6CS005 High Performance Computing

Lecture 3

Parallel Computing

Contents

- Parallel Computing Overview
- Key Components of Parallel Computing
- Serial and Parallel Computing
- Sequential and Parallel Programming
- Relationship Between Tasks
- Classification of Computing Systems: Flynn’s Classification
- Enhancing Computational Efficiency: Key Objectives
- Classification of Computer Architecture by Memory Organization
- Homogeneous Computing
- Homogeneous Architecture

Parallel Computing Overview

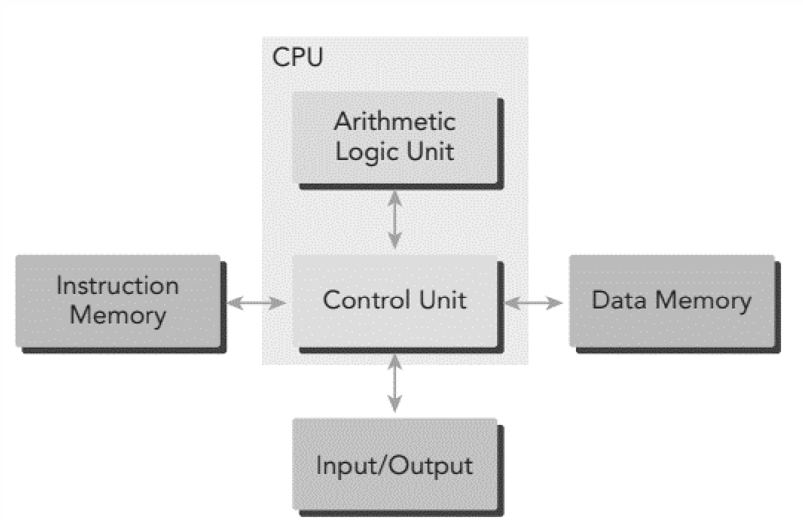
- Primary goal of parallel computing is to improve the speed of computation by performing many **calculations simultaneously**
- *Definition:*
  - **Calculation perspective**: Large problems are divided into smaller tasks, solved concurrently
  - **Programmer perspective**: Concurrent tasks are mapped onto multiple computing resources (cores or computers) for parallel execution
- Parallel computing involves the close integration of **hardware** and **software** to achieve efficient performance in solving complex problems

Key Components of Parallel Computing

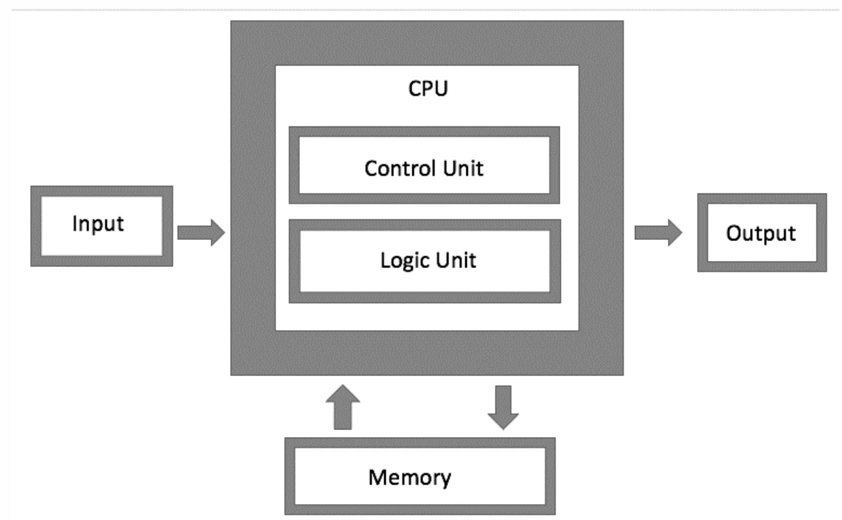
- Hardware (Computer Architecture)
  - Focuses on supporting parallelism at the architectural level
- Software (Parallel Programing)
  - Focuses on solving a problem concurrently by fully using the computational power of the computer architecture
- Note: *To enable parallel execution in software, the hardware must support concurrent execution of multiple processes or threads.*

Hardware (Computer Architecture)

- Most modern processors utilize the *Harvard architecture* instead of the *Von Neumann architecture*



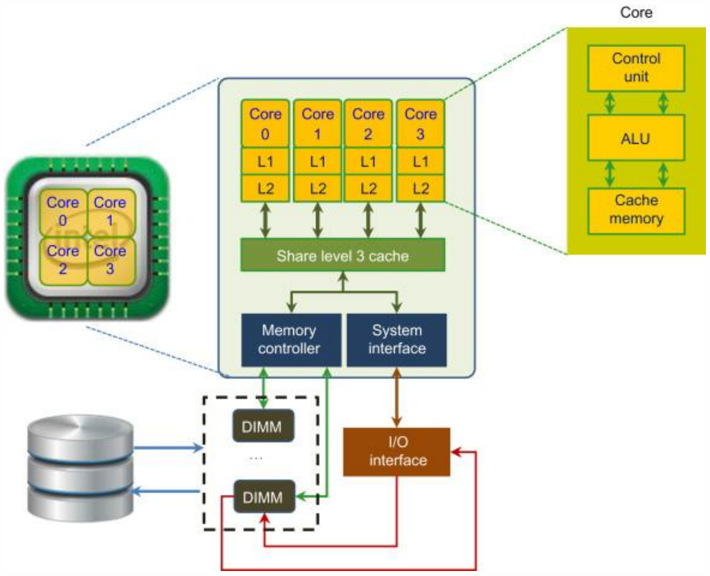
Harvard Architecture



Von Neuman Architecture

Contd…

- CPU(Core):
  - The primary component for processing tasks
  - Early computers operated with a single core on a chip, known as **uniprocessor**
  - Modern chip design integrates multiple cores(individual processing unit within a processor) into a single processor, referred to as **multicore architecture**.
  - This design supports parallelism, allowing multiple tasks to be processed simultaneously
  - Example Quad-core processor(Intel Core i7-7700 or AMD Ryzen 7, has four cores (Core 0, Core 1, Core 2, Core 3) that can run separate thread simultaneously



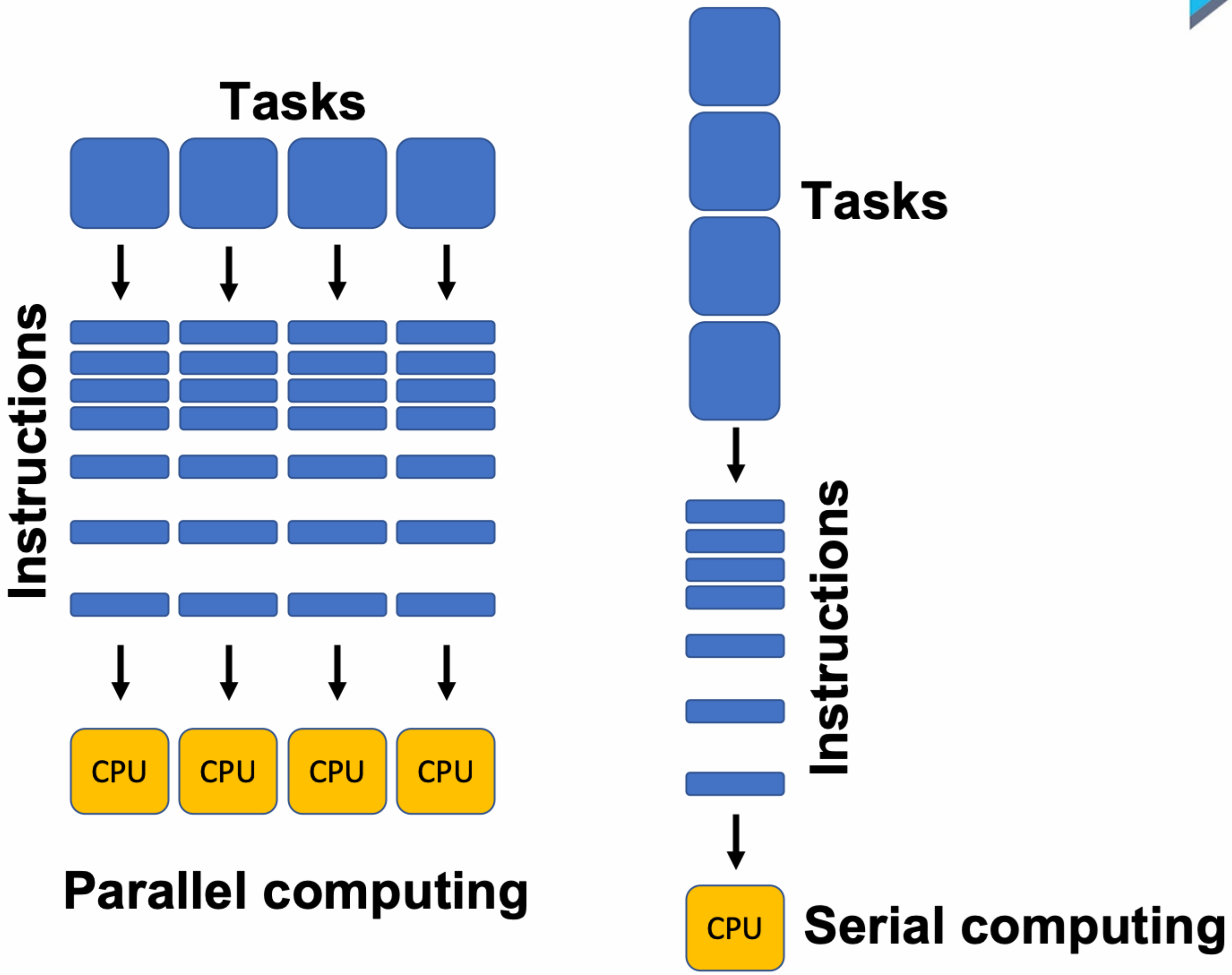
Core Processor Architecture

DIMM: *Dual Inline Memory Module are essential components in core processor architecture, providing the necessary memory resources for efficient processing, types or volatile memory*
Level 3 Cache: *Type of cache memory used in computer architecture to improve the performance of the CPU by storing frequently accessed data and instructions*

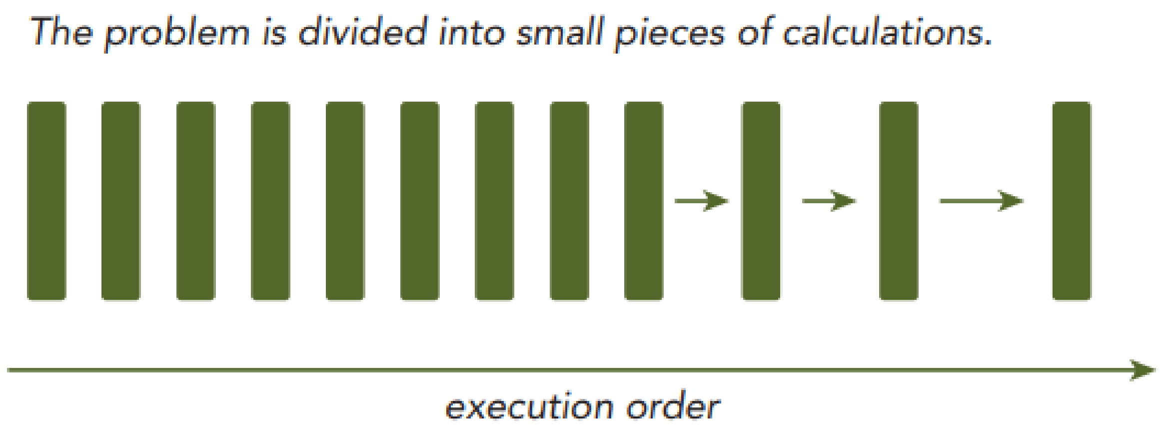
Can a quad-core processor handle four tasks at the same time?

- The ability of a quad-core processor to handle four tasks simultaneously is not automatic; it depends on several factors:
- **Multithreading**:
  - Single-Threaded Applications:
    - If an application is single-threaded, it can only use one core at a time, even on a quad-core processor, activating only one core for that task.
  - Multi-Threaded Applications:
    - Multi-threaded applications can utilize multiple cores by splitting workloads into separate threads, enabling concurrent execution. A well-optimized multi-threaded application can effectively use all four cores of a quad-core processor.
- **Operating System and Task Management**
  - The operating system manages tasks and allocates them to available cores, distributing processes or threads based on their availability and workload.

Serial and Parallel Computing



- **Sequential Programming**
  - Involves writing code where tasks are executed one after the other
  - Each calculation or instruction waits for the previous one to complete



Relationship Between Tasks

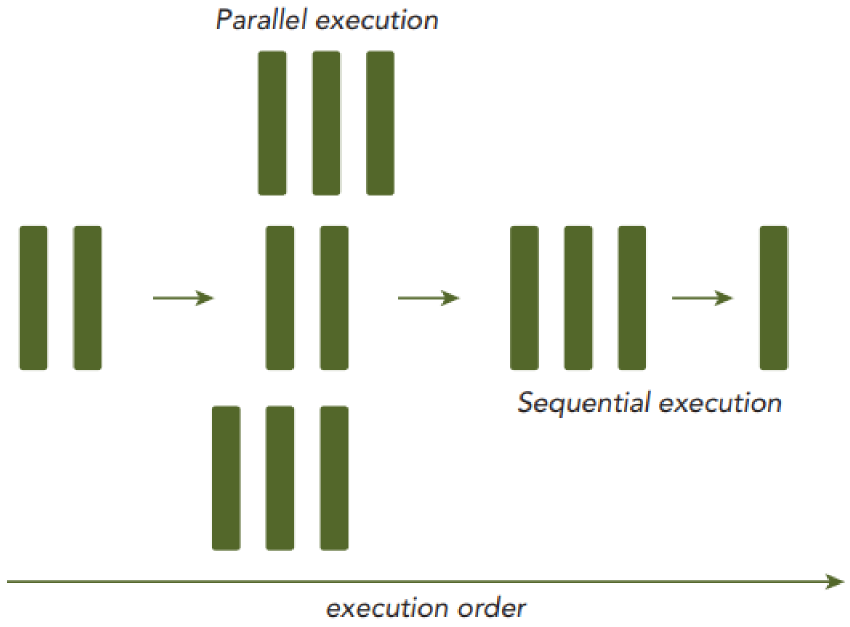
- Based on execution constraints:
- **Sequential/Dependent Tasks:**
  - Dependent tasks are tasks that have a direct relationship where the outcome of one task is necessary for the next task to execute
- **Concurrent/Independent Tasks:**
  - Independent tasks are tasks that can operate on their own without needing to wait for the results of other tasks
- Understanding data dependencies is crucial for implementing parallel algorithms, as they are major barriers to achieving parallelism
- Often, **multiple independent chains of dependent tasks** provide the best opportunities for effective parallelization.

Contd…

- Multiple independent chains of dependent tasks
- Example:
- **Chain of dependent task:**
  - Tas1□ Task2□ Task3
  - In this chain, Task2 can only start once Tasks is finished, and Task3 can only start after Task2 is completed
- **Multiple independent chains:**
  - Chain 1: TaskA1□ TaskA2□ TaskA3
  - Chain 2: TaskB1□ TaskB2□ TaskB3
  - Here, Chain 1 and Chain 2 can run concurrently because the do not share any dependencies

Contd…

- **Parallel Programming**
  - Involves writing code that enables multiple tasks to be executed concurrently, often by splitting a problem into smaller tasks
  - Takes advantage of multi-core or distributed systems to improve performance



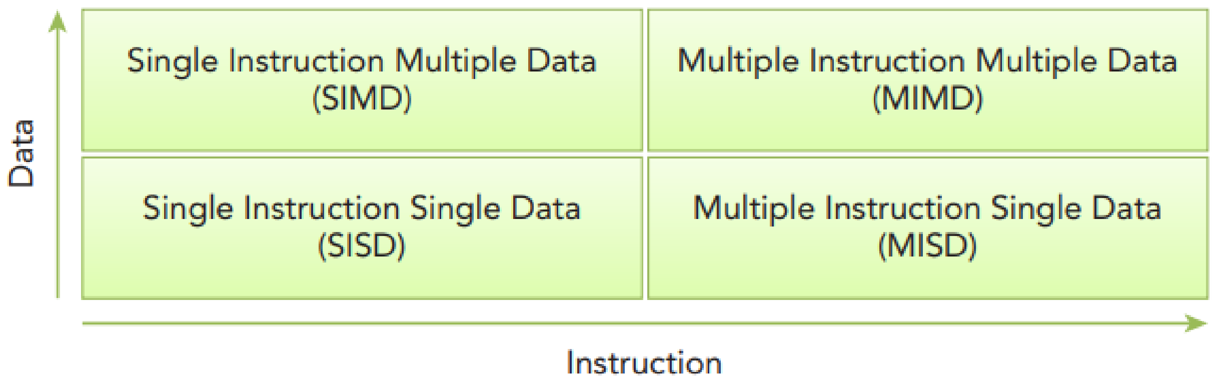
Parallelism Overview

- Parallelism is essential in modern computing
- It improves performance by allowing multiple tasks to run at the same time
- Types or Parallelism:
  - **Task Parallelism**
    - Multiple independent tasks or function running at once
    - Tasks or function are distributed across different CPU cores
    - Eg: Database Server(*Multiple user can query and update data simultaneously*), Web Browsers(*Manage different browser tabs*), Operating Systems(*Run multiple programs at the same time*)
  - **Data Parallelism**
    - Many pieces of data processed simultaneously
    - Data is divided among multiple cores for faster processing
    - Eg: Image Processing: Each pixel can be processed independently with the same filter operation such as blur, sharpen, or edge detection

Classification of Computing System: Flynn's Taxonomy

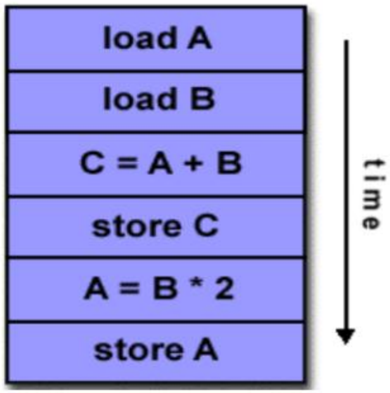
- Computing systems can be classified into four major categories based on the **number of instruction** and **data streams** they can process simultaneously
- *Instruction Stream*: A sequence of instructions executed by a processor
- *Data Stream*: A sequence of data required by an instruction stream

  - Single Instruction Single Data (SISD)
  - Single Instruction Multiple Data (SIMD)
  - Multiple Instruction Single Data (MISD)
  - Multiple Instruction Multiple Data (MIMD)

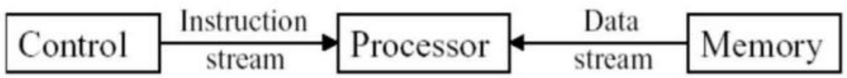


Contd…

- SISD
  - Traditional serial architecture with a single core
  - At any time, only one instruction stream is executed, and operations are performed on one data stream
  - Examples: most PCs, single CPU workstations and mainframes

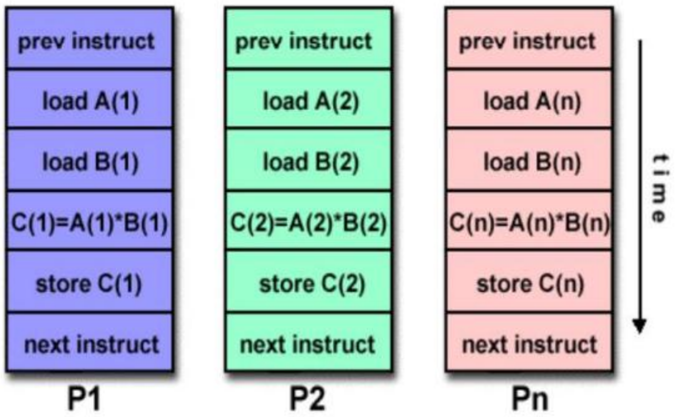


The SISD organization

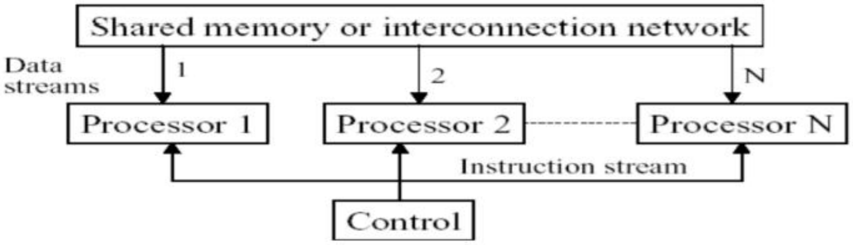


Contd…

- SIMD
  - A type of parallel computer
  - Best suited for specialized problems such as image processing and vector computation

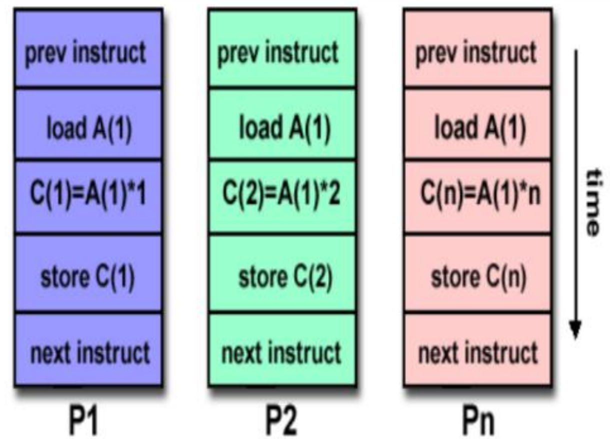


The SIMD organization

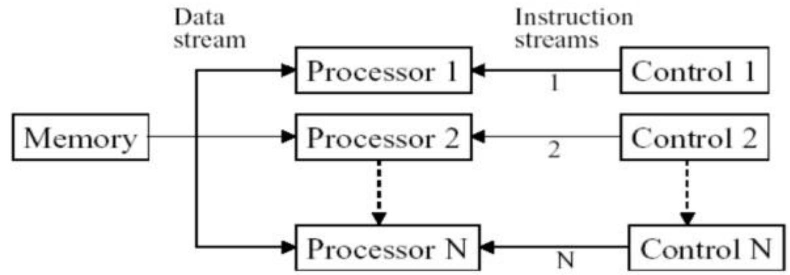


Contd…

- MISD
  - Uncommon architecture
  - Each core operates on the same data stream but uses different instruction stream

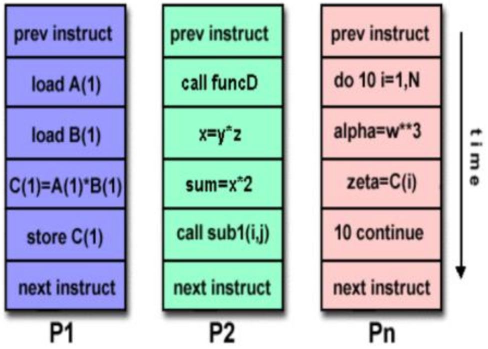


The MISD organization

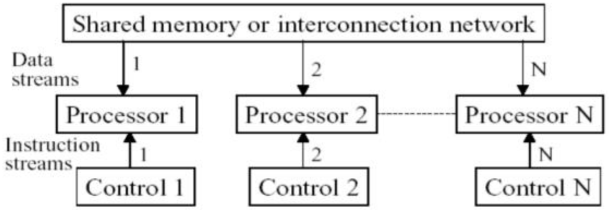


Contd…

- MIMD
  - Advanced parallel architecture
  - Multiple cores operate on multiple data streams, each executing independent instructions
  - Multi-core processors and distributed computing systems

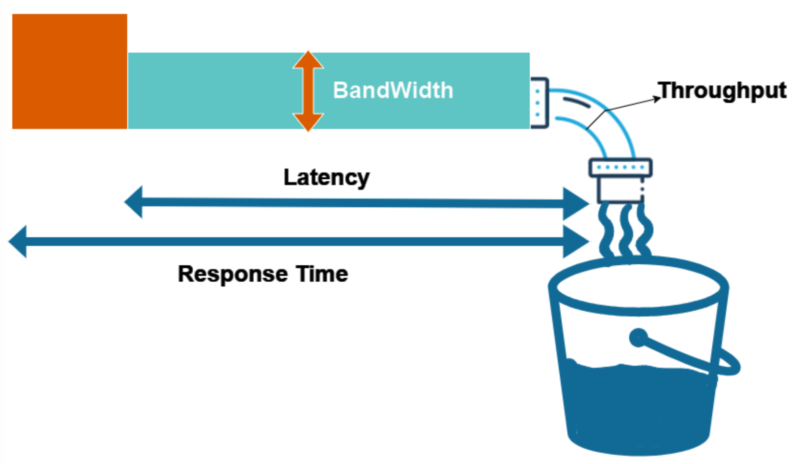


The MIMD organization



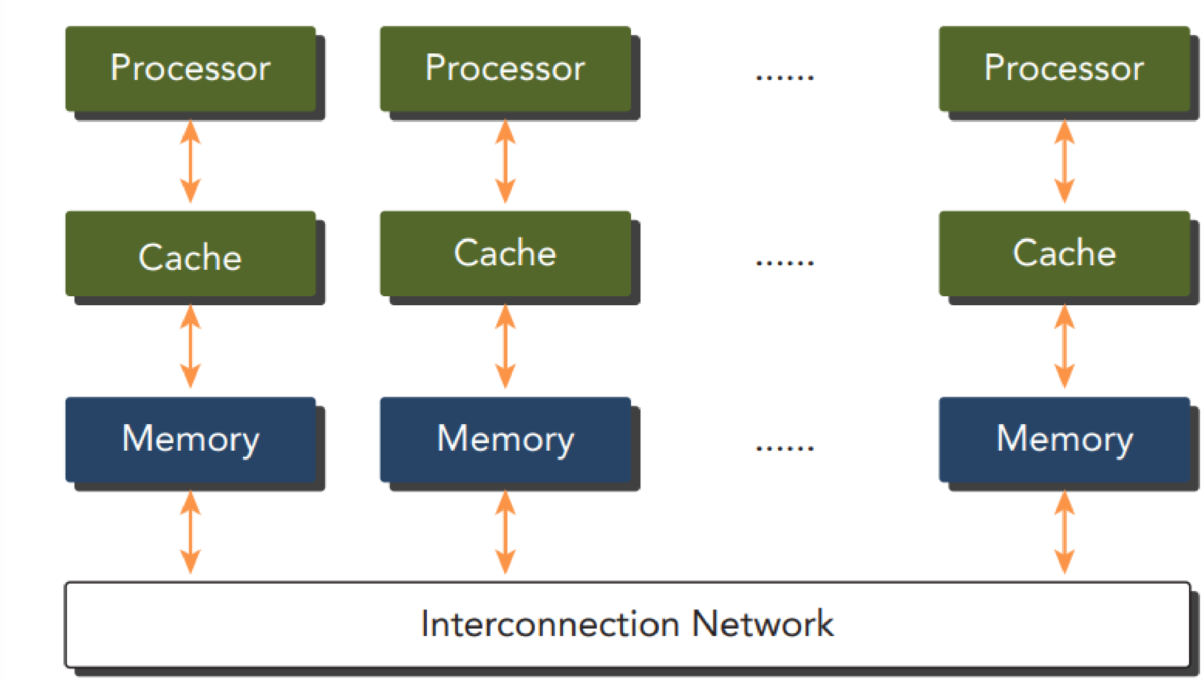
Enhancing Computational Efficiency: Key Objectives

- At the architectural level, significant advancements have been made to accomplish the following goals:
  - **Reduce Latency**: Minimizing the time taken for an operation to start and complete
  - **Enhance Bandwidth**: Increasing the volume of data that can be processed per unit of time
  - **Increase Throughput**: Maximizing the number of operations executed within a given timeframe



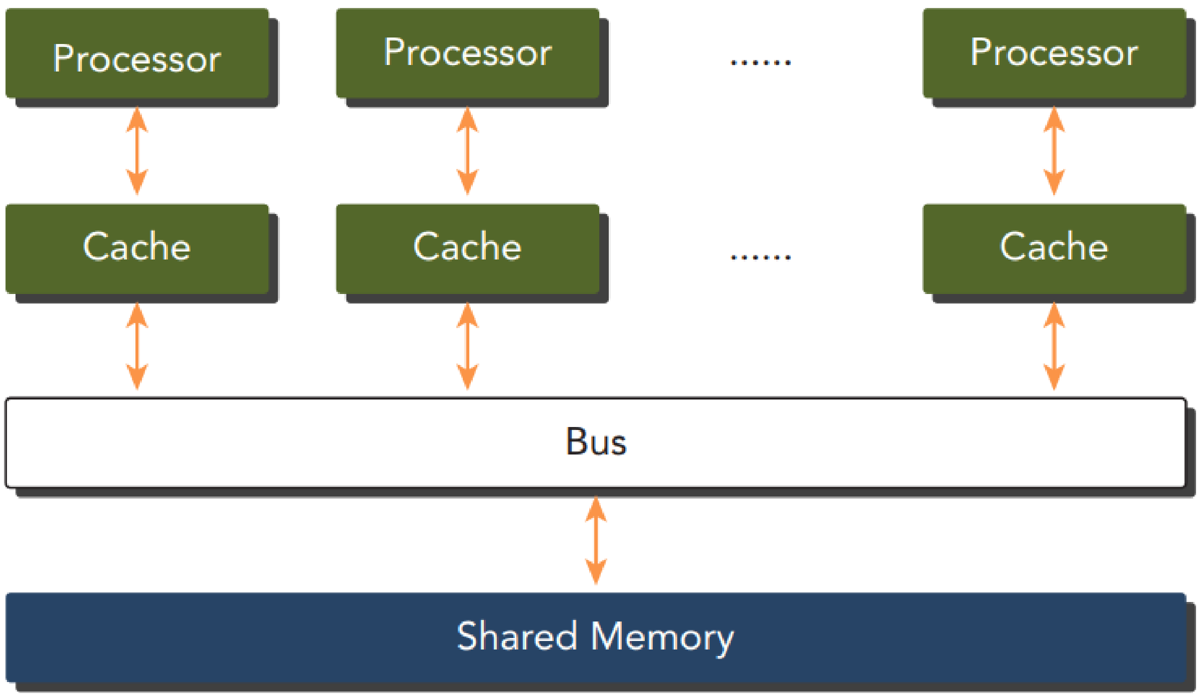
Classification of Computer Architectures by Memory Organization

- **Multi-node with distributed memory**
  - Many processors, each with local memory, communicate over a network
  - Suitable for clusters



Contd…

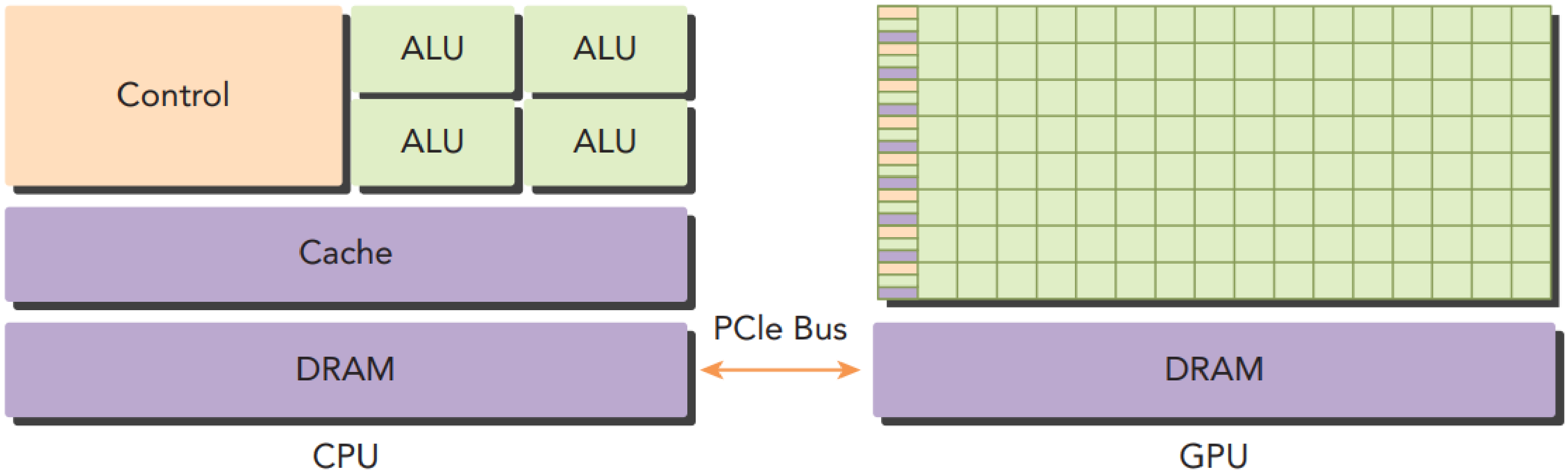
- **Multiprocessor with Shared Memory**
  - Multiple processors sharing a common memory space, enabling direct data access and faster inter-processor communication



Heterogeneous Computing

- **Homogeneous Computing**
  - Involves one or more processors of the same architecture to execute applications
  - In these systems, all processing units are identical and perform the same tasks in a similar manner
  - Developers can write programs that assume all processors behave identically, leading to simpler software design
- **Heterogeneous Computing**
  - Utilizes a variety of processor architectures to execute applications, allowing tasks to be assigned to the most suitable architecture for performance improvements
  - Systems can include CPUs, GPUs, FPGAs, and other specialized processors, each optimized for specific tasks
  - Programming for heterogeneous systems can be more complex as developers must manage different architectures and ensure efficient task distribution

Heterogeneous Architecture



Terminology:

Host: CPU

Device: GPU

PCIe: Peripheral Component Interconnect Express

End of Lecture 3