# PMC Lecture 04

Gianmaria Romano

7 October 2025

# Chapter 1

# Parallel computing

## 1.1 Writing parallel programs

Generally speaking, it is possible to write parallel programs by using some dedicated APIs provided by the C programming language.

| | Shared Memory | Distributed Memory |
|---|---|---|
| SIMD | CUDA | |
| MIMD | Pthreads/ OpenMP/ CUDA | MPI |

### 1.1.1 Implementing memory structures

Regarding memory structures, a parallel system can implement a shared memory structure, where cores can access a common memory space, or a distributed memory structure, which provides a private memory for each core and thus coordinates activity by message passing over a dedicated network.

### 1.1.2 Implementing control units

Regarding control units, a parallel system can opt for a multiple-instruction multiple-data approach, where each core has its own control unit and can work independently of the other cores, or a single-instruction multiple-data approach, where cores share the same control unit and can either execute the same instructions on different data or stay idle.

## 1.2 Classifying multicore systems

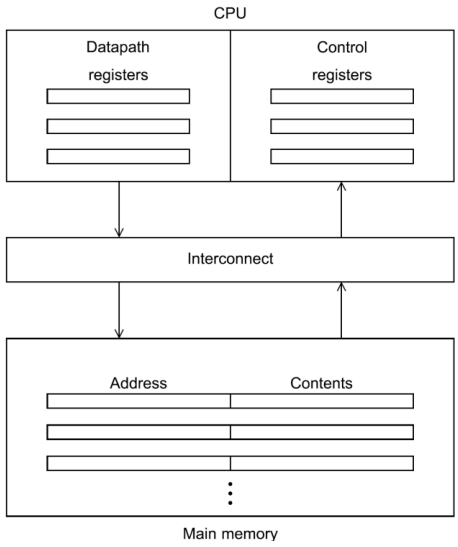
Generally speaking, multicore systems tend to experience performance benefits compared to single-core system.
In particular, it is possible to classify a multicore system in one of the following groups:

- **Concurrent systems:** the system allows multiple tasks to run at any time, although said tasks can still be independent.
- **Parallel systems:** the tasks cooperate closely, typically by means of a shared memory or a communication network, in order to solve a common problem.
- **Distributed systems:** the tasks can cooperate to solve a common problem, but, compared to parallel systems, they work together less frequently.

## 1.3 The von Neumann architecture

When implementing a (parallel) system, it can come in handy to know its high-level architecture in order to simplify the optimization procedure.
Nowadays, most systems are based on the von Neumann architecture, which contains the following components:

- **Main memory:** this structure collects data and instructions, which are accessed by referring to the address of their location within the memory space.
- **Processor:** this component acts as a control unit as it executes a program with the help of registers, which typically store variables or information about the status of the program, and a program counter, which contains the memory address of the next instruction to execute.
- **Interconnect:** this component, typically implemented as a bus, allows to connect the main memory with the processor, meaning that it will also need to deal with the "von Neumann bottleneck", which represents a limitation concerning the communication capability of the system.


CPU
Datapath
registers
Control
registers
Interconnect
Address
Contents
Main memory