1. Introduction to the computer

Informática

Ingeniería en Tecnologías Industriales

Raúl Durán Díaz Juan Ignacio Pérez Sanz
Álvaro Perales Eceiza

Departamento de Automática
Escuela Politécnica Superior

Course 2024–2025

Universidad de Alcalá

Rev: 1.3

Contents

1. Basic definitions
2. Functional structure
3. Historical Evolution

Universidad de Alcalá

Rev: 1.3

They are everywhere. . .

- Computing systems are everywhere.
- A complete and unforeseen revolution in 30 years.
- Everything based on **Solid State Physics.**

Universidad de Alcalá

Rev: 1.3

Here we have the founding fathers of our technology:

Figure: Solvay Conference, Brussels, 1927 (Foto: `pastincolour.com`)

Universidad de Alcalá

Rev: 1.3

They are everywhere. . .

- Software development represents a great percentage of the GIP (Gross Domestic Product) in many countries.
- Systems price has decreased dramatically in the last 30 years.
- This has allowed the third revolution of our civilization: the creation of the Information Society

Information Society

- Main assets (i.e. software) are intangible.
  - Costly in designing and debugging.
  - Easy to move and copy, almost without cost.
- Bad news:
  - Illegal copying damages the industry severely.
  - It is necessary to establish intellectual property protection.

What is the instrument that supports all this?

Information Systems

An **Information System** takes information as input, process it, and give it back transformed according to an established plan.

It is like a factory where the raw material is *information*.

- Stores: → principal memory.
- Technical office: → Control unit.
- Production line: → Data path and functional units.

Some definitions

Definition

Computer: Machine that receives some *input data*, makes *arithmetical and logical operations* on them, and provides the results as *output data*. The whole process is set by an instruction program that is loaded previously in the memory of the same computer.

Definition

A **datum** is a set of one or more symbols that can represent some quantitative or qualitative reality (ie. a temperature, a person's name or a color)

More definitions

Definition

An **instruction** is a symbol that represents an order to the computer. Every possible order that the computer understands is codified in an instruction.

Definition

A **program** is a sequential list of instructions. The computer executes the instructions in the order that is established in the list.

Observation

Some instructions can alter the sequential order, *jumping* to a different instruction than the successive one written in the program (e.g. conditional instruction).

More definitions

Definition

Codification: is a bijective correspondence among the elements of two sets

Observation

As it is bijective (i.e. one-to-one and onto) we can identify the elements of the first set using the ones of the second set.

More definitions

Definition

In a computer the information is codified using **binary code** whose elements are **bits** (1 or 0). A **byte** is a set of 8 bits.

Universidad de Alcalá

Rev: 1.3

Units and multiples

The symbol *b* represents a 'bit' and the symbol *B* represents a 'byte'.

Prefix	Symbol	Factor
Kilo	K-	2^{10}
Mega	M-	2^{20}
Giga	G-	2^{30}
Tera	T-	2^{40}
Exa	E-	2^{50}
Peta	P-	2^{60}

- 1 KB $= 2^{10}$ bytes $=$ 1024 bytes.
- 5 Mb $= 5 * 2^{20}$ bits $=$ 1024 Kb.

Universidad de Alcalá

Rev: 1.3

Units and multiples

- **BUT** powers of 10 are also used for multiples, creating some confusion (e.g 1 Kb can also mean 10^3 bits, 1 Mb $= 10^6$ bits, 1 Gb $= 10^9$ bits ...
- **We** will use the most common actual convention of using always
  - Powers of 2 for bits and bytes (capacity) (1 Kb $= 2^{10}$ bits)
  - Powers of 10 for processor velocity (1 KHz $= 10^3$ Hz)
- See wikipedia.org/wiki/Binary_prefix for the explanation of this mess.

Universidad de Alcalá

Rev: 1.3

Before opening the lid. . .

- Input/Output devices (I/0):
  - Keyboard,
  - Mouse,
  - Screen.

What you can see. . .

Figure: Keyboard (Pic: www.codinghorror.com)

Universidad de Alcalá

Rev: 1.3

What you can see. . .

Figure: Mouse (Pic: www.germes-online.com)

Universidad de Alcalá

Rev: 1.3

What you can see...

Figure: Screen (Foto: `www.hkc-eu.com`)

Opening the lid. . .

- Motherboard, with
  - Processor,
  - Memory,
  - Connection buses:
    - System bus, EISA (Extended Industry Standard Architecture).
    - IDE (Integrated Drive Electronics) bus for discs,
    - PCI (Peripheral Component Interconnec) bus for main I/O devices
    - Other I/O buses (SCSI,. . .).

Universidad de Alcalá

Rev: 1.3

Opening the lid. . .

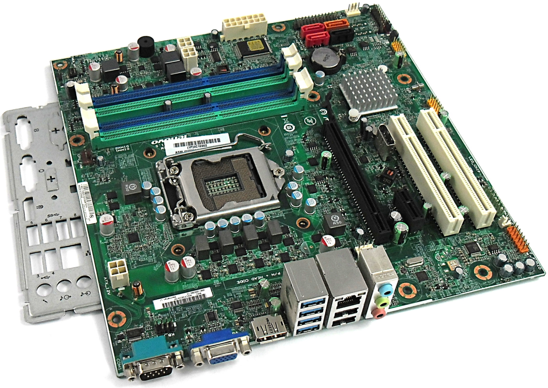

Figure: Motherboard (Pic: www.learnthat.com)

Universidad de Alcalá

Rev: 1.3

Opening the lid...

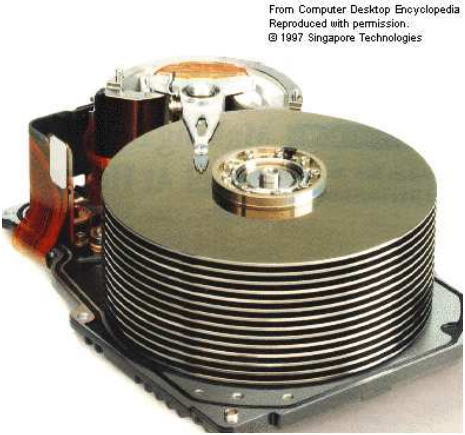

Figure: Magnetic disc (Pic: `img.zdnet.com`)

Universidad de Alcalá

Rev: 1.3

Opening the lid. . .

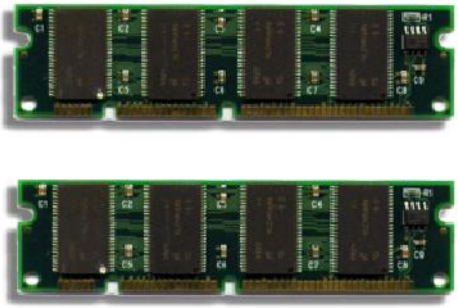

Figure: RAM memory (Pic: www.ciscomonkeys.com)

Opening the lid. . .

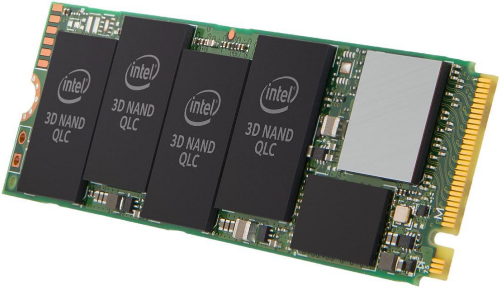

Figure: Solid state disc (Source: www.rakuten.com)

Universidad de Alcalá

Rev: 1.3

Opening the lid. . .

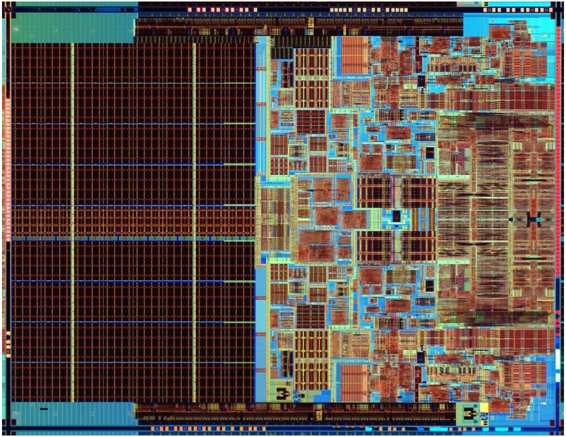

Figure: Intel Core Duo Processor (Pic: www.linuxhardware.org)

Universidad de Alcalá

Rev: 1.3

von Neumann machine structure

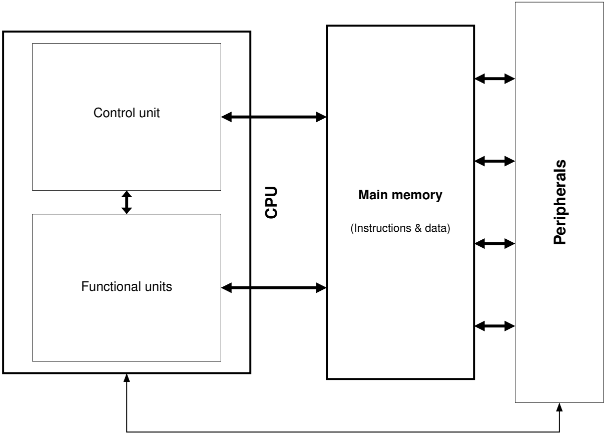

Figure: von Neumann Architecture

Memory

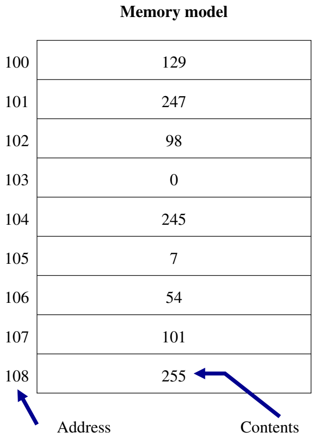

Figure: Memory: each position stores 1 byte

Universidad de Alcalá

Rev: 1.3

CPU functional structure

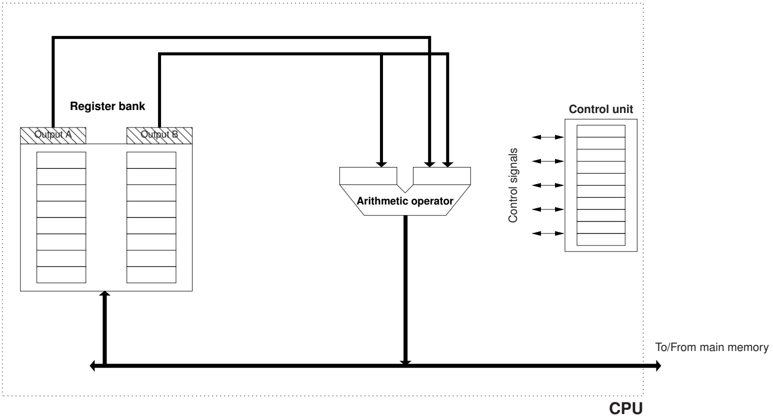

Figure: CPU internal structure

Instructions

- To work with the machine we have to speak its language:
  - The "words" of that language are *Instructions*.
  - The whole vocabulary is the ≪Instruction Set≫.
- The instructions must be:
  - As *simple* as possible, but...
  - They must allow *any* operation, i.e. the set must be *complete*.
- There are many different instruction sets (e.g. x86, MacOS), but in the end all are similar.

Universidad de Alcalá

Rev: 1.3

von Neumann stored-program concept

- von Neumann's key idea was to represent instructions with numerical codes that can be stored in memory as any other data
- The set of all numerical codes is the *Machine Language*
- Generally we don't understand those, instead we use a *mnemonic* associated to any instruction code
- The set of all *mnemonics* is the *Assembler Language*.
- Before execution, instructions and data are stored in *Registers*.

Examples of instructions and registers

Instruction	Function
ADD $R3, $R2, $R1	$R3 ← $R2 + $R1
SUB $R3, $R2, $R1	$R3 ← $R2 - $R1
ADDI $R2, $R1, N	$R2 ← $R1 + N
AND $R1, $R2, $R3	$R1 ← $R2 & $R3
OR $R1, $R2, $R3	$R1 ← $R2 | $R3

Register type	Name
General purpose	$R0, $R1, $R2, $R3,...
Program Counter	$PC
Stack Pointer	$SP

Instruction cycle

There some differences in the way instructions are executed depending on the machine, but all are based on the following cycle:

- *Fetch* the instruction contained in the memory position specified by the $PC and take it to the Control Unit.
- *Decode* the instruction and read operands.
- *Execute* operation.
- *Store* the result.

Architecture concept

Computer Architecture Definition (Instruction Set level)

Computer Architecture (Instruction Set Architecture ISA) is the specification of the **Instruction Set**, the **Registers** and some other details of their relations.

Observation

Two computers that share the same ISA can execute the same program obtaining the same results even if they are physically different (e.g. Intel and AMD).

Computer logical description

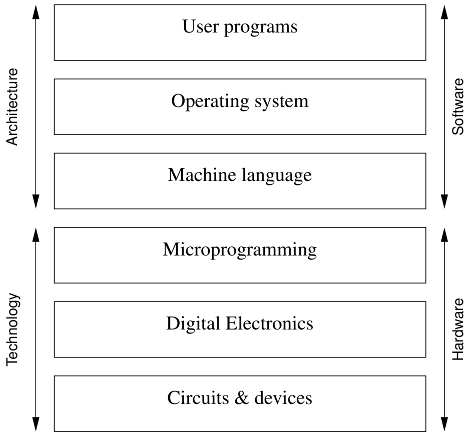

Figure: Hierarchical and logical vision of a computer

Universidad de Alcalá

Rev: 1.3

Programming Languages

- To program Assembler Language is very complicated.
- Instead normal programmers use High Level Languages (e.g. C/C++, Java, HTML...)
- They are simpler and more similar to human written language than assembler.
- A file containing a program written in high level language is called *source code*.

Compiler

- The compiler is a program that translates the source code (high level language) to assembler or machine language (instructions).

Example in C programming language

```
int i, j, f, g, h;
    f = (g + h) - (i + j);
```

After compiling to assembler

```
    ADD $R5, $R3, $R4
    ADD $R6, $R0, $R1
    SUB $R2, $R5, $R6
```

Universidad de Alcalá

Rev: 1.3

Development process

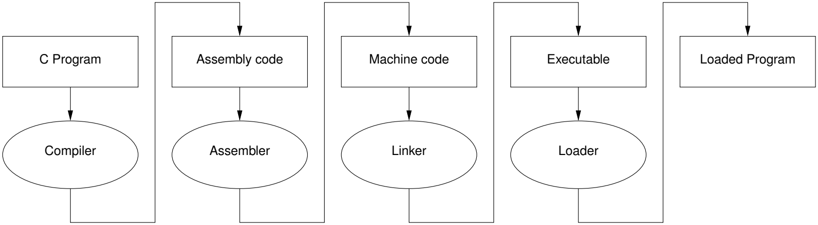

Figure: Development and execution cycle of a program written in high level language

Universidad de Alcalá

Rev: 1.3

A bit of history

- Charles Babbage (London 1791–1871): *analytical engine*. The first programmable machine, with ideas taken from a loom which could make different types of cloth using punched cards.
- Ada Lovelace (London, 1815–1852). She is recognized as the first programmer. She developed a program for the *analytical engine* that calculated the Bernoulli Numbers with an algorithm designed by herself.

II World War

- ENIAC project (Electronic Numerical Integrator And Computer), directed by J. Mauchly and J.P. Eckert, and presented in 1946
- Main characteristics:
  - 18.000 vacuum valves,
  - 25 meters long, 2.5 meters high,
  - 20 registers of 10 digits each,
  - Perform 1.900 additions per second.
  - *Wired* programmable and read data from punched cards.

Universidad de Alcalá

ENIAC Project

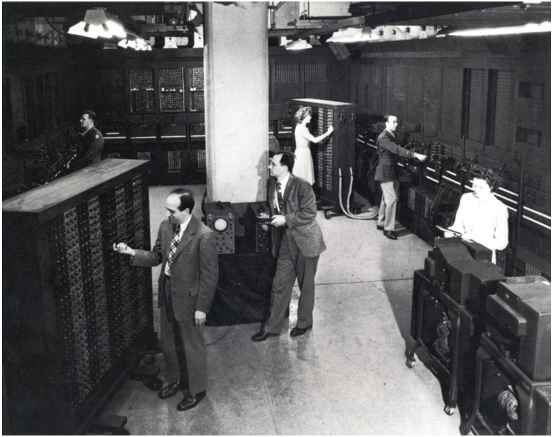

Figure: ENIAC machine (Pic: www.mrsec.wisc.edu)

ENIAC Project

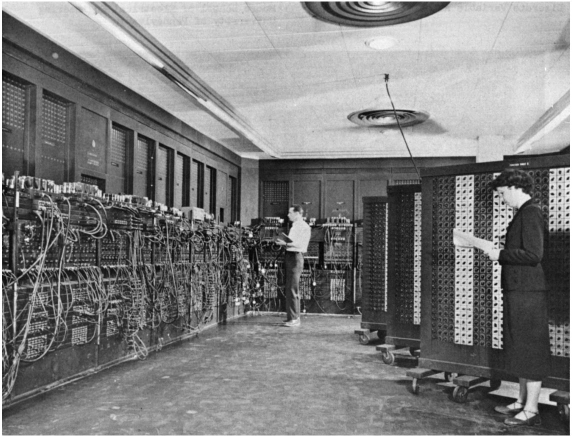

Figure: ENIAC machine (Pic: `www.mrsec.wisc.edu`)

Universidad de Alcalá

Rev: 1.3

von Neumann Machine

- In 1944, J. von Neumann[1] joined the ENIAC project and proposed to codify instructions as numbers and store them in the machine memory.
- With the help of Goldstine y Burks, they wrote an historical document [2], that is considered the foundations of modern computers.

This is the origin of the ≪von Neumann Architecture≫

[1] John von Neumann (Budapest, 1903–Washington, 1957).

[2] A.W. Burks, H.H. Goldstine, J. von Neumann, *Preliminary discussion of the logical design of an electronic computing instrument*, Report to the U.S. Army Ordnance Department, 1946.

Technology stages in the history of computing

- First stage:
  - Vacuum valves.
  - Slow speed.
- Second stage:
  - Integrated circuits in the processor.
  - Ferrite Core Memories (slower than the processor).
  - Complex instructions (to reduce its number).

Universidad de Alcalá

Rev: 1.3

Technology stages in the history of computing

- Third stage
  - Increase of integration density.
  - *Cache* memory
  - Still the complexity of instructions is a disadvantage.
- Fourth stage
  - Increase in processor speed.
  - Simpler instructions and minimum number of them.
  - Bigger caches memories that contain bot data and instructions.

Commercial Developments

- **1947**: Eckert-Mauchly Corporation. First BINAC machine. Does not succeed.
- **1951**: E-M purchased by Remington-Rand. UNIVAC I. Success: 48 machine sold at $1 million each.
- **1952**: IBM 701, first IBM computer, just 19 sold.
- **1964**: System/360: IBM defines the concept of *ISA computer architecture* developing 360 family.

Universidad de Alcalá

Rev: 1.3

Commercial Developments

- **1965**: DEC PDP-8. First commercial mini-computer. Low cost, *just* $20.000.
- **1963**: CDC 6000. First Supercomputer developed by Seymour Cray.
- **1976**: Cray still leads development of supercomputers: CRAY-1.
- **1977**: First Personal Computers (PC), Apple-II.
- **1981**: IBM Personal Computer (Intel and Microsoft).
- **2000's**: Computers in many personal electronic gadgets (Ipod, tablets, mobiles...)
- **2020's**: Quantum Computing?