Unit 1

Languages and Basic Notions

Alberto Casagrande
Email: acasagrande@units.it

a.a. 2019/2020

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 sequence of instructions that (hopefully) leads computers to perform an aimed task

What is a program?

As far as we are concerned:

It is a sequence of instructions that (hopefully) leads computers to perform an aimed task

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:

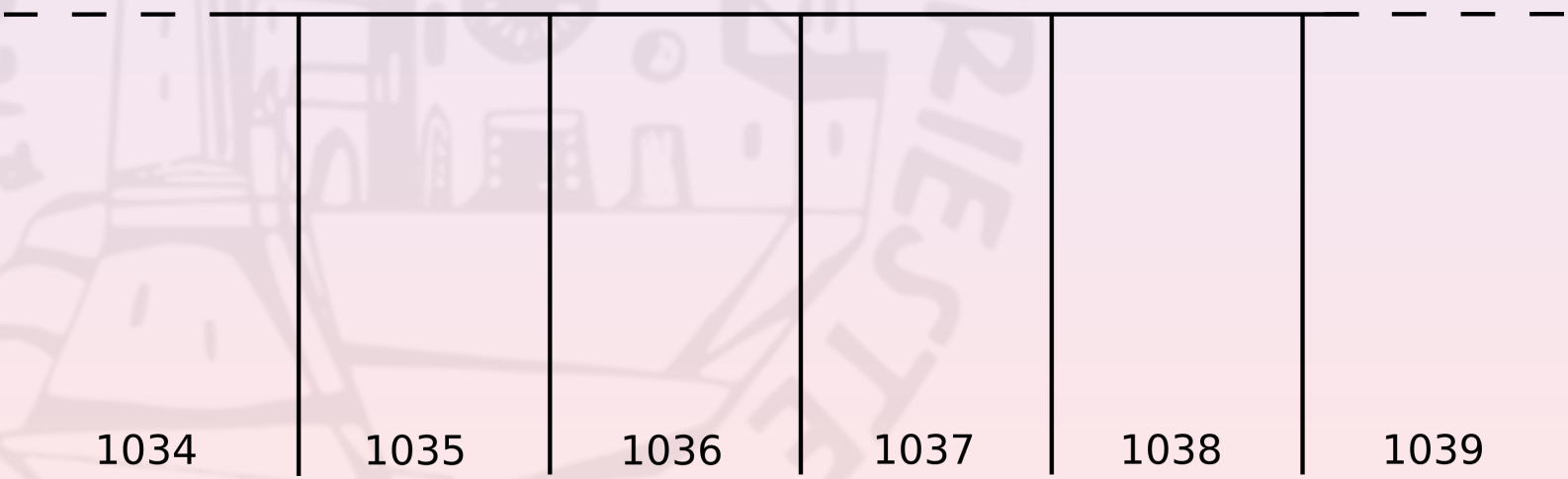
- a central processing unit (CPU)
- a memory storing both data and instructions
- I/O mechanisms

Virtually every electronic computer implements it

Memory

It is indexed by addresses

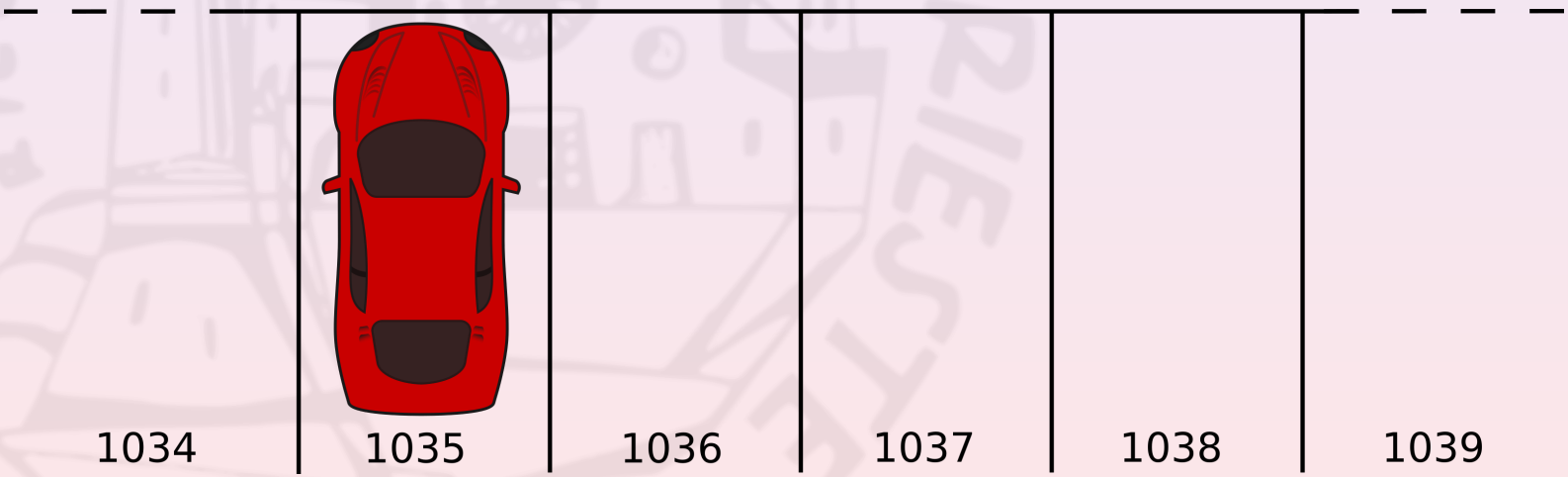
Just like . . . parking stalls



Memory

It is indexed by addresses

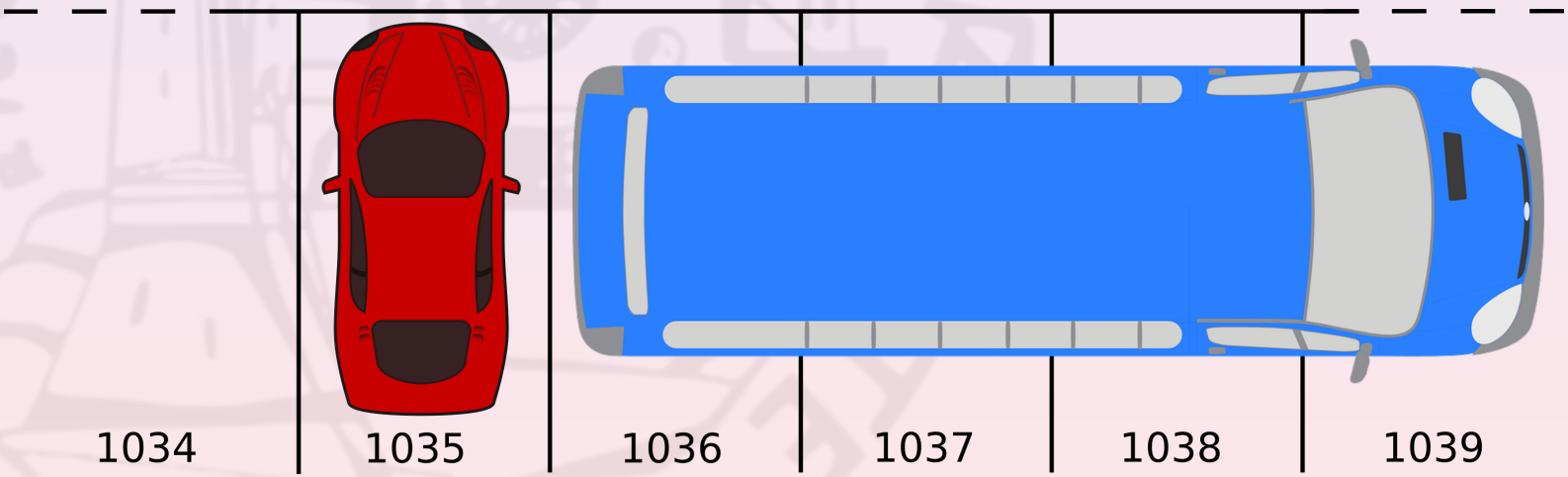
Just like . . . parking stalls



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
- compute the sum
- 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 digiTs (bits)

E.g.,

'a'
-3
13
27

How to represent data?

Digital computers store data as sequences of BInary digiTs (bits)

E.g.,

'a'	$\rightarrow$	00011011
-3	$\rightarrow$	11111101
13	$\rightarrow$	00001101
27	$\rightarrow$	00011011

How to represent data?

Digital computers store data as sequences of BInary digiTs (bits)

E.g.,

'a'	$\rightarrow$	00011011
-3	$\rightarrow$	11111101
13	$\rightarrow$	00001101
27	$\rightarrow$	00011011

In 2019 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:

Store 10 and 2 on R0 and R1
Load R0 on ACC
Sum R1 to ACC
Save ACC in R1

How to represent instructions?

As bit sequences too

E.g., on Intel 4004:

Store 10 and 2 on R0 and R1	$\rightarrow$	0010 0000 1010 0010
Load R0 on ACC	$\rightarrow$	1010 0000
Sum R1 to ACC	$\rightarrow$	1000 0001
Save ACC in R1	$\rightarrow$	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

Source code
(Sentece of the PL)

Bit sequences
(Machine code)

Compilers

Source code
(Sentece of the PL)

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

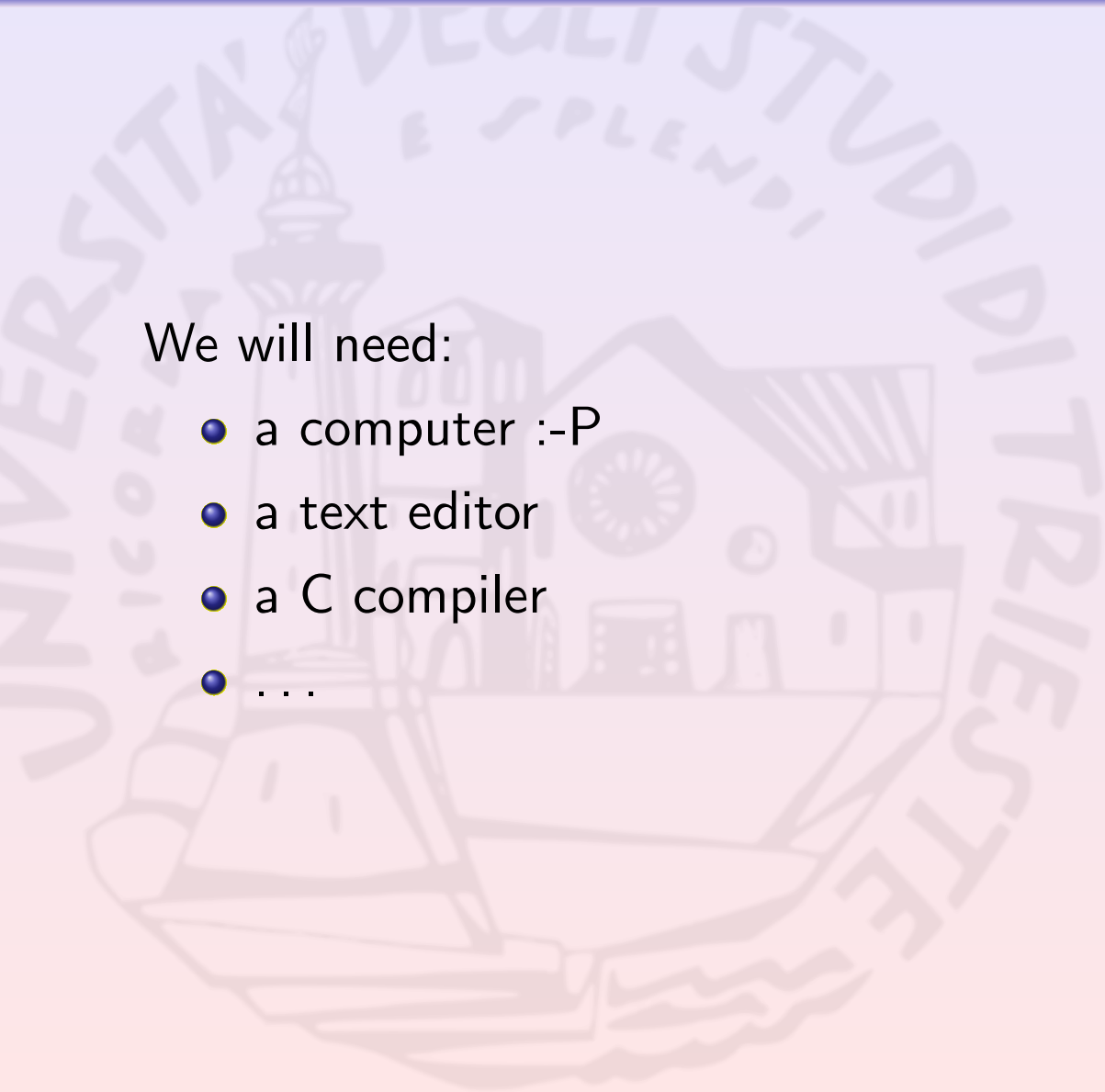
Pros

- 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
- ...

Coming soon...

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