It uses beads that are manually slid over ten axes. It is a simple but very practical instrument that is still used in several countries.

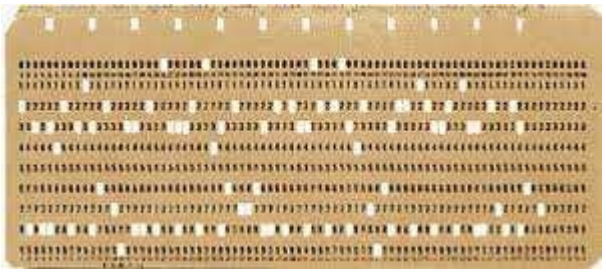
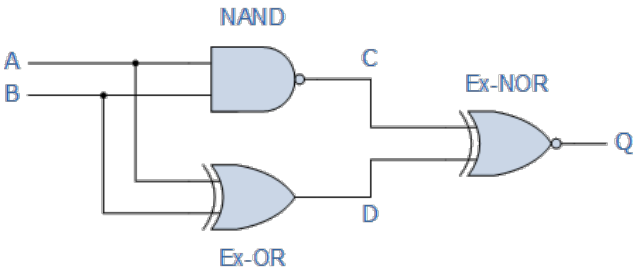
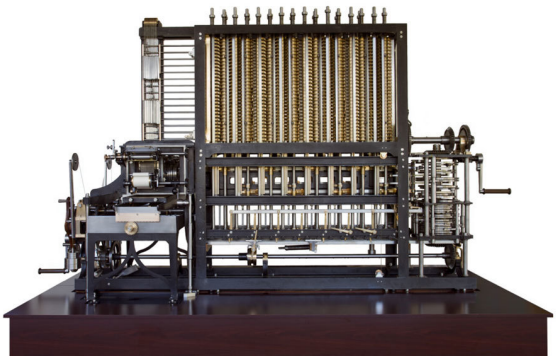
In the modern era, the first antecedents of data processing devices date back to the early seventeenth century. These are the Napier's bones or rods devised by John Napier (used to perform multiplications) and William Oughtred's circular slide rules (used to calculate logarithms).

In both cases, it was not even a question of machines, but of instruments that facilitated the calculations, although they represented a great advance and a considerable saving of time.

But the volume of operations to be carried out and the complexity of these calculations required a higher data processing capacity, which led, in the seventeenth century, to the invention of rudimentary calculation machines (composed of mechanical elements) capable of automatically process this data.

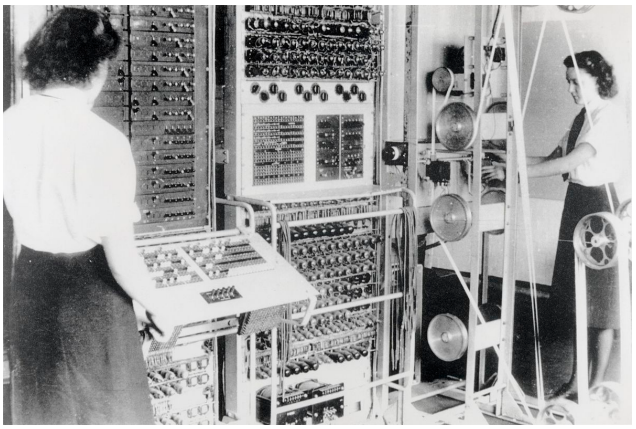
The mechanical age	
About the year 3.500 BC appeared the first calculator created by man, the **Chinese abacus**. This device allowed to represent numbers in the decimal system and perform operations with them. It consisted of some beads that were moved manually on 10 axes, in a similar way to the system to count goals in the table football. They are divided into two parts. The top beads are worth 5, while the bottom ones are worth 1.	

The mechanical age	
In 1.623 **Wilhelm Schickard** designed a mechanical **calculating machine** to assist in all the four basic functions of arithmetic (addition, subtraction, multiplication and division). It could help in the laborious task of calculating astronomical tables. The machine could add and subtract six-digit numbers, and indicated an overflow of this capacity by ringing a bell. The adding machine in the base was primarily provided to assist in the difficult task of adding or multiplying two multi-digit numbers. To this end an ingenious arrangement of rotatable Napier's bones were mounted on it. It even had an additional "memory register" to record intermediate calculations.	
In 1.642 the French scientist **Blaise Pascal** invented an arithmetic calculator that made additions and subtractions, the **Pascalina**. The result was displayed by small windows. It was formed by a series of cogwheels like those of watches. When rolling the 10 teeth of the first wheel, the second wheel advances one position. When rolling the 10 teeth on the second wheel, the third wheel advances one position, and so on. Each wheel has 10 teeth since the numerical system used is decimal, with base 10. They had as many wheels as digits we wanted to represent.	
In 1.670 the German mathematician **Gottfried Wilhelm Leibniz** perfected Pascal's machine and invented another machine, **Leibniz's arithmetic machine**, which in addition to adding and subtracting could also multiply and divide. This machine had reliability problems and was not used much.	

The mechanical age	
In 1.805 the French inventor **Joseph Marie Jacquard** designed an automatic loom that used **punched cards** to "program" certain designs or patterns.	
In 1.854, the English mathematician **George Boole** published the development of **Boolean algebra**, a system that schematized and developed the logical operations performed on the operators AND, OR and NOT. It is fundamental in the operation of the current computer systems, since they suppose the base from which electronic circuits capable of carrying out operations can be designed.	
In 1.880 the American statistician **Herman Hollerith**, based on the jacquard cards, devised a machine to process data. This machine was used to manage the population census of 1.890 of the United States. It was called the **Hollerith tabular machine**.	
In 1.822, the British mathematician **Charles Babbage** designed the **differential machine** of Babbage, although he never built it because the technology of the time did not allow it. This machine solved complex mathematical problems using polynomials. Charles Babbage together with the British mathematician **Augusta Ada Byron** are considered as the inventors of the modern digital computer. In 1.837 Babbage's **analytical machine** appeared, which already had many of the characteristics of a modern computer.	

The mechanical age

At the beginning of the 20th century, the first analogue computers were built, using axes and rotating gears. These computers were used to track torpedoes in submarines and bombs in aviation during the two world wars.
For example, in 1.943 the **Colossus**, which was used to decode the encrypted radio messages of the Germans. It was the first digital computer used with a specific purpose.



In 1.944 the IBM Automatic Sequence Controlled Calculator (ASCC), called **Mark I**, was the first computer built with electromechanical elements (**relays**) that worked without problems. It was a general purpose electromechanical computer that was used in the war effort during the last part of World War II.
One of the first programs to run on the Mark I was initiated by John von Neumann. It also computed and printed mathematical tables, which had been the initial goal of British inventor Charles Babbage for his "analytical engine".

In 1.931 the **Vannevar Bush differential analyser**, a system capable of solving differential equations. It introduces an important technological component during this first stage: the **vacuum tube**.
It was a mechanical analogue computer designed to solve differential equations by integration, using wheel-and-disc mechanisms to perform the integration. It was one of the first advanced computing devices to be used operationally.

2.2. The electronic age

In this phase, the electronic elements replace the mechanic ones, with what the machines go from being simple calculators to become real computers. Then begins a technological career looking for a greater reliability and a higher speed in the smallest space that continues to this day.

Due to the rapid advances in the world of electronics since the beginning of this era, computers are classified by generations, each of which is characterized by the components that are part of them.

The electronic age is divided into five generations: the first, second, third, fourth and fifth generations.

Relay

Vacuum tube

2.3. The first generation

It covers from the year 1.941 to the mid-50s. The main features of the computers of this era consist of:

- The main structural component was the **vacuum tubes**.
- The **programming** was done by **hardware**, through cables, switches and interconnections between the elements, so the programmer used to be the designer of the system.
- They did **not** have an **operating system** and the data was provided by **punched cards**.
- They were very huge, consumed a lot of energy, produced a lot of heat and noise, and broke down frequently.

The first general purpose computer of this generation was the **ENIAC** (Electronic Numerical Integrator And Computer), developed in 1.946 at the University of Pennsylvania for military purposes. It was a huge machine, weighing 30 tons. Its structure integrated 18.000 vacuum tubes. It worked on a decimal basis and was able to perform 5.000 sums per second.

The team responsible for ENIAC soon joined the mathematician John von Neumann, who provided the concept of a stored program (main memory stores programs and data). This led to an evolution, as a result of which emerged, in 1.951, the **EDVAC** (Electronic Discrete Variable Computer), which was the first computer with the program stored in memory. Unlike ENIAC, it worked in binary.

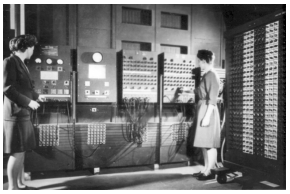
In addition, the introduction of software programming (using machine language) led to the appearance of a certain specialization, since independent programmers could appear who had not participated in the design of the hardware architecture of the machine.

Also in 1.951 the **UNIVAC-I** (UNIVersal Automatic Computer) was built, the first commercial computer in history, and in 1952 the **UNIVAC-II** arrived, which already included memories based on ferrite cores.

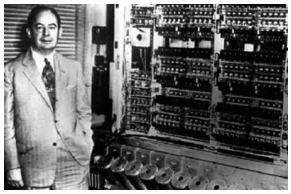
The **Colossus** was the first electronic digital programmable computing device (1.943), and was used to break German messages during World War II. It remained unknown, as a military secret, well into the 1970s. It was a non-general purpose computer.

This generation finished in 1.955 with the development of the first models of **IBM** computers (**701, 704, 705**, etc.).

ENIAC



EDVAC



UNIVAC-I

IBM 701

2.4. The second generation

It covers from 1.956 to 1.964. The main characteristics of the computers of this generation were the following:

- **Transistors** replace vacuum tubes, which allows a greater capacity of process in a smaller space, and with lower consumption and noise.
- Thanks to the appearance of the **operating systems**, **high-level programming languages** (FORTRAN, COBOL, ALGOL,...) begin to be used.
- The computers have **ferrite core memories**, which provides greater flexibility in the process and allows faster calculations by not having to depend on the delays caused by loading the data from the punched cards.
- The introduction of **magnetic storage** methods also increases the volume of data that can be stored.

As examples of computers of this generation, there are the **IBM (704, 1401, 1620)** and the **PDP-1** of DEC (Digital Equipment Corporation).

IBM 704

PDP-1

2.5. The third generation

The computers developed between 1.965 and 1.971 belong to this generation, which means the birth of **microelectronics**. As main characteristics of this generation, we can highlight:

- Computers started using **integrated circuits** or **chips**. In a single tiny device, several transistors and other electronic components were encapsulated.
- The speed of the computers increased a lot (100 million operations per second), its reliability grew significantly and a smaller size was achieved in them.
- Operating systems are developed that allow the use of **multiprogramming** and **time-sharing**, which meant a reduction in costs, since several programs could be run simultaneously, it was easier to make the initial investment profitable.

The first computer of this generation was the **360 series of IBM** (System / 360), which could share programs and peripherals and were the first to use microprogramming and integrated circuits.

IBM-360 series computer

Honeywell 6000 series computer

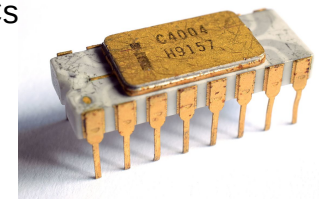
2.6. The fourth generation

This generation goes from 1.971 to 1.983 and is characterized by two fundamental aspects: the incorporation of **silicon chips** and the **miniaturization** of electronic circuits, which enabled the appearance of **microprocessors** (millions of electronic components stored on a chip). Therefore, the milestones that mark this generation are not marked by computer models, but by microprocessor models.

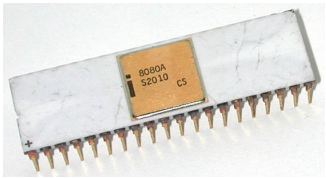
The first microprocessor was the **Intel 4004**, a 4-bit processor; but the first specifically designed for general use was the 8-bit **Intel 8080**, capable of performing 200.000 operations per second.

As a result of the miniaturization of the components, the computers became much simpler, smaller and cheaper than the previous ones, being able to be used by individuals. They are called microcomputers or Personal Computers (PC). The first were manufactured by companies such as **Apple Compute**r or **Commodore Business Machines**. This process was joined, in 1.981, by the **IBM** company with its Personal Computer (PC), from which it derives the name with which these devices are currently designated. The IBM PC included, as the main novelty with respect to its competitors, a standardized operating system for all of them, the **MS-DOS** (MicroSoft Disk Operating System), developed by **Microsoft**.

The term PC became popular and the identical computers that other companies manufactured were called compatible or cloned PCs the same components as the IBM and the same software.



Intel 4004 microprocessor



Intel 8080 microprocessor

Name	Signification	Year	Transistors number[46]	Logic gates number[47]
SSI	*small-scale integration*	1964	1 to 10	1 to 12
MSI	*medium-scale integration*	1968	10 to 500	13 to 99
LSI	*large-scale integration*	1971	500 to 20,000	100 to 9,999
VLSI	*very large-scale integration*	1980	20,000 to 1,000,000	10,000 to 99,999
ULSI	*ultra-large-scale integration*	1984	1,000,000 and more	100,000 and more

2.7. The fifth generation

Although for some authors there have been only four generations of computers and we would still be in the fourth, the truth is that it is commonly admitted the existence of a fifth generation that would begin in the early 80s.

The fact that gives rise to this generation is the design of computers with **parallel processing** capacity (computers can work, simultaneously, with several microprocessors, which greatly increases their calculation capacity and speed). This technology allowed **Seymour Cray** to develop the first supercomputers (**CRAY**), which are still being perfected today. These computers are huge and have the ability to perform billions of operations per second.

As a result of this progress, in 1.983 a research project was launched in Japan whose main objective was to build computers that use **Artificial Intelligence** systems (AI) at the machine language level (using the PROLOG language), to facilitate communication with the computer in a language similar to the human being. These computers also used microprocessors, worked in parallel and could **recognize voice** and images.

This project ended in 1.995 because the investments of time and money did not justify the results obtained. For this reason, for some authors this generation ends on that date, when a **sixth generation** begins; others, on the contrary, think that we are still in the fifth.

CRAY Computer

PIM/m computers system

3. THE VON NEUMANN ARCHITECTURE

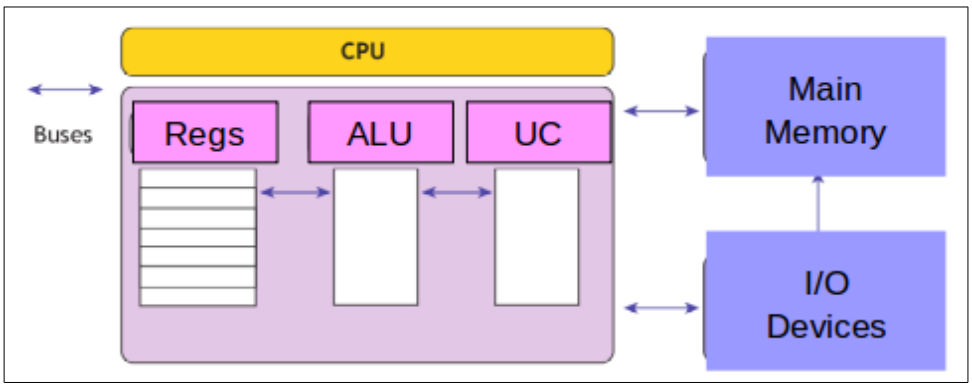
As we mentioned in the previous section, the equipment belonging to the first generation of computers had operating problems and could only be programmed by a few people who had participated in its construction and knew perfectly the architecture of the machine. This architecture was called Harvard architecture and was characterized because computers used storage devices for instructions and for data that were physically separate.

The team responsible for **ENIAC** joined the American mathematician of Hungarian origin, **John von Neumann** to solve these problems. von Neumann proposed an architecture that supposed an authentic revolution when devising the concept of **stored program**. When hardware programming is changed by programming through software programs that are stored together with the data, flexibility and versatility are gained. In addition, to be able to use the same program in different systems that had the same architecture, supposed the beginning of the software industry and the appearance of professionals specialized in programming.

As a result of their contributions, **EDVAC** and **UNIVAC-1** (the first commercial computer following this model) were developed.

In addition, the model proposed by von Neumann, which was called von Neumann architecture, was based on the division of the computer into independent functional units that were permanently connected, with one of these units being attributed the control and direction of the entire process.

Functional units of von Neumann architecture	
Central Processing Unit (CPU)	Arithmetic Logic Unit (ALU)
	Control Unit (CU)
	Registers (Regs)
Main Memory (MM)	
System buses	
I/O devices	



Scheme of the von Neumann architecture.

3.1. The Central Processing Unit (CPU)

The central processing unit (CPU) is the element of the computer whose function is to search in the memory the instructions contained in the programs stored there, interpret them, and execute them. It also processes the data entered through the input units and sends them to the output units. The CPU is composed of the following elements:

- **Control Unit (CU)**. It looks for the instructions stored in the MM, and interprets and executes them.
- **Arithmetic Logic Unit (ALU)**. It is responsible of performing arithmetic operations (addition, subtraction, ...) and logic operations (and, or, or exclusive, not) with the data received and generating the results.
- **Registers (Regs)**. They are in charge of temporarily storing small amounts of data (normally, intermediate results of operations) inside the CPU. There are work registers (general purpose), and special registers (not visible to the programmer).
- The set of connection line are called internal **buses** of the CPU. Their mission is to let the circulation of the data among the different elements of the CPU.

System Registers

The registers are memories of very little capacity and very fast access located inside the CPU. They are used by the CPU to store the instructions that will be carried out, the intermediate data to be used in these instructions and the resulting data after performing the operations.

They store only a small number of bits that can be handled in blocks. This number of bits, which is always a multiple of eight, usually coincides with the size of the system buses, and it is called **word size**.

The power of a CPU is related to the size of the registers (the larger the size, the greater the number of data that can be worked on simultaneously, and the larger the size of the memory that can be addressed). The first commercial processors had 8 bits, but this number has gradually increased to the present 32 or 64 bits. We can distinguish the following types of registers:

- **Work or general purpose registers**. They are visible to the user. They are only used directly by programmers who use assembler language, and are dedicated to programming drivers or compilers. They are divided into three categories:
  - **Address registers**: contain memory addresses.
  - **Data registers**: temporarily store data that is exchanged between the CPU and the memory.
  - **Status registers** or flags: store some conditions on the last operation executed to take them into account in subsequent operations.

• **Control registers**. They are used by the CPU for its operation. They have a specific mission and can not be accessed by the user. Among these we have: the Program Counter (PC), the Instruction Register (IR), the Memory Address Register (MAR), the Memory Data Register (MDR), etc.

Program

Sequence of instructions that a computer must interpret and execute.

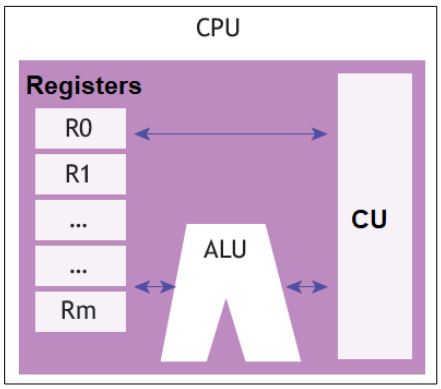
Instruction

Set of data inserted in a structured or specific sequence that the processor interprets and executes.

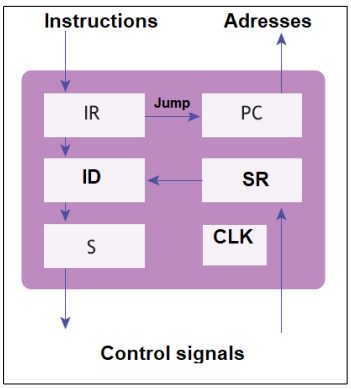
Status register

Although each CPU model has its own status bits, some of the most common are:

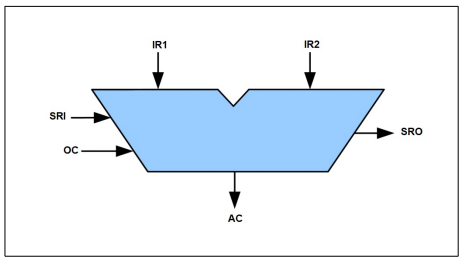
- ZF (zero flag): the result is zero.
- SF (sign flag): the result is negative.
- VF (overflow flag): the result exceeds the number of bits that can be operated.
- DF (direction flag): indicates the direction of operations (increase or decrease).



Detail of a CPU with von Neumann architecture.



Detail of the Control Unit.



Schema of the operational circuit of an ALU:

- IR1 and IR2: Input Register for the operands.
- AC: ACcumulator register . It stores the result of the operations.
- OC: Operation Code
- SRI: Status Register Input.
- SRO: Status Register Output.

Control Unit

The functions of the control unit (CU) are:

- Fetch in the memory the instructions of the program in execution, interpret them, and execute them.
- Generate the necessary control signals so that the rest of the components of the computer perform the appropriate tasks at the right moment, synchronized with the system clock.

The control unit is composed of the following components:

Component	Function
Program Counter (PC)	Contains the address in memory of the next instruction to be executed.
Instruction Register (IR)	Stores the instruction that is being executed at this time.
Instruction Decoder (ID)	Each instruction is divided into two parts: the operation code (OC) and the address of the operands (if the instruction requires them). The decoder interprets or decodes the operation code field to find out which operation to perform.
Sequencer (S)	Generates the necessary micro-orders to execute, step by step and in a synchronized way, the instruction.
System Clock (CLK)	It is an oscillator circuit that generates electrical impulses at a constant frequency, which synchronizes the execution of each instruction. Its speed is measured in hertz (Hz), according to the number of pulses per second. Each instruction may take one or more clock pulses to execute.
Status Register (SR)	Stores the different status conditions of the last operation.

The Arithmetic Logic Unit (ALU)

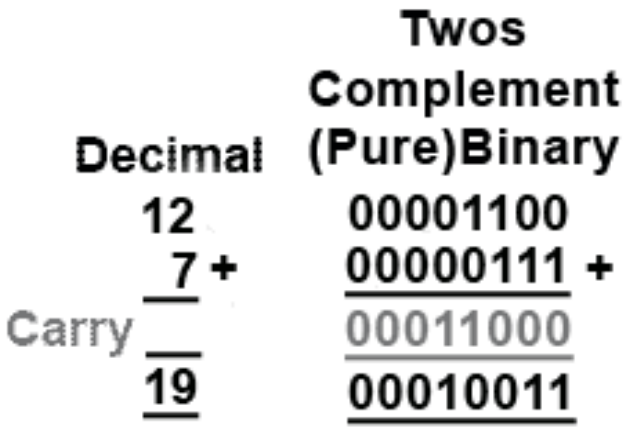
Its purpose is to perform arithmetic and logic operations with numbers under the control of the UC. It performs the following operations:

- **Arithmetic** operations with integers, mainly additions and subtractions. Some ALUs can also perform multiplications and divisions directly, without the need to decompose these operations into additions and subtractions.
- **Logic** operations between two numbers, normally comparisons, using the logical operators (AND, OR, NOT).
- **Bit shift** operations, which consist of moving an operand a specific number of bit positions to the left or to the right.

Two's complement

The two's complement of a binary number is obtained substituting the digits of the binary number for their complementary ones (the ones by zeros and the zeros by ones) and adding 1 to the result. For example:

- Decimal 45 = binary 101101
- Changing 0 and 1: 010010
- Adding 1: 010011



The ALU performs operations in two's complement format, which greatly simplifies its construction, since using this numbers representation, binary subtractions are transformed into sums.

The ALU is composed of:

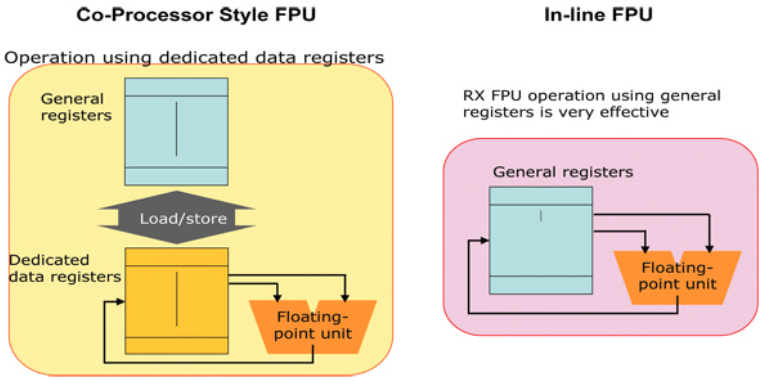
Component	Function
Operational Circuit (OC)	It is the component that performs the operations with the operands from the input registers.
IR1 and IR2	Input registers that contain the operands with which the operation will be performed.
ACcumulator register (AC)	Temporarily stores the result of the operations performed by the operational circuit.
SRI	Information coming from the Status register (flags).
SRO	Information going to the Status register (flags).
OC	Operation code (generally addition or subtraction).

Floating Point Unit (FPU)

Floating Point Units perform arithmetic operations between two real numbers in floating point format. These units were formerly called mathematical co-processors. Modern CPUs include it, but it was not present in the von Neumann architecture.

Like an ALU, they also perform arithmetic operations between two values, but for this they use the numbers in floating point representation, which is much more complex than the two's complement format, but which greatly reduces the time needed to perform the calculations.

Not all CPUs have an FPU. In case they do not have it, they emulate its operation through the ALU, which means a loss of speed.



Floating point (FP)

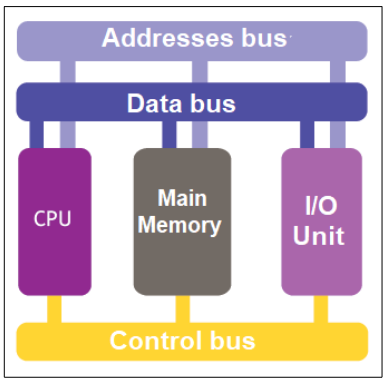
It is a form of scientific notation with which you can represent extremely large and small real numbers in a very efficient and compact way. Using FP we can perform arithmetic operations. The standard for floating point representation is **IEEE 754**.

The system buses

The buses of the system are paths, roads or routes (electrical or digital) through which the information circulates among the different functional units of the computer. Each bus connection transmits a bit of information, that is, a 1 or a 0 (current flows or current don't flows).

Within the computer system we distinguish three types of buses:

- **The addresses bus**. It transmits memory addresses. The larger the size of this bus, that is, the more bits it has, the greater the amount of memory that can be accessed. With an n-bit address bus, 2^n memory locations can be accessed.
- **The data bus**. It transports the instructions of the program in execution or the data with which the computer works between its different functional units. The more bits this bus has, the greater the amount of information that can be accessed at one time.
- **The control bus**. It transmits the control signals to guide the operations of the rest of the functional units of the computer. The more bits this bus has, the greater the number of signals available.



The system buses.

3.2. The Memory (M)

In a generic sense, the memory of a computer is used to refer to any device in which the information is stored in digital format, that is, in bits or binary values (0 or 1).

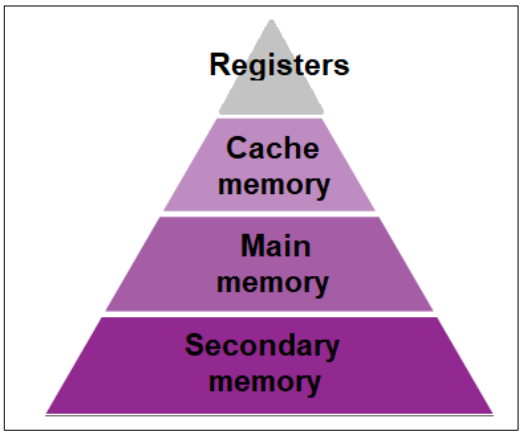
The first classification we can make of the memories differentiate them according to their location:

- **Primary memory**: it is located inside the CPU or the motherboard (registers, cache memory and main memory or RAM).
- **Secondary memory**: the CPU accesses it through the input and output units, because it is implemented in peripherals, for example: hard disk drives, flash memory cards, CD, etc.

The secondary memories have much more capacity than the primary ones and are much cheaper, although in exchange they are much slower. Therefore, a hierarchy of memories is established in which the highest level is represented by the most expensive, the fastest access and lowest capacity memories (the CPU registers). The lowest level of the hierarchy corresponds to the largest capacity, slowest access and cheapest memories (the secondary memories).

In another classification of the memories we can distinguish between:

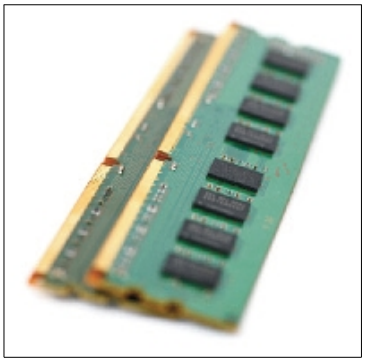
- **Volatile memories**: they must be electrically powered to keep the information stored, for example, the RAM. They allow to read and write the data stored in them.
- **Non-volatile or permanent memories**: the information remains stored even if the power supply is interrupted, for example, the ROM. They only allow reading the information stored in it, because the data are recorded in the factory and can not be modified.



Hierarchical pyramid of memories.

The units of measurement of information are used to measure the storage capacity of the memories. The basic unit of information is the **bit**, an acronym for Binary digIT. A bit is a digit of the binary numbering system (a 0 or a 1). It is the smallest amount of information that can be represented. Given the size of the memory, the bit is too small, so the **byte** or its multiples are usually used as units:

Unit	Description	Unit	Description
Bit (b)	1 or 0	Tera byte (TB)	1.024 GB
Byte (B)	8 bits	Peta byte (PB)	1.024 TB
Kilo Byte (KB)	1.024 B	Exa byte (EB)	1.024 PB
Mega byte (MB)	1.024 KB	Zetta byte (ZB)	1.024 EB
Giga byte (GB)	1.024 MB	Yotta byte (YB)	1.024 ZB



Main memory of a computer (RAM modules).

Number of bits	Accessible memory
1	2^1 = 2 B
4	2^4 = 16 B
8	2^8 = 256 B
16	2^{16} = 65.536 B
32	2^{32} = 4 GB
64	2^{64} = 18,4 EB

Amount of memory accessible according to the size of the addresses bus.

The Main Memory (MM or RAM)

The main or central memory is the element of the computer that stores the information and, therefore, the device from which the CPU receives the necessary data and instructions to operate, and where it keeps the results of its operations.

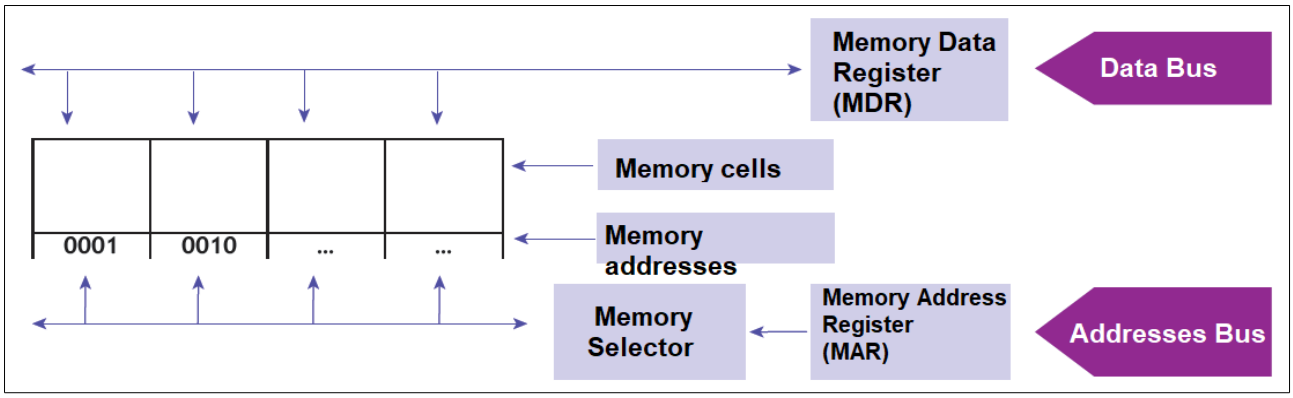
It receives the name of RAM, which is an acronym for the words Random Access Memory, to indicate that the access time to each memory address is the same, because you do not have to go through the previous positions. In other devices such as hard disk drives, the access time depends on the positioning of the R/W head in the previous operation, so they have different access time.

The main memory is composed of a series of cells numbered and identified by an address in which the information is stored in binary values (0 or 1). These cells are grouped into a certain number of bits. Each time an operation is performed in the memory, this whole set of bits is accessed, which has a unique address that identifies it: the **memory address**.

This element contains both the programs and the data they handle, so that, in order for any program that is going to be executed to be able to access the CPU, it must be previously loaded in RAM. The memory communicates with the CPU through the system buses already seen.

For carrying out read or write operations, the memory has two associated registers and a device responsible for selecting a cell.

- The **Memory Address Register (MAR)** contains the memory address of the cells you want to access to read or write information.
- The **Memory Data Register (MDR)** contains the information that you want to write in memory or that has just been read from memory.
- The **Memory Selector (MS)** connects the memory data register with the cell (whose address appears in the address register) in which it will perform a read or a write operation.



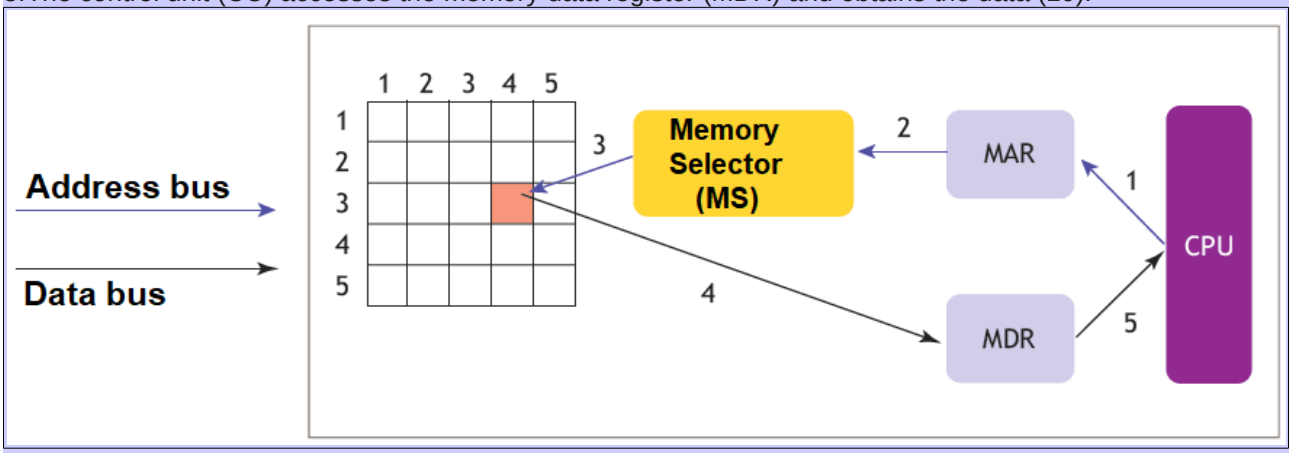
Schema of the main memory.

Examples

Memory read operation

A memory read operation involves the following steps:

1. The control unit (CU) loads the memory address in the memory address register (MAR) you want to read (43).
2. The memory selector (MS) tells the memory that we want to perform a read operation in the position 43.
3. The memory access is produced, what is in position 43 is read, for example, the number 10.
4. The data read (10) is loaded into the memory data register (MDR).
5. The control unit (CU) accesses the memory data register (MDR) and obtains the data (10).

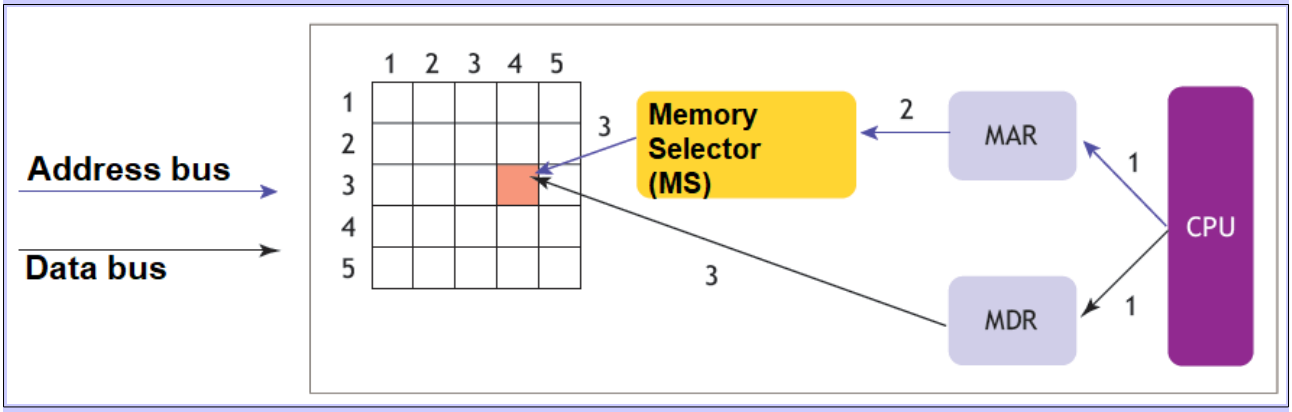


Examples

Memory write operation

A memory write operation involves the following steps:

1. The control unit (CU) of the CPU loads in the memory address register (MAR) the memory address to be read, in this case the 43. It also loads the data to be written, for example 27, into the memory data register (MDR).
2. The memory selector (MS) tells the memory that we want to perform a write operation on the position 43.
3. The memory access occurs and the number 27 is written in the position 43.

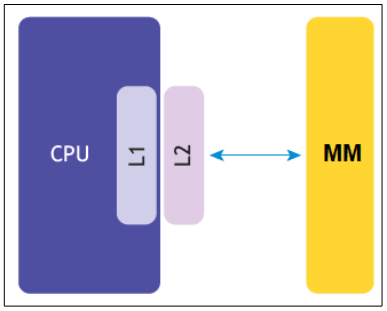


The cache memory

The cache memory is a buffer that is placed between the main memory and the CPU to accelerate access to memory. The cache is a much faster memory than RAM, but also much more expensive, so its size is small (10 Kb to 1 Gb).

The basis of this memory is the following: when accessing the RAM to read a data for the first time, a copy is made in the cache; the next time you want to access that data, you can directly access the copy made, which saves time, due to the higher speed of the cache.

There are several cache levels called L1, L2 and L3, although not all motherboards have all of them. The first ones are located within the encapsulation of the CPU, while the latter, if it exists, is located outside the CPU. Other computers with only two levels of cache memory usually carry the L1 inside the CPU and the L2 on the motherboard.

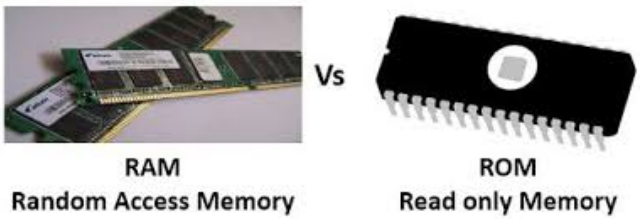


Location of the L1 and the L2 cache.

The ROM memory

The ROM name is an acronym for Read Only Memory and refers to a type of non-volatile memory that is recorded at the factory, whose purpose is to contain the equipment's start-up routines. These routines are basically two:

- **POST (Power On Self Test)**: when turning on the computer, it checks all the available resources in the system (RAM memory, peripherals, etc.), whose information is in a memory called RAM-CMOS.
- **BIOS (Basic Input-Output System)**: locates all the system's units and executes the necessary boot logs to load the operating system into RAM. As we will see in the next unit, this routine is currently stored in flash memories, which can be updated by the user.



Hybrid memories

For the construction of ROM memories (Read Only Memory), different technologies have been used:

- **PROM**: Programmable ROM. The data is recorded with special machinery outside the computer and can not be modified.
- **EPROM**: Erasable PROM. The data is recorded outside the computer and can also be deleted and re-recorded there.
- **EEPROM**: Electrically EPROM. The data is recorded and can be modified inside the computer, but remains stable without power
- **FLASH**: it is a volatile type of memory, but since it needs very little electrical power, its operation simulates that of a non-volatile one.

3.3. The cycle of execution of an instruction

We have seen that computers work with data and instructions that tell the computer what to do with that data. To execute each of these instructions, a series of steps are followed that are divided into two phases: search and execution.

Fetch phase

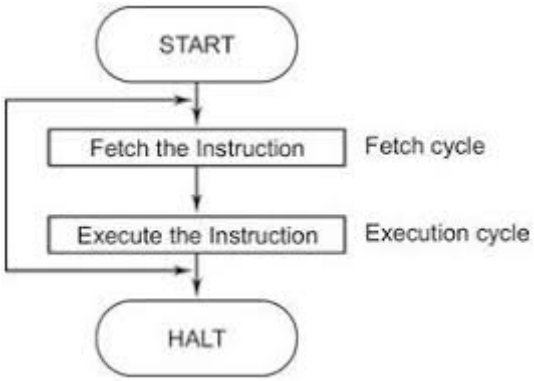
The first step to execute each instruction is to locate it in the RAM memory, where it is stored, and take it to the control unit to process it. This phase is integrated by the following actions:

- The program counter (PC) indicates the memory location where the next instruction to be executed is located. The control unit sends a micro-command, via the address bus, to load this address in the memory address register (MAR).
- The memory selector uses the contents of the MAR to access the indicated memory address, reads its contents and transfers it to the memory data register (MDR).
- The control unit (CU) accesses the memory data register, obtains the instruction and passes it to the instruction register (IR) through the data bus.
- The instruction decoder interprets the instruction that comes from the instruction register to find out what is the operation that must be performed and generates the necessary micro-orders to execute it informing the sequencer.

Execution phase

In this phase the actions indicated by the instruction are carried out, so it will be different depending on the type of instruction to be executed. The usual procedure could be the following:

- The address of the first operand is transferred from the control unit to the memory address register.
- The memory selector uses the contents of the MAR to access the indicated memory address, reads its contents and transfers the read data to the memory data register.
- The first operand is carried from the memory data register to the input register 1 (IR1) of the ALU.
- The same steps are repeated with the second operand, but in this case they are taken to the input register 2 (IR2).
- The sequencer generates a micro-orders so that the operation is executed, saving its result in the accumulator.
- The result is transferred, via the data bus, from the accumulator to the memory data register.
- The address where the result is to be saved is transferred, via the address bus, to the memory address register.
- The memory selector activates the cell where the result is to be stored and it passes from the memory data register to that cell.
- Finally, the program counter is incremented by 1 to point to the next instruction to execute.



Example

Execution of an instruction

Indicate the steps that the CPU would follow to execute the following program

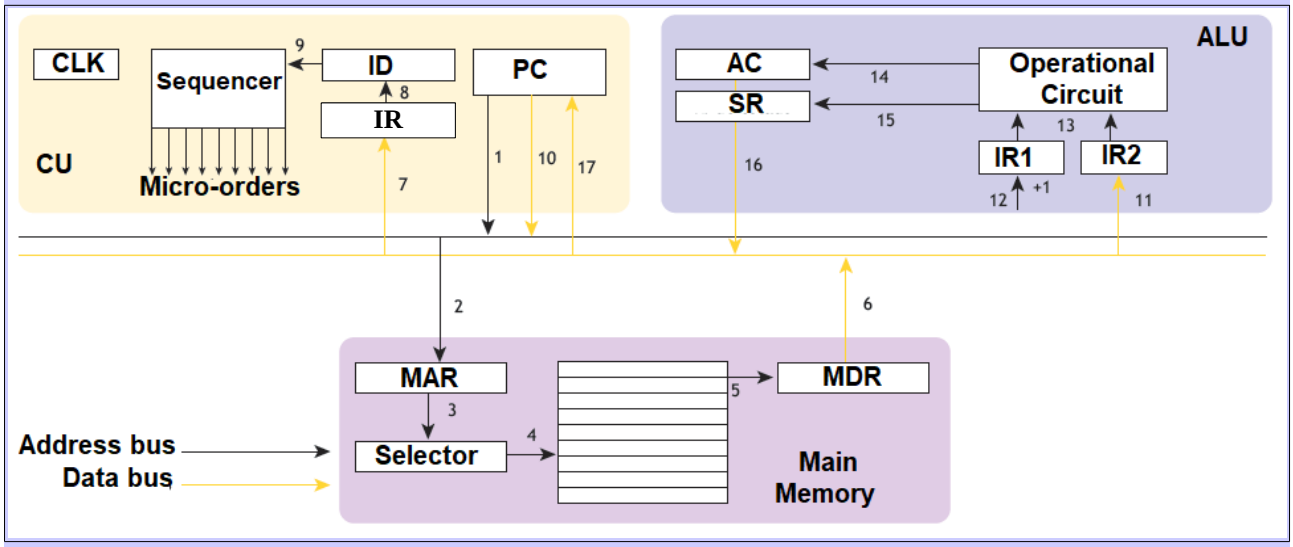
Solution

To perform this program, it is necessary that the computer executes several instructions and operate with various data. This would be done through a sequence of instructions in assembler language similar to the one shown in the following table:

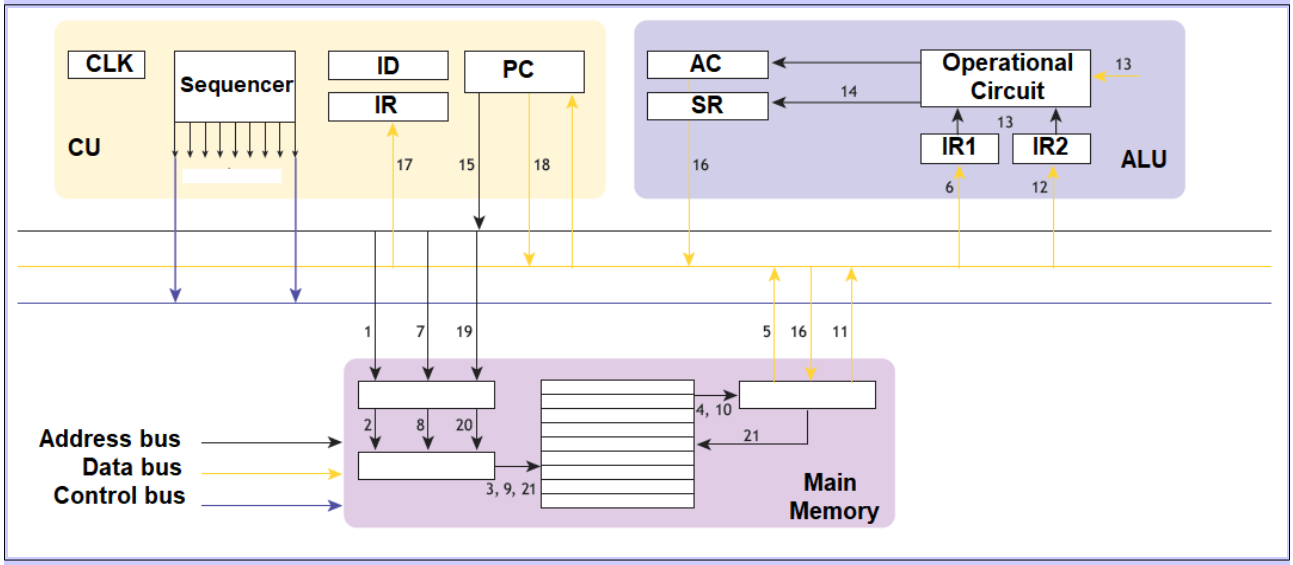
Memory address	Memory content			
101	OC	SO	DE	
	Read	#106	R1	
102	OC	SO	DE	
	Read	#107	R2	
103	OC	S1	S2	DE
	Add	R1	R2	R3
104	OC	SO	DE	
	Write	R3	#108	
105	End of program (EOP)			
106	A			
107	B			
108	C			

This process would be sequenced following the following phases:

- The content of the PC (PC = 101) is passed to the MAR via the address bus (1) and (2) and the memory selector indicates to the memory that it wants to perform a read operation at address 101 (3).
- The indicated memory position is accessed and its content is read, which is the first instruction (4) loaded by the MDR (5). The control unit accesses the data and transfers it to the IR through the data bus (6) and (7).
- The decoder analyses the instruction to find out which is the operation that must be performed (read #106) and generates the necessary micro-orders to execute it, informing the sequencer (8) and (9).
- The PC is increased by 1 (PC = 102) to point to the next instruction to be executed (10) to (17).



- Then the address of the first operand (#106) is loaded in the MAR, through the address bus, and the memory selector reads in the address 106. Its content is read and it is loaded in MDR, from where it is carried to IR1 through the data bus (1) to (6).
- Then the second instruction is processed (Read #107 R2), which would be carried out in the same way that we executed the first one (with the change that the address to read would be 102), increasing the program counter by 1 (PC = 103). Then the memory address 107 would be accessed to read the second data, it would be loaded into the MDR and from there it would be passed to IR2 through the data bus (7) to (12).
- The third instruction would be the one included in memory address 103 (ADD R1 R2 R3) and would be executed analogously to the first two, in this case, the decoder finds that the instruction to be executed is the sum (A + B), generates the necessary micro-orders to execute it, informing the sequencer and the PC is increased by 1 (PC = 104).
- At this moment, through the control bus, the sequencer sends a micro-order to the ALU to execute the addition operation (13) and stores the result of the operation (C) in the accumulator (14).
- Now we must write the result data in the memory. The content of the PC (remember it was the address 104) is passed to the MAR register through the address bus (15). On the other hand, the result C is sent from the accumulator to the MDR (16) and, through the data bus, it is passed to the IR (17). The decoder identifies that it is desired to perform a write operation of the result C in the memory address 108 and the program counter is incremented by 1 (PC = 105).
- Finally, through the address bus, the address of the memory cell where the program will be saved is loaded in the MAR. It is indicated to the memory selector that it is desired to perform a write operation in address 108 and the content of the MDR (the result of the operation) is transferred to that memory address (18) to (21).



3.4. The Input/Output unit (I/O)

The last element of von Neumann architecture are the input and output devices, commonly known as I/O devices or simply **peripherals**.

We can define peripherals, such as independent devices, connected to the CPU and the memory of a computer, which allow information to be entered in that equipment or extracted from it.

The role played by these devices is essential: through them you can enter the data that you want to process in the computer, as well as show the results of the processes carried out in the CPU; that is, they are the devices that allow communication between the computer and the user.

Peripherals can be classified as:

- **Input peripherals**: allow entering information to the computer for further processing. The I / O devices transform the input data into electrical signals that are stored in the main memory. For example: the keyboard.
- **Output peripherals**: those that receive information from the CPU (in digital format), convert it to a format understandable to the user and show it to the outside. For example: the monitor.
- **Input and output peripherals**: they are capable of both entering information in the system and extracting it. At the same time, they are classified as:
  - **Storage peripherals**: used to store the data used by the CPU once they have been deleted from the main memory. They are called secondary memory of a computer. For example: hard drive, flash memory, etc.
  - **Communication peripherals**: allow communication between two computers or between a computer and a peripheral. For example: modem, router, network card, etc.
  - **Others**: multifunction printers (MFP), touch screens, interactive whiteboards (IWB), ...

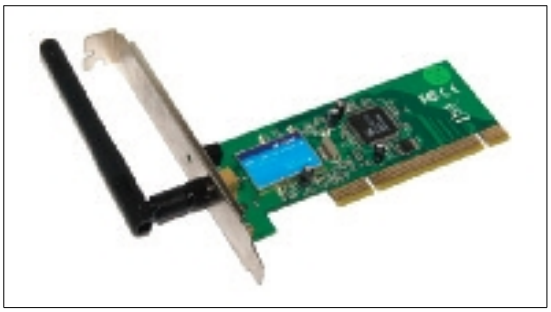
The I/O devices are composed of:

- A **mechanical part**: it is the physical device that allows entering or extracting information from the equipment. For example, a mouse or speakers.
- A **device controller** (**driver**): is a computer program through which the computer operating system controls the physical device and communicates with it. Each device has a different driver, it is even possible for a single device to have several different possible drivers that allow different functionalities of the same.

Input peripheral.

Output peripheral.

Storage peripheral.



Communication peripheral.

4. REPRESENTATION OF THE INFORMATION

4.1. Basic concepts

The **information** is the raw material of computing. It is formed by the **instructions** of the programs in execution and the **data** they manage. Both of them must be encoded in some **binary code**, which is the only code that the computer understands. Therefore, any information must be translated into binary before being entered into the computer.

An **alphabet** is a set of **symbols**.

A **code** is an association table between symbols of two different alphabets. For example: Braille code, Morse code, machine code, etc.

Computers, due to technical and electrical constraints, can only use an alphabet with two symbols (0 and 1), hence it is called binary code. All the software (data and instructions) is encoded in codes of two symbols. There are several binary codes. All use two symbols (0 and 1) but combine them differently.

There are two types of codes:

- **Numeric codes**: Allow coding of numerical values. Examples: Natural Binary, BCD, etc.
- **Alphanumeric codes**: Allow coding of numeric and non-numeric (alphabetic) values. Examples: ASCII-7, ISO-8859-1, Unicode, etc.

4.2. Numeric codes

The binary code

It is the code used by computers and only has two symbols (0 and 1) that represent two types of electrical voltage (current passes or not). Therefore, its base is 2. It is a weighted code, that is, the value of each digit, its weight, depends on the position it occupies.

The minimum unit of information is called **bit** (b). A set of 8 bits form a **byte** (B). A set of 4 bits form a **nibble**.

The decimal code

It is the code used by the human being and has ten symbols (0,1, 2, 3, 4, 5, 6, 7, 8 and 9). Therefore, its base is 10. It is also a weighted code. We use this base since we started counting using the fingers of our hands, which are 10.

Conversion from binary to decimal

As we have already said, the natural binary code is weighted: each digit has a weight based on its position. The decimal number will be the sum of the weights of each bit (decomposition in sums of the digit by the weight).

The weight of a digit is $X * 2^i$, where X is the digit, and i is its position.

Example: the binary number 11 has the decimal value 3.

$1 * 2^1 + 1 * 2^0 = 2 + 1 = 3$

Example: the binary number 1001 has the decimal value 9.

$1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 8 + 0 + 0 + 1 = 9$

Conversion from decimal to binary

Based on successive divisions of the decimal number by 2 till we obtain 0 as quotient. Then we will take the remainders in reverse order and we have the binary number. It is similar to factoring the number.

Example: the decimal number 90 corresponds to the binary number 1011010.

90 : 2	=	45	Remainder	0
45 : 2	=	22	Remainder	1
22 : 2	=	11	Remainder	0
11 : 2	=	5	Remainder	1
5 : 2	=	2	Remainder	1
2 : 2	=	1	Remainder	0
1 : 2	=	0	Remainder	1

The octal and the hexadecimal codes

Usually used to simplify the notation, since they are more compact than the binary (they need fewer digits to express the same amount) and it is very easy to convert the number between these encodings whose base is a power of two. The octal has base 8 (2^3) and the hexadecimal 16 (2^4). This causes one octal digit to correspond to three binary digits, while one hexadecimal digit corresponds to four. The digits used by the octal are: 0, 1, 2, 3, 4, 5, 6 and 7. The digits used by the hexadecimal are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

Hexadecimal	A	B	C	D	E	F
Decimal	10	11	12	13	14	15

Conversion from octal to decimal

The octal code is also weighted, so the decimal number will be, as in the binary, the sum of the weights of each digit, but in this case the base is 8.

The octal number 123 has the decimal value 83.

$1 * 8^2 + 2 * 8^1 + 3 * 8^0 = 64 + 16 + 3 = 83$

The octal number 20 has the decimal value 16.

$2 * 8^1 + 0 * 8^0 = 16 + 0 = 16$

Conversion from hexadecimal to decimal

The hexadecimal code is also weighted, so the decimal number will be, as in the binary, the sum of the weights of each digit, but in this case the base is 16.

The hexadecimal number 71B has the decimal value 1819.

$7 * 16^2 + 1 * 16^1 + B * 16^0 = 7 * 256 + 1 * 16 + 11 * 1 = 1819$

The hexadecimal number 20 has the decimal value 32.

$2 * 16^1 + 0 * 16^0 = 32 + 0 = 32$

Conversion from decimal to octal

Based on successive divisions of the decimal number by 8 till we obtain 0 as quotient. Then we will take the remainders in reverse order and we have the octal number.

The decimal number 90 corresponds to the octal number 132.

90 : 8 = 11 Remainder 2 ▲
11 : 8 = 1 Remainder 3 |
1 : 8 = 0 Remainder 1 |

Conversion from decimal to hexadecimal

Based on successive divisions of the decimal number by 16 till we obtain 0 as quotient. Then we will take the remainders in reverse order and we already have the hexadecimal number.

The decimal number 90 corresponds to the hexadecimal number 5A

90 : 16 = 5 Remainder 10 ▲ ($A_{hex} = 10_{dec}$)
5 : 16 = 0 Remainder 5 |

Conversion between Binary, Octal and Hexadecimal

The conversion between codes with base 2 is trivial, it is enough, starting from the digit more to the right and going to the left, grouping or dividing digits in groups of the power of two that represents the base, filled with zeros at the left if it is needed to complete the first group.

Binary → Base 2 = 2^1 → Groups of 1 bit
Octal → Base 8 = 2^3 → Groups of 3 bits
Hexadecimal → Base 16 = 2^4 → Groups of 4 bits

Octal	Binary	Hexadecimal
010 = 2 100 = 4 101 = 5 ↔ 245	10100101	1010 = A 0101 = 5 ↔ A5

decimal	hexadecimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂*

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

4.3. Alphanumeric codes

ASCII

The ASCII uses 7 bits to encode each digit, so it only encodes 128 characters (2^7). With these 128 characters are encoded the English alphabet characters, the numbers, the punctuation symbols and some non-printable characters. Numerous symbols of other alphabets are missing.

ISO-8859-1

It is an extension of the ASCII character set that uses 8 bits to encode each digit, so it encodes 256 characters (2^8). In addition to the 128 characters already included in the ASCII, it allows encoding other characters such as: ñ, ß, ç ¸ ... so it is also known as Latin1 code.

Unicode

It is the encoding standard to collect all the alphabets of the world. It has different implementations: UTF-8, UTF-16, etc. The UTF-8 is one of the most used implementations. Use up to 4 bytes per character. It is capable of representing all Unicode characters (2^{32}). The problem is that the codes have a variable length.

5. THE SOFTWARE

Computers are made up of two parts:

- A physical part, called **hardware**, that integrates the tangible elements (CPU, memory, peripherals, etc.).
- A logical part, called **software**, which includes both the programs that allow the operation of the computer and its communication with users, and other intangible elements necessary for the operation of the equipment.

The federal standard 1037C of the Institute for Telecommunication Sciences of the USA defines the software in a very complete way as:

> A set of computer programs, procedures and associated documentation concerning the operation of a data processing system.

The software of the computer system can be classified into three main categories:

- System or base software
- Application software
- Programming software

5.1. System software

The system software is the one that serves as the basis for the other types of software (hence its name) and its function is to serve as an interface between the user and the hardware, controlling it completely. For example: operating systems, device drivers, etc.

An operating system is the set of programs that manage to organize the central processor and peripherals as a unit of work. Its mission is also to direct peripheral's activities by transferring information between them and the computer, as well as serving as an interface between the user and the CPU.

The operating systems most used in practice are:

- **Microsoft Windows**: XP, Vista, 7, 8, 10, ...
- **Apple MacOS**: X, Tiger, Leopard, Snow Leopard, Lion, …
- **GNU/Linux** and its most known distributions: Ubuntu, Debian, Fedora, Red Hat, Lliurex, …

5.2. Application software

The application software (app or application for short) is a computer software designed to perform specific tasks related to the fulfillment of the user's objectives. For example: text processors, spreadsheets, databases, photo editor programs, accounting applications, web browsers, media players, aeronautical flight simulators, console games, ... The collective noun application software refers to all applications collectively. This contrasts with system software, which is mainly involved with running the computer.

5.3. Programming software

The programming software is what allows to create and develop other computer programs through the use of programming languages. For example: compilers, assemblers, debuggers, interpreters, linkers, editors, etc.

An integrated development environment (IDE) is a combination of all these software.

Programming software is also known as programming tool or software development tool. It is a program or set of programs which helps the software developers by assisting them in creating, debugging and maintaining other programs and applications. According to some sources, it is a sub-category of system software instead of a separate category of software along with application and system software.

To create programs we use programming languages. They usually have a very strict syntax intended to be translated literally into machine language. There are three types of programming languages: compiled, interpreted and hybrid.

Compilers translate all the source code to machine language (object code and executable code).

Interprets translate the source code line by line, executing it immediately and not generating any object code.

Compiled	Interpreted	Hybrid
C, C++, Pascal	PHP, Python	Java, Visual Basic .NET

5.4. Software licenses

The software license is a contract by which the owner of the exploitation rights agrees with the user of it, the conditions of use of that software (if he can or not copy it, distribute it or modify it, and in what way).

There are many types of licenses, of which only the main ones will be mentioned here.

Free software

Free software is the one that allows anyone to use it, copy it, modify it and distribute it. This does not imply that it is free of charge, since it can be of payment. It operates under open source licenses and the users have the code with which the software has been designed (source code).

One of the most widely used licenses of this type is the GPL (General Public License). Under the terms of this license, the author of the software, without giving up his owner rights, allows that software to be used, adapted, improved and distributed freely, according to the terms expressed in the license itself, so that the resulting software remains within the terms of the license.

Software not free

This type of software is one whose redistribution or modification is prohibited (the source code is not available). Therefore, its licensing mode is called closed source. It is also known as proprietary software.

Freeware

It is software that is distributed free of charge regardless of the license mode you use. However, since this is common in free software, the term freeware is often used in closed-source cases where free use of the program is permitted.

Shareware

It is a type of software that allows free use only during a period of time of evaluation or that has a free version with reduced functionality.

There are different types of software licenses:

- Closed and protectives: Windows, Oracle, etc.
- Opened and protectives: GPL.
- Opened and permissives: MIT