032CM - 2026

PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

UNIX and Linux shell

Gianluca Levi

gianluca.levi@units.it, giale@hi.is

Office: Building C11, 3rd floor, Room 329

Fall 2026

What is a computer?

A computer is a machine that executes sequences of instructions (algorithms)

What is a computer?

A computer is a machine that executes sequences of instructions (algorithms)

- **Store instructions and information** as data
- Execute instructions at **very high speed** (billions per second)

What is a computer?

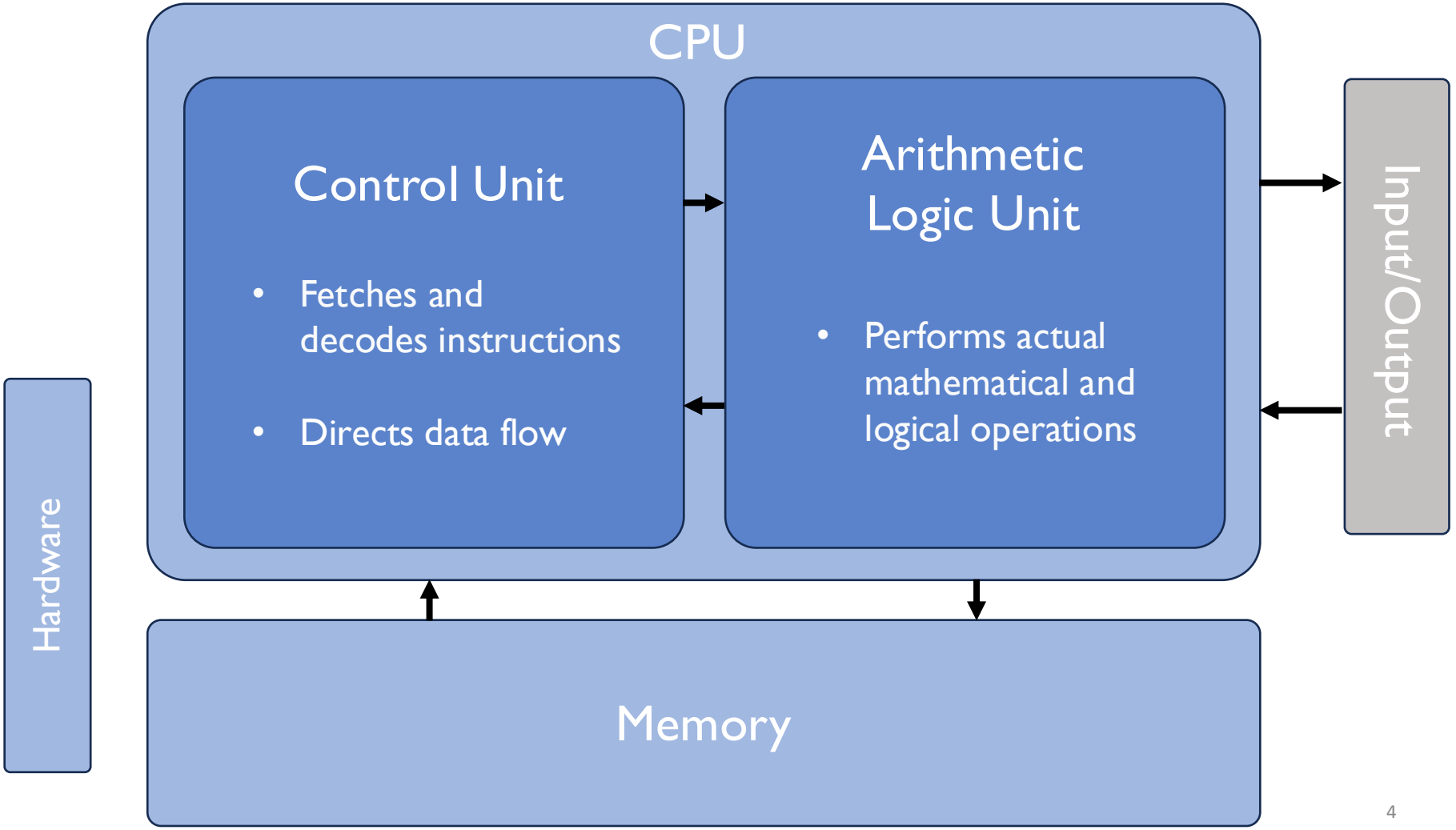
A computer is a machine that executes sequences of instructions (algorithms)

- **Store instructions and information** as data
- Execute instructions at **very high speed** (billions per second)

Programs tell the computer what sequence of instructions is required

- Written in **high-level language** (Fortran, C, Python, etc.)
- Translated into machine language (binary instructions) by a **compiler**/**interpreter**

The computer only does what you tell it to do!

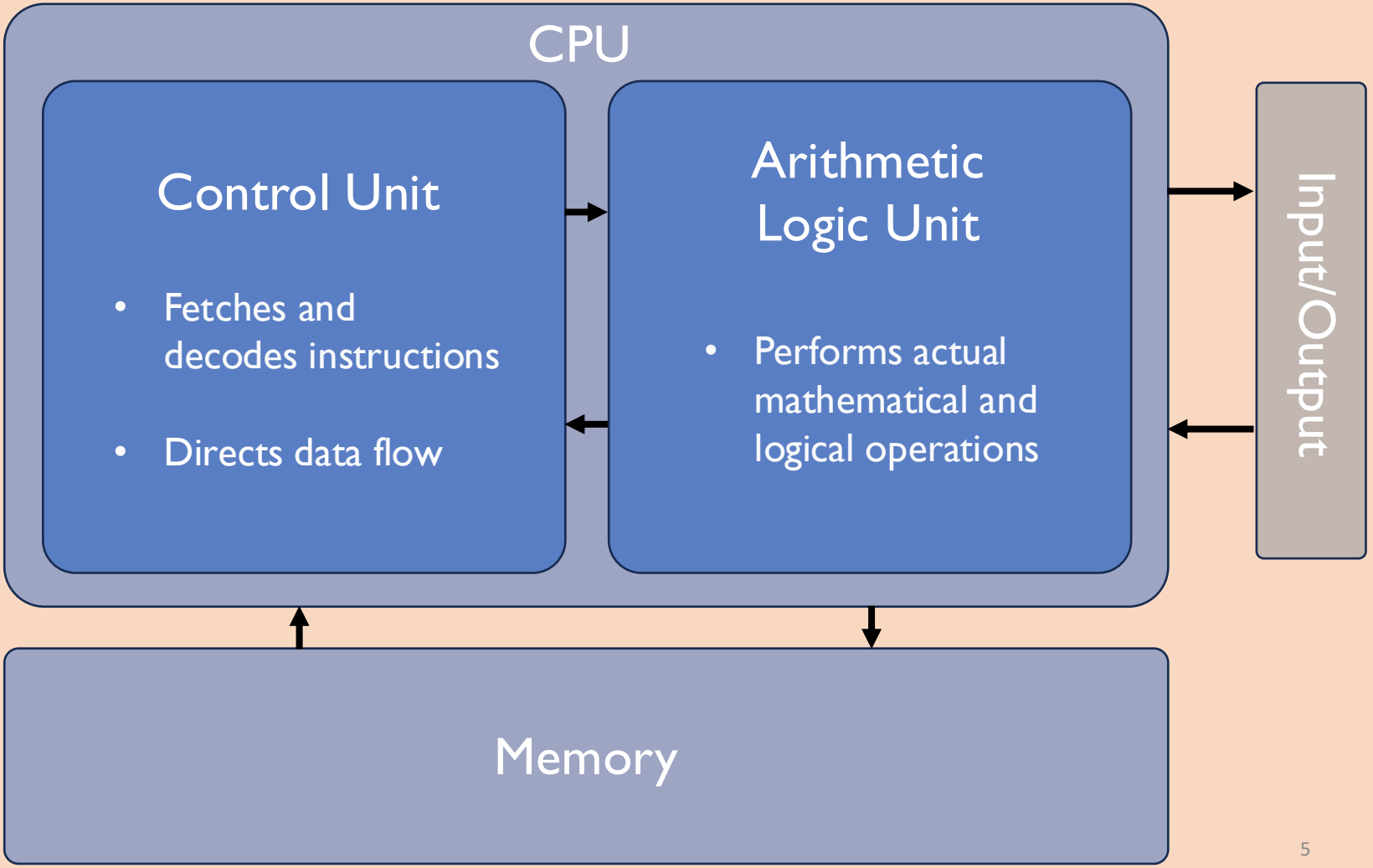

CPU
Control Unit
• Fetches and decodes instructions
• Directs data flow
Arithmetic Logic Unit
• Performs actual mathematical and logical operations
Input/Output
Hardware
Memory

Operating System (OS)

System software that manages all hardware resources and provides interface for programs to run (**Resource manager**)

Software

Hardware



UNIX

- First portable operating system (born in the 1970s)

UNIX and Linux

UNIX

- First portable operating system (born in the 1970s)

Linux

- **Free**, **open-source** UNIX-like system (1991)
- Stable, secure, flexible, with wide hardware/software support
- Everyone can download, use, modify, and distribute
- Widely used
  - Most supercomputers for **High-Performance Computing** (HPC)
  - Most **computational chemistry software** run on Linux

UNIX is an essential environment for scientific computing

Kernel and Shell

Kernel

- Core of the operating system
- **Controls all hardware** (CPU, memory, devices)
- Manages **interactions between hardware and software** components

Kernel and Shell

Kernel

- Core of the operating system
- **Controls all hardware** (CPU, memory, devices)
- Manages **interactions between hardware and software** components

Shell

- **Interface** between user and kernel
- Takes commands from the user and passes them to the kernel for execution
- **Bash** is one of the most popular shells (default on most Linux distributions and macOS)

The shell is the primary way we interact with the operating system

Everything treated as a file

Ordinary files

- Contain text, data, or programs
- Filenames can contain any character except /

> **Note!**
> Do not use *, ?, # and &, they have special meanings
> Do not leave spaces in filenames, rather use underscore _

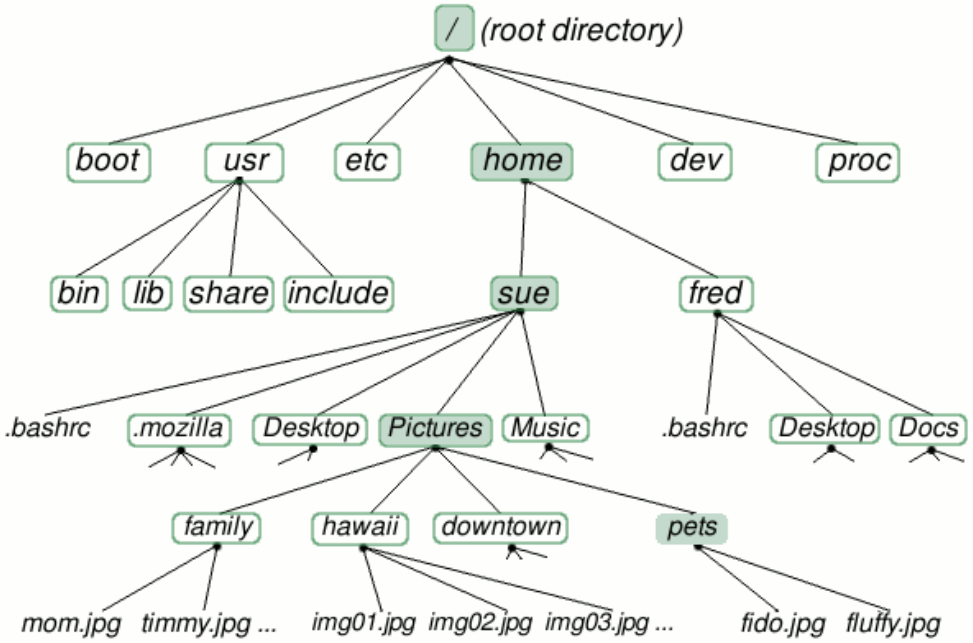
Directories

- Containers for other files (including other directories)

UNIX file system

Hierarchical tree structure with root (/) as top directory

- **Absolute path** starts from
  - / ⟶ Root, e.g. /home/sue/Pictures/pets
  - ~ ⟶ Home directory, e.g. ~/sue/Pictures/pets
- **Relative path** starts from
  - . ⟶ Current directory
  - .. ⟶ Parent directory



Login to Linux machine

1. Open **Bitvise SSH client**
2. Enter

   Host: dscfalpha7.units.it
   Username: pcc
3. Log in

To transfer files use **Bitvise SFTP**

Try it!

Type uname –s and press Enter. What do you see on the terminal?

Note!

We all share the same home. Let's keep it tidy. Create a personal folder with your name. From now on, **place all your files inside your personal folder only**.

Executing Linux commands

General structure of a Linux command:
>> command [options] [arguments]

Some **common Linux commands**:

pwd ⟶ display the current working directory

cd ⟶ change directory

ls ⟶ list of files and directories

mkdir ⟶ create directories

cp (-r) ⟶ copy files and directories

rm (-r) ⟶ delete files and directories

mv ⟶ rename or move file and directories

history ⟶ chronology of commands

exit ⟶ close the session and log out

man ⟶ manual and info of commands

clear ⟶ Clears the terminal screen

Editing and working with files

Create and edit text files with vi (vim)

vim filename → open and edit files in the Vim editor

- i → insert mode
- :wq → save & quit
- :q! → quit without saving

vimdiff filename1 filename2 → compares files line by line

Try it!

Type vimtutor and press Enter.

Quickly examine files without opening an editor

cat filename → display contents of a file

head filename → show the first 10 lines

tail filename → show the last 10 lines (tail -f follows updates)

grep "text" filename → print every line in filename that contains text

Special characters and shortcuts

< ⟶ redirect input

> ⟶ redirect output

* ⟶ wildcard: matches any number of characters

? ⟶ wildcard: matches a single character

Tab ⟶ command or filename completion

Ctrl+C ⟶ kill a running process

Ctrl+Z ⟶ stop a running process

Ctrl+A ⟶ move cursor to the beginning of the command line

Ctrl+E ⟶ move cursor to the end of the command line

Assignment 1

Problem 1

Two XYZ files are provided in ~/gianluca/assignment1/:

- `molecule1.xyz`
- `molecule2.xyz`

Note: The first line of an XYZ file contains the number of atoms, the second line is a comment, and subsequent lines contain the Cartesian positions of the atoms, one atom per line: `Element x y z`.

(a) In your personal course directory on the Linux machine, create a subdirectory `assignment1`. Copy the two files from ~/gianluca/assignment1/ into your `assignment1` folder.

(b) Extract the number of atoms for each XYZ file without opening the files with a text editor. Report the number of atoms in a text file (e.g. `assignment1.txt`) with two lines (for this task you can use the text editor `vim`):

```
molecule1.xyz: <N1> atoms
molecule2.xyz: <N2> atoms
```

(c) Extract only the nitrogen atoms (lines starting with "N") from molecule1 and save them to a file.

(d) Use `vimdiff` to compare `molecule1.xyz` and `molecule2.xyz`. Briefly describe the differences you observe (e.g., atom counts, elements, coordinates).