Computer Systems

UNIT 01. FUNCTIONAL ELEMENTS OF A COMPUTER

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📚 Importante

✏ Atención

📢 Interesante

INDEX

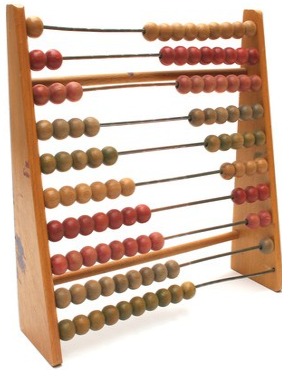# UNIT 01. FUNCTIONAL ELEMENTS OF A COMPUTER

1. HISTORICAL EVOLUTION

Figure 1. Abacus

1.1 Non-digital computers

The Abacus was invented in 500 B.C. and it was the first computer of the history. It was created to perform quick arithmetical operations and it was the fastest calculator machine since the arrival of other mechanical calculators, like Pascaline created by Blaise Pascal (XVII Century) or Bagage Machine created by Charles Bagage (XIX Century). These were machines with mechanical elements that performed basic arithmetical operations.

Also, fundamentals of modern computers were set: binary code (Gottfried Wilhelm Leibniz, XVII century), Bool algebra (George Bool, XIX century) and Turing machine (Alan Turing, 1936).

🔊 Bool defined the rules for working in a binary algebraic system, that is, in which there are only two digits (0 and 1) and not 9 as in the decimal system. In this way any quantity can only be represented by defining two states. This system is basic in today's computers, which contain elements that can be located in these two states[1].

1.2 Digital computers

Electrical energy gave us a new way to build computers more reliable and powerful.

The first programmable binary computer was Z1, created by German engineer Konrad Suze in 1938. In U.S.A., John Atanasoff and Clifford Berry created ABC in 1939. These were the first machines that used electrical switches to store information (a "switch off" meant zero, a "switch on" meant one).

Later, there were several programmable computers like Colossus (1943) and ENIAC (1946) that were the first to use vacuum tubes[2] to store information.

The problem of those computers is that they were hard to re-program, because the programming was done modifying hardware by hand and it was hard and expensive.

1.2.1 General purpose computers

In 1946 EDSAC was created. It was the first machine with internal instructions, making easy to be re-programmed by hand.

In 1949 EDVAC was born and was the first computer with stored programs. It was a revolution because that computer was able to run different kinds of programs without modifying his hardware. It was the birth of "Software".

---

1   We'll study the binary system in unit 2
2   Vacuum tubes are devices that controls electric current between electrodes in an evacuated container. Used to represent 0 and 1.

1.2.2 Transistor and Integrated circuits

The vacuum tubes had several problems. They were big, unreliable and consume a lot of energy, so engineers needed to find a system to reduce them.

In 1947 the transistor was invented and it replaced vacuum tubes, reducing size of the computers, saving energy and making them more reliable

Transistors were a hit but, like vacuum tubes, they have to be wired to connect all the components together. To solve that problem Integrated Circuits (IC) were invented in 1949. They put several transistors in one package. Transistors and integrated circuits help to shrink computers during 1960s.

In 1970s personal computers were born. There were a lot of them (MITS Altair, Apple II, IBM 5100, Mark 8, etc.) but the most successful was IBM PC (1981). It was the base from most of the personal computers that we use today.

In 1990s, personal computers are starting to be connected to Internet. In 2000s most of them have access to Internet.

Today, there are computers everywhere (desktop PC, laptop PC, mobile phone, tablets, smart watch, fitness devices, smart TV...).

2. DEFINITION OF COMPUTER SYSTEM

What is a computer system? Man have always wanted to build machines that help them to solve their problems. But they needed to build a machine for each problem.

The idea was: can we build a machine to resolve different problems? The answer is: yes, we can. This machine is called a computer system and its main characteristic is that It is composed of two parts: hardware (the physical elements) and software (the instructions that tell the hardware what it have to do, how to solve the problem). Sometimes, some authors include two parts more: the user and the data.

In this didactic unit we are going to talk about functional elements of a computer, that is, about the conceptual elements that are part of any computer system and that in real life will be part of the hardware.

3. COMPUTER ARCHITECTURES

3.1   Von Neumann architecture

Von Neumann architecture is an abstract model created in 1946 that describes a computer architecture. In Von Neumann architecture the main elements of a computer are:

- CPU (Central Process Unit).

- Memory unit.

- Buses.

- I/O (Input/Output to external devices).

Von Neumann architecture works as described:

- Components are connected by buses.

- It can execute elemental instructions (machine code).

- The instructions are stored in the memory unit. Also the data has to be stored in memory unit.

- Both instructions and data could be provided by input devices or generated by other programs (compiler, linker, etc.).

- The CPU reads an instruction and the control unit generates each signal to perform the instruction.

  - If it is required, CPU can perform arithmetic-logic operations.

- The CPU results can be stored on the memory unit or it can be sent to output devices.
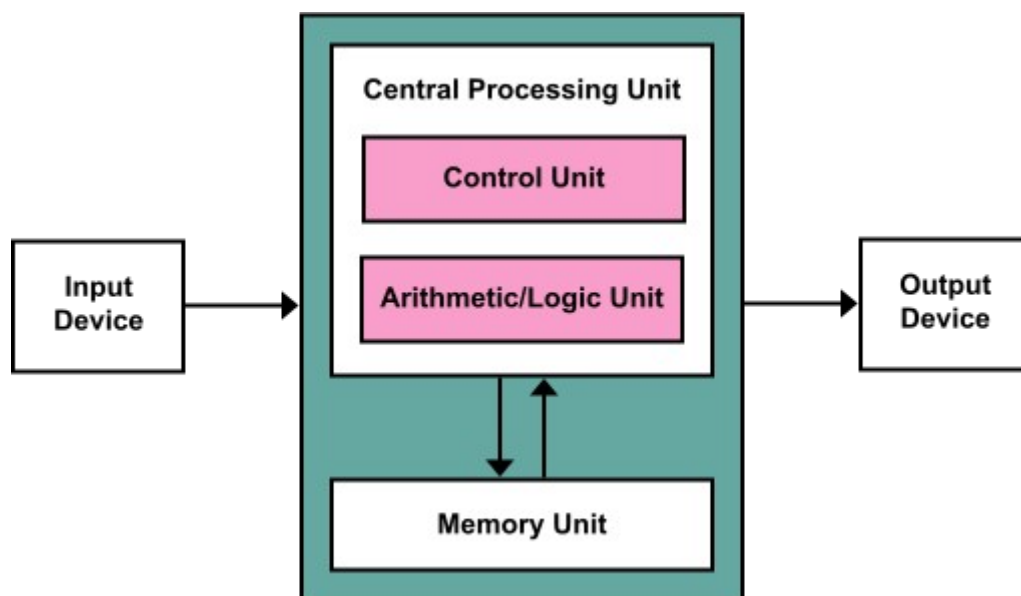
Figure 2. Von Neumann architecture

🔊 Nowadays, there are several improvements of this architecture, including features like mapped I/O (memory unit and I/O devices are treated like the same memory).

3.2 Harvard architecture

Harvard architecture is other abstract model that describes a computer architecture. It is less implemented than Von Neumann architecture.

The main differences between Harvard architecture and Von Neumann architecture are:

- In Von Neumann architecture, data and instructions are stored on the same memory. CPU only can read an instruction or a data at the same time. In Harvard architecture, data and instructions are stored separately an each one has its own bus. CPU can read both at the same time (an instruction and a data).

- In Von Neumann architecture, data and instructions are stored on the same memory and they share address space. In Harvard architecture, they are stored in different memories.

> ✎   Usually, real computer don't use pure Von Neumann architecture or pure Harvard architecture. The only are abstract models.

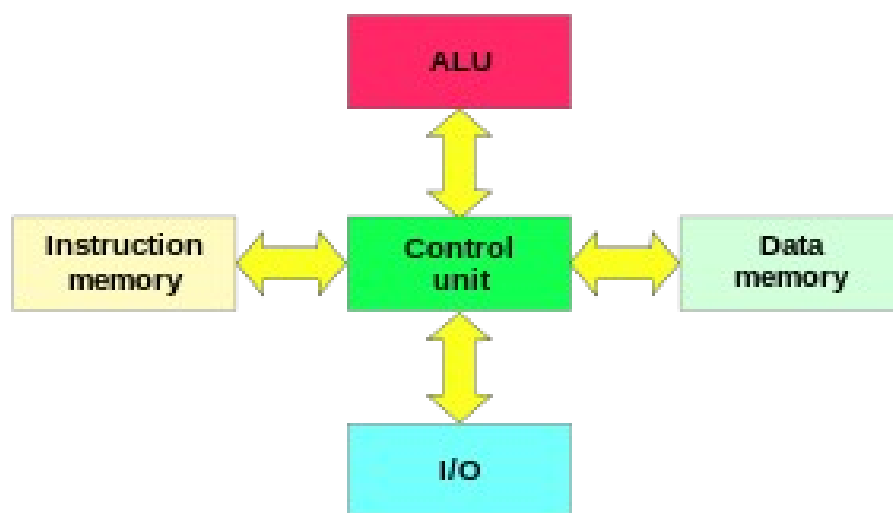Figure 3. Harvard architecture diagram

4. FUNCTIONAL ELEMENTS OF A COMPUTER

In this point, we are going to describe functional elements of a computer that implements Von Neumann architecture.

4.1   CPU (Central Process Unit)

The CPU is the core of the computer. CPU read instructions from memory and execute them, doing the required operations. It is composed of:

- **Registers**: they are little and fast memories. They are used as temporal storage when an instruction is being executed or to store temporally results of operations.

  The most important registers are:

  - **Program counter (PC)**: an incremental counter that stores the memory address of the <u>next instruction</u> that is going to be executed.

  - **Memory address register (MAR)**: it stores the address of a block of memory for reading from or writing to.

  - **Memory data register (MDR)**: a two-way register that stores data fetched from memory (and ready for the CPU to process) or data waiting to be stored in memory.

  - **Instruction register (IR)**: a temporally register to store the instruction that has just been fetched from memory.

  - **General purpose registers**: those registers are used to store information temporally, for example to store operands of and addition or to store the result of an arithmetic-logic operation.

  - **Flags**: These are very simple records. They can only be in two states. on or off. They are used to indicate the status of an operation. For example, flag Z is set to 1 if the operation has resulted in 0., N is set to 1 if the result is negative.

- **Internal bus**: is an internal bus that connect registers, control unit and arithmetic-logic unit.

- **Control unit (UC)**: it decides what to do in each moment. It's the *boss.* Its components are:

  - **Decoder**: its function is to decode instructions and prepare all the signals to execute the program properly. For instance an instruction can be ADD (which adds up two numbers that are in memory). When it comes to executing, the decoder generates a lot of small signals such as reading the first number of the memory, taking it to the ULA, reading the second number, taking it to the ULA, perform the operation (adding) and saving the result in memory. In this case an instruction has generated six signals [3].

  - **Clock**: is the system that sets the pace at which things are done. It's like the conductor's baton. I allows all system components to operate perfectly together.

---

3   Here, the process es simplified. Actually many more signals are generated

- **Arithmetic-Logic Unit (ULA)**: it is a unit designed to perform arithmetic operations (addition, subtraction, multiplication,…) and logic operations (not, and, or, …). It's the *worker*.

To execute an instruction the CPU follow those steps:

- Use the *Program Counter* to fetch the new instruction from memory.

- Decode the instruction and fetch data from memory if it is required.

- Perform the instruction and store data in an internal register or memory .

The instruction cycle will be detailed in later in the section *Instruction cycle*.

> To measure the speed of a CPU we use frequency units that indicate the number of times a thing repeats itself per unit of time. In general we use the Hz (or its multiples) that indicates the number of operations that can be performed in a second.

4.2   Memory unit

The memory unit is an internal memory that contains the instructions to execute and the data or programs to use. It consist of several cells that store 0 or 1 (1 bit).

A set of cells of a determined size is called *word*. Computers usually use words of 32 bits or 64 bits. Each word has an address associated with it. When you read information of the memory unit, you have to read the entire selected word.

For instance, a memory of 192 bits has 6 words of 32 bits. The memory addresses of those 6 words will be from 0 to 5. The first word will go from bit 0 to 31, the second from 32 to 63, etc.

Usually this memory is RAM (Random Access Memory). RAM is volatile memory (it needs energy to store information, without energy the information is gone). Also the time it takes to read / store a cell is the same for all words.

> Memory speed is measured by the number of words that can read/store in a time unit. It is usually measured in Hz (number of words that can be read/stored in a second).

4.3   Other memory types

Other types of memory present in a computer are:

- **ROM**: Read Only Memory. Is a Read-Only memory. It can not be erased. Usually it is used in old BIOS.

- **EPROM**: Erasable Programmable Read-Only Memory. It is a non volatile memory. It is a read-only memory, but it can be erased (with ultraviolet light) and re-written. Usually it is used in BIOS, that are commonly read but rarely written.

- **EEPROM**: Electrically Erasable Programmable Read-Only Memory. It is a non volatile memory. It is a read only memory, but it can be erased electrically and re-written. Usually it is used in BIOS ROM, that are commonly read but rarely written.

- **Flash**: is an evolution of EEPROM. It lets to write and read in several memory places at the same time. Usually it is used in USB memories. Nowadays it is the most used technology to store computer BIOS.

4.4   I/O (Input / Output external devices)

 The I/O devices are the devices used to obtain information (input devices) or to provide information (output devices). They are usually called peripheral devices. A mouse or a keyboard are examples of input devices and a printer or a screen are examples of output devices. Also there are devices that are input and output at the same time (a hard disk, a touch screen, internal components like a sound card, graphic card, expansion port,…).

> ☕ There is a trick to know if a device is a peripheral or not. If it is part of the Von Neumann machine it is not a peripheral. Otherwise, it is.

4.5   Buses

 A bus is a set of wires that connect components of the computer.

 A bus is defined by its boundary (number of bits that can be sent in one operation). Usually it is 32 bits, 64 bits,128 bits, etc.

> ☕ The bus speed can be measured in the number of operations that can be performed in a time unit. Usually it is measured in Hz (operations that can be done in one second).

Physically, buses can be classified like:

- **Serial bus**: buses send information by one wire each time. They are very simple and require a simple hardware to be managed. For this reason, in spite of using only one wire, the obtain very good performance. For example, SATA and USB are serial buses.

- **Parallel bus**: buses send information parallel, by several wires. They are complex to process and require complicated hardware. Usually it obtains less performance than serial bus. For example, the old "com" port or the old printer port were parallel buses.

 By the place were buses are placed buses can be classified like:

- **Internal**: connect "internal components" of a component. For example "internal components" of CPU: Registers, arithmetic-logic unit and control unit.

- **External**: connect components, for example CPU with memory unit, CPU with I/O devices, etc.

Von Neumann architecture distinguish between three external buses:

- **Control bus**: it carries the control signals from the control unit of the CPU.

- **Address bus**: it carries the memory address to read / store information.

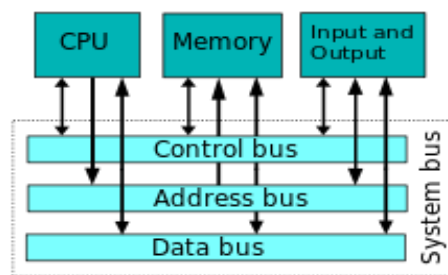- **Data bus:** it carries the data and de instructions to be read / stored.

Figure 4. Buses

4.6   Instruction set

CPU has a set of instructions that can be performed. The instructions could be:

- Arithmetic-logic operations: perform arithmetic-logic operations.

- Data handling and memory operations: read and store date from/to memory and from/to input/output devices.

- Control flow operations: operations that change the flow of the program modifying the *Program Counter* (PC).

5. INSTRUCTION CYCLE

An instruction is the data that CPU interprets to perform an operation.

When we talk about a CPU of 16 bits, 32-bit, 64-bits, … we are talking about the number of bits than an instruction can use.

Usually an instruction is divided on two fields:

- Operation code: is the code that indicates the CPU the type of operation to perform.

- Operation data: is the data associated with the operation that is going to be performed.

The instruction cycle is the steps that CPU takes to obtain the next instruction and perform it.

The instruction cycle is similar in every computer model. The steps are:

1. **Fetch the instruction**: the next instruction is fetched from the memory address that is stored in the Program Counter (PC) and it is stored in the Instruction Register (IR). At the end of this operation, the Program Counter (PC) points to the next instruction (that will be read at the next iteration of the cycle).
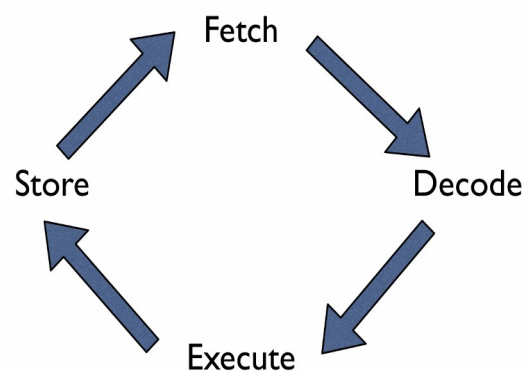

Figure 5. Instruction Cycle

2. **Decode the instruction**: during this cycle the encoded instruction present in the Instruction Register (IR) is interpreted by the decoder of the CPU.

3. **Execute the instruction**: The control unit of the CPU passes the decoded information as a sequence of control signals to perform the actions required by the instruction. (Example: reading values from registers, passing them to the ALU, writing the result back to a register).

4. **Store**: The result generated by the operation is stored in the main memory, in a register or sent to an output device.

   The Program Counter (PC) may be updated to a different address from which the next instruction will be fetched. (It is the way to jump from one place of the code to another).

When the cycle ends, it is repeated again.

6. EXAMPLE OF AN INSTRUCTION SET

In this point we are going to describe a fictional set of instructions of a fictional computer to understand what is a set of instructions and how an instruction is codified.

In our fictional computer:

- Each instruction is codified using 8 bits. The first 2 bits are used to store the operation and the next 6 bits are used to store information of the related operation.

- We have a memory with 2^6 addresses, each one with one word of 8 bits (1 byte).

- Our CPU has 3 temporally registers R1, R2 and R3. They can store 8 bit.

- In our memory, address 000000 contains the word 00001001 (9 in two's complement) and 000001 contains the word 00001000 (8 in two's complement).

In our instruction set, there are 4 instructions:

- Instruction 0: in binary 00. It indicates that we can read 2 numbers from temporally CPU registers R1 and R2 and add them. The result of the addition will be stored on R3. In this instruction only 2 first bits are used, the next 6 bits are ignored.

- Instruction 1: in binary 01. It uses a memory address. The content of that memory address is stored in R1.

- Instruction 2: in binary 10. It uses a memory address. The content of that memory address is stored in R2.

- Instruction 3: in binary 11. It shows on screen the value of the temporally register R3. In this instruction only 2 first bits are used, the next 6 bits are ignored.

For example, if we want to codify an instruction for reading the word on the memory address 000010 and store it in R2, we use the following instruction:

10 000010

Where the first 2 bits indicate the operation (10) and the next 6 bits indicate the memory address to read.

The next code will show in screen a number 17.

01 000000	Load in R1 content of address 000000.
10 000001	Load in R2 content of address 000001.
00 000000	Perform the addition R1 + R2 and saves it in R3.
11 000000	Show the result (R3) on screen.

7. CPU DESIGN: RISC AND CISC

RISC and CISC are the most popular approach to design computers. It determines how is the instruction set of the CPU.

7.1   RISC computers

Reduced **I**nstruction **S**et **C**omputer (RISC) is a computer design with those characteristics:

- Reduced set of instructions.
- Fixed size instructions.
- Only load/store instructions access to memory.

7.2   CISC computers

Complex **I**nstruction **S**et **C**omputer (RISC) is a computer design with those characteristics:

- High set of instructions.
- Variable size instructions.
- Complex instructions, doing several things in one instruction or doing complex operations.

7.3   RISC vs CISC

RISC is an improvement to CISC computers.

Usually, the main advantage of RISC CPUs is that they obtain better performance than CISC CPUs (It is because RISC CPUs are simpler than CISC CPUs and they can obtain higher clock frequency).

An advantage of CISC computer is that a program require less instructions to perform the same operation, saving space and making compilers simpler.

Modern CPUs that have a CISC design, inside have a RISC design and the most complex CISC instructions are separated in several RISC instructions.

For example, X86 CPUs are CISC CPUs in design, but internally they are RISC CPUs.

> ✐ Why X86 do not change from CISC design to a RISC design?
>
> Because if they change the instruction set, they would not be binary compatible with previous CPUs of their family.

8. ADDITIONAL MATERIAL

[1] Glossary.

[2] Exercises

9. BIBLIOGRAPHY

[1] Von Neumann architecture

https://en.wikipedia.org/wiki/Von_Neumann_architecture

[2] Harvard architecture

https://en.wikipedia.org/wiki/Harvard_architecture

[3] Modified Harvard architecture

https://en.wikipedia.org/wiki/Modified_Harvard_architecture

[4] Instruction cycle

https://en.wikipedia.org/wiki/Instruction_cycle

[5] RISC

https://en.wikipedia.org/wiki/Reduced_instruction_set_computing

[6] CISC

https://en.wikipedia.org/wiki/Complex_instruction_set_computing

[7] Computer

https://en.wikipedia.org/wiki/Computer