

Unit 1

Languages and Basic Notions

Alberto Casagrande

Email: acasagrande@units.it

A.Y. 2022/2023



Guess what it is...



Guess what it is...



Guess what it is...

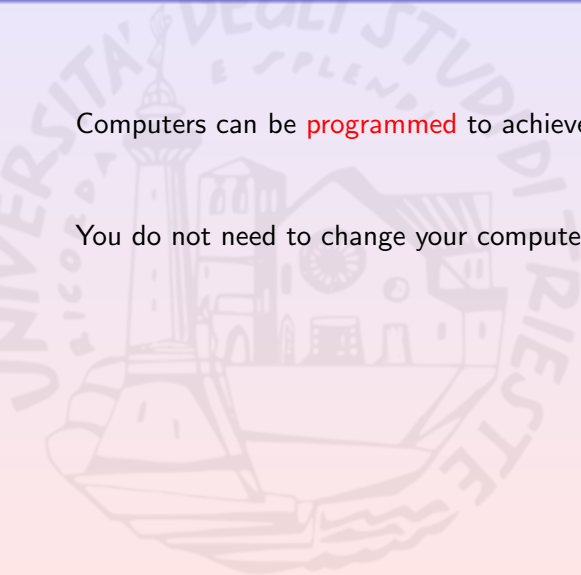


What is the main difference?

Programmability

Computers can be **programmed** to achieve new tasks.

You do not need to change your computer for any new task . . .



Programmability

Computers can be **programmed** to achieve new tasks.

You do not need to change your computer for any new task . . .

just install a new program

Programmability

Computers can be **programmed** to achieve new tasks.

You do not need to change your computer for any new task . . .

just install a new program

Please, do **NOT** dry any cat in an oven!

What is a program?

As far as we are concerned

it is a finite sequence of instructions that
(hopefully) leads computers to perform an aimed
task in a finite amount of time

What is a program?

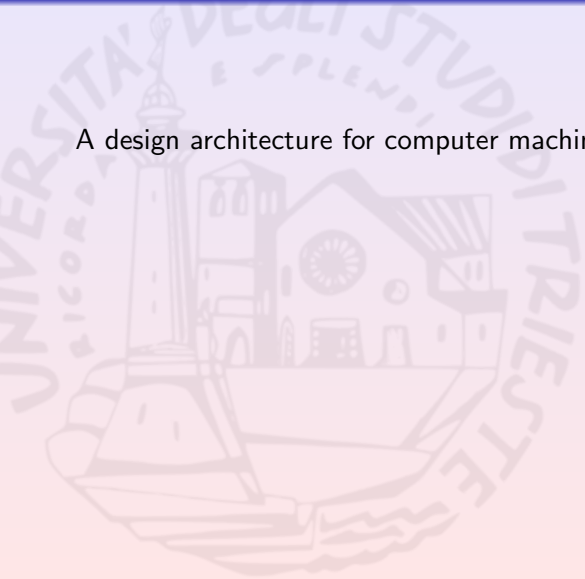
As far as we are concerned

it is a finite sequence of instructions that
(hopefully) leads computers to perform an aimed
task in a finite amount of time

So, what is a computer and what is an instruction?

von Neumann Architecture

A design architecture for computer machinery



von Neumann Architecture

A design architecture for computer machinery

More or less, it consists in:

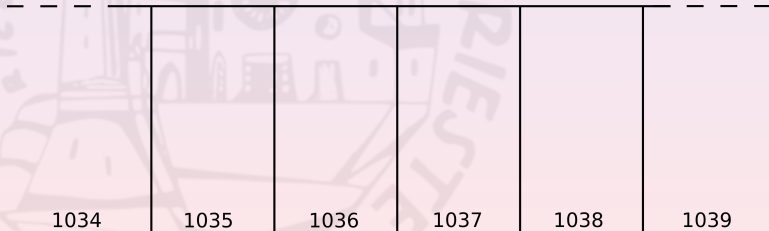
- a **central processing unit (CPU)**
- a **memory** storing both data and instructions
- I/O mechanisms

Nowadays, virtually every electronic device implements it

Memory

It is indexed by **addresses**

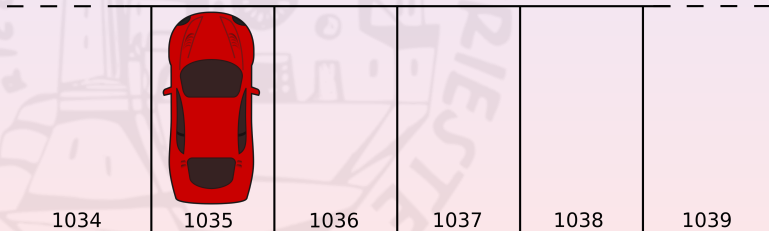
Just like ... parking stalls



Memory

It is indexed by **addresses**

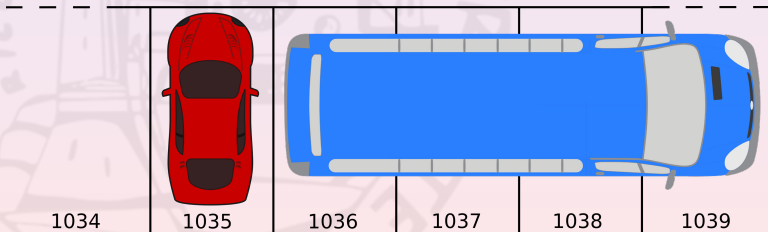
Just like ... parking stalls



Memory

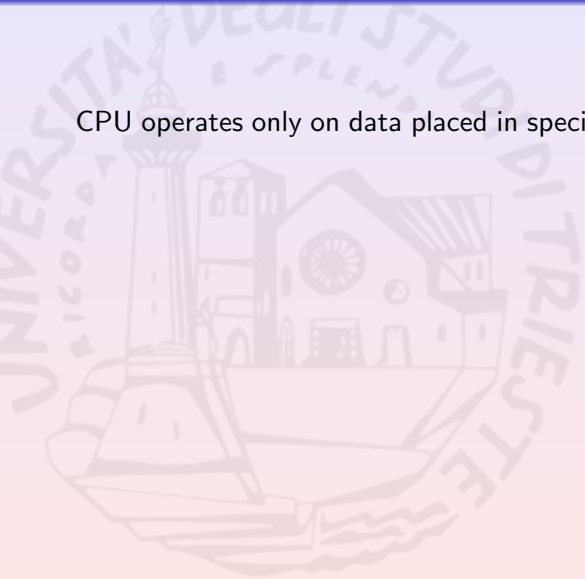
It is indexed by **addresses**

Just like ... parking stalls



Registers

CPU operates only on data placed in special locations: the **registers**



Registers

CPU operates only on data placed in special locations: the **registers**

E.g., To sum two numbers:

- load 1st number into R1
- load 2nd number into R2
- evaluate the sum and store it in R3
- save the result from R3 into the memory

Their names, number and usage depend on the CPU type.

How to represent data?

Digital computers store data as sequences of **Binary digiT**s (bits)

E.g.,

'a'
-3
13
27

How to represent data?

Digital computers store data as sequences of **Binary digiT**s (bits)

E.g.,

'a'	→	00011011
-3	→	11111101
13	→	00001101
27	→	00011011

How to represent data?

Digital computers store data as sequences of **Binary digiT**s (bits)

E.g.,

'a'	→	00011011
-3	→	11111101
13	→	00001101
27	→	00011011

In 2021 the length of PC's registers is usually 64 bits (**64-bit architecture**).

How to represent instructions?

As bit sequences too

E.g., on Intel 4004 (the first microprocessor):

Store 10 and 2 in R1 and R0

Load R0 in ACC

Sum R1 to ACC

Save ACC in R1

How to represent instructions?

As bit sequences too

E.g., on Intel 4004 (the first microprocessor):

Store 10 and 2 in R1 and R0	→	0010 0000 1010 0010
Load R0 in ACC	→	1010 0000
Sum R1 to ACC	→	1000 0001
Save ACC in R1	→	1011 0001

Let's see whether you understood

Question

What bit sequence does assign 13 and 7 to R1 and R0?

Let's see whether you understood

Question

What bit sequence does assign 13 and 7 to R1 and R0?

Writing programs as bit sequences is:

- extremely difficult
- error prone
- CPU dependent

Programming Languages

Programming languages describe programs by using a human viable syntax.

Syntax is simple (few rules) and semantics is not ambiguous (no sentences like *"I saw a man on a horse using a telescope"*).

We will focus on the **C** programming language.

Compilers



```
image timer;  
thread timer;  
public void init() {  
    lastcount = 10; count = 0;  
    pictures = new Image[10];  
    mediaTracker tracker = new Media  
    for (int a = 0; a < lastcount;  
        pictures[a] = getImage(  
            tracker.addImage(pictures  
                er.checkAll(true);  
    }  
    public void start() {  
        if (timer == null) {  
            timer = new Thread(  
                timer.start();  
        }  
    }  
    public void paint(Graphics  
        g.drawImage(picture  
        if (count == last  
    }  
    id run() {
```

Source code (PL
sentences)



Bit sequences
(Machine code)

Compilers

```
image timer;  
thread timer;  
public void init() {  
    lastcount = 10; count = 0;  
    pictures = new Image[10];  
    mediatracker tracker = new Media  
    for (int a = 0; a < lastcount;  
        pictures[a] = getImage (a,  
            tracker.addimage(pictures  
                tracker.checkAll(true);  
    }  
    }  
public void start() {  
    if (timer == null) {  
        timer = new Thread(  
            timer.start();  
    }  
    }  
}   
public void paint(Graphics  
    g.drawImage(pictures  
        if (count == last  
    }  
}   
id run() {
```

Source code (PL
sentences)



Bit sequences
(Machine code)

Compilers “translate” source code into binary executables

Different architectures



Every architecture represent data in “its own way”, ...

Different architectures

... every binary code can be executed exclusively on the architecture for which was compiled

Never mind: new architecture \Rightarrow new compilation

Interpreted Languages

Some languages avoid compilation (e.g., Python)

Interpreters = programs that interpret other program on request

Pros

- No compilation
- Same executables for every architecture
- Executables are human readable

Cons

- Interpreter needed
- Execution time
- Resources eager

Developing C Programs

We will need:

- a computer :-P
- a text editor
- a C compiler
- ...

Developing C Programs

We will need:

- a computer :-P
- a text editor
- a C compiler
- ...

Actually, for the programming alone, we can use repl.it, onlinegdb.com, programiz.com, ...

Developing C Programs

We will need:

- a computer :-P
- a text editor
- a C compiler
- ...

Actually, for the programming alone, we can use repl.it, onlinegdb.com, programiz.com, ...

... but we are not *exclusively* interested in programming

Coming next...

- User/HW abstraction layers
- operating systems
- what is POSIX?
- Ubuntu Linux
- working with Command-line User Interface (CLI)