# Parallel and Distributed Computing CS3006 (BCS-6C/6D) Lecture 03

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

31 January, 2023

## Administrative Information

- Website: Google Classroom will be announced soon, iA
    - 6C: https://classroom.google.com/u/0/c/NTg1ODczMjk5NDg3
        - dt2nwb4
    - 6D: https://classroom.google.com/u/0/c/NTg1ODczNTQ3OTc5
        - 5puwvng

## Previous Lecture

- Parallel Computing v. Distributed Computing (shared-memory, distributed memory)
- Practical applications of PDC
- Limitations to Parallel and Distributed Computing
- Speedup
  - Amdahl’s Law
  - Karp-Flatt Metric
- Types of Parallelism

*Karp-Flatt Metric: Proposed by Alan H. Karp and Horace P. Flatt in 1990. To determine to what extent an algorithm is parallelized.*

*One more use case: -*
*if metric value remains consistent w.r.t. increasing number of processors and speedups → You need to reduce sequential fraction.*
*if metric value increases as increasing number of processors and speedups → You need to parallelize overheads.*

Solution: Compute $\mathrm{e}(n,p)$ corresponding to each data point:

| $p$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| $\Psi(n,p)$ | 1.82 | 2.50 | 3.08 | 3.57 | 4.00 | 4.38 | 4.71 |
| $\mathrm{e}(n,p)$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

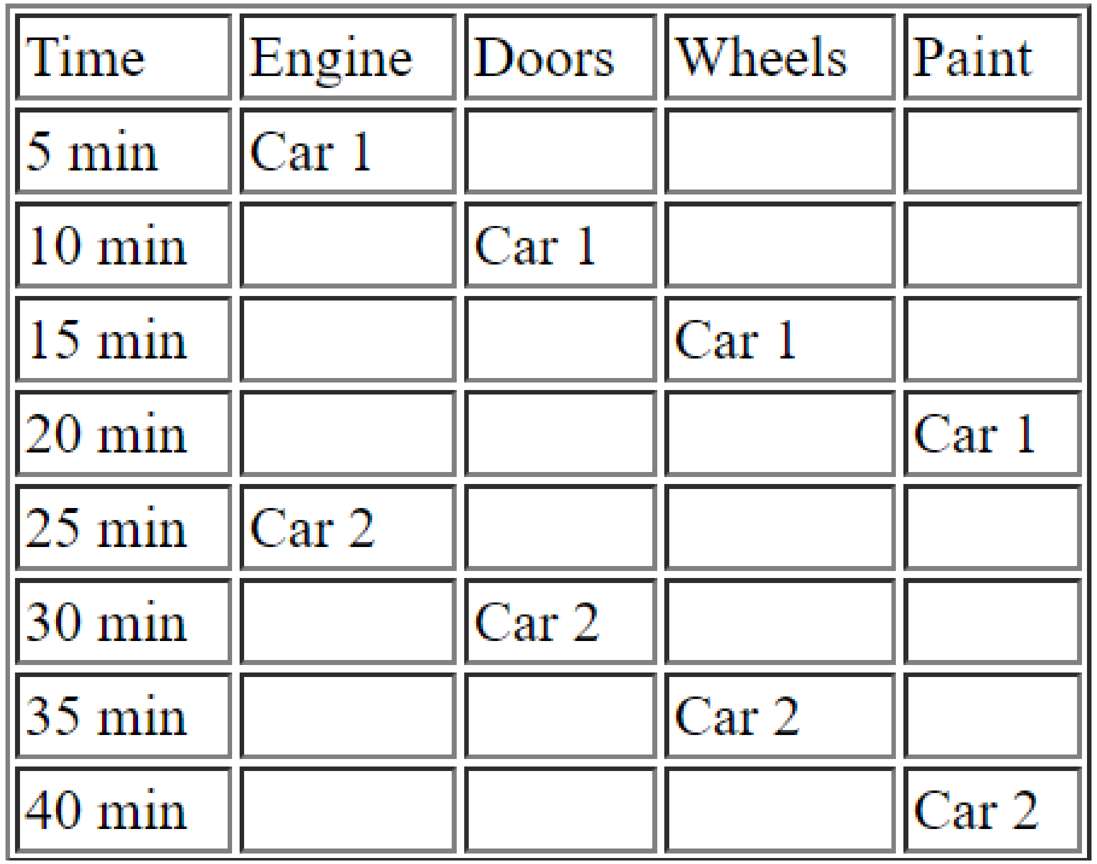
Since the experimentally determined serial fraction $\mathrm{e}(n,p)$ is not increasing with $p$, the primary reason for the poor speedup is the 10% of the computation that is inherently sequential. Parallel overhead is not the reason for the poor speedup.

*Parallel Overhead: The amount of time required to coordinate **parallel** tasks, as opposed to doing useful work. **Parallel overhead** can include factors such as: Task start-up time. Synchronizations.*

## Types of Parallelism

### 3. Pipelining

- Example: Assembly line analogy



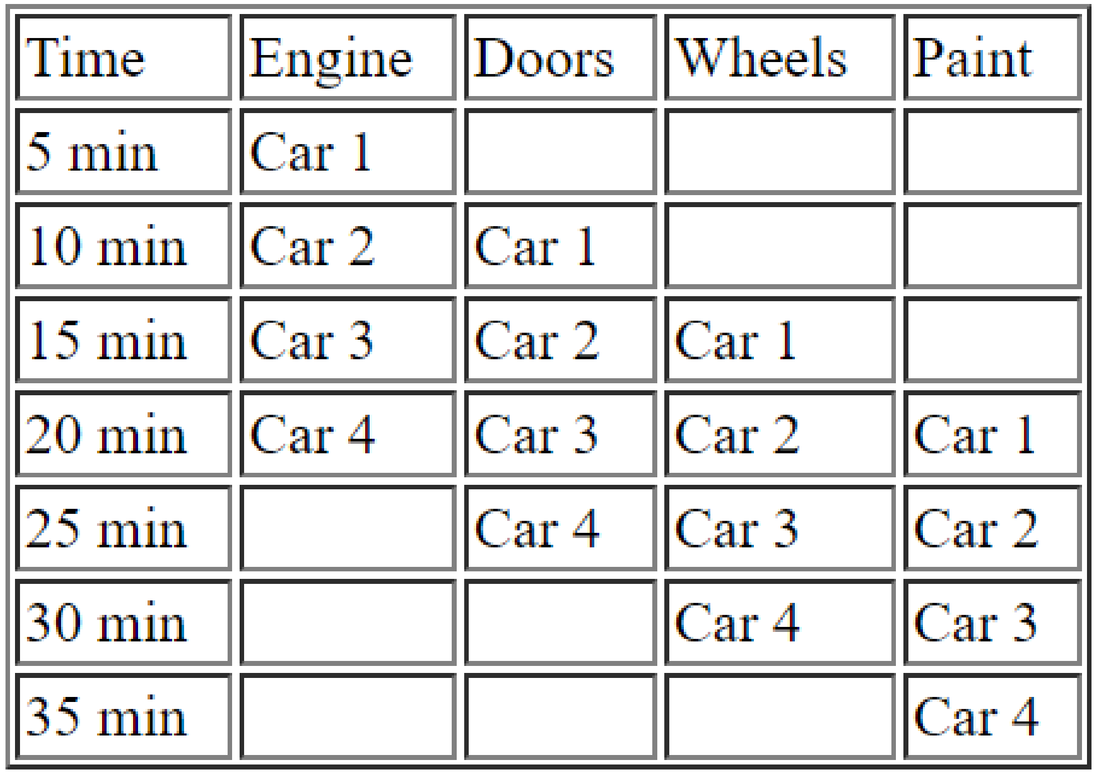
| Time | Engine | Doors | Wheels | Paint |
|---|---|---|---|---|
| 5 min | Car 1 | | | |
| 10 min | | Car 1 | | |
| 15 min | | | Car 1 | |
| 20 min | | | | Car 1 |
| 25 min | Car 2 | | | |
| 30 min | | Car 2 | | |
| 35 min | | | Car 2 | |
| 40 min | | | | Car 2 |

**Sequential Execution**

## Types of Parallelism

3. **Pipelining**

- Example: Assembly line analogy



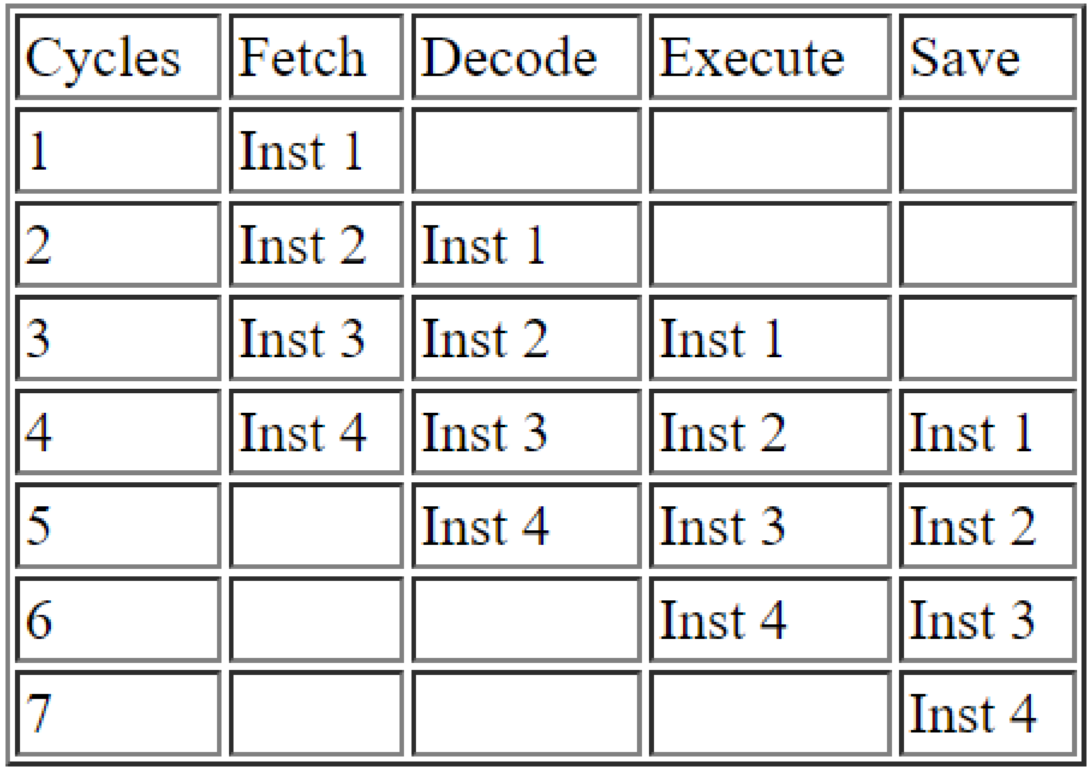
| Time | Engine | Doors | Wheels | Paint |
|---|---|---|---|---|
| 5 min | Car 1 | | | |
| 10 min | Car 2 | Car 1 | | |
| 15 min | Car 3 | Car 2 | Car 1 | |
| 20 min | Car 4 | Car 3 | Car 2 | Car 1 |
| 25 min | | Car 4 | Car 3 | Car 2 |
| 30 min | | | Car 4 | Car 3 |
| 35 min | | | | Car 4 |

**Pipelining**

## Types of Parallelism

### 3. Pipelining

- Example: Overlap instructions in a single instruction cycle to achieve parallelism



| Cycles | Fetch | Decode | Execute | Save |
|---|---|---|---|---|
| 1 | Inst 1 | | | |
| 2 | Inst 2 | Inst 1 | | |
| 3 | Inst 3 | Inst 2 | Inst 1 | |
| 4 | Inst 4 | Inst 3 | Inst 2 | Inst 1 |
| 5 | | Inst 4 | Inst 3 | Inst 2 |
| 6 | | | Inst 4 | Inst 3 |
| 7 | | | | Inst 4 |

**4-stage Pipelining**

## Multi-processor
## vs
## Multi-Computer

## Multi-Processor

- Multiple-CPUs with a shared memory

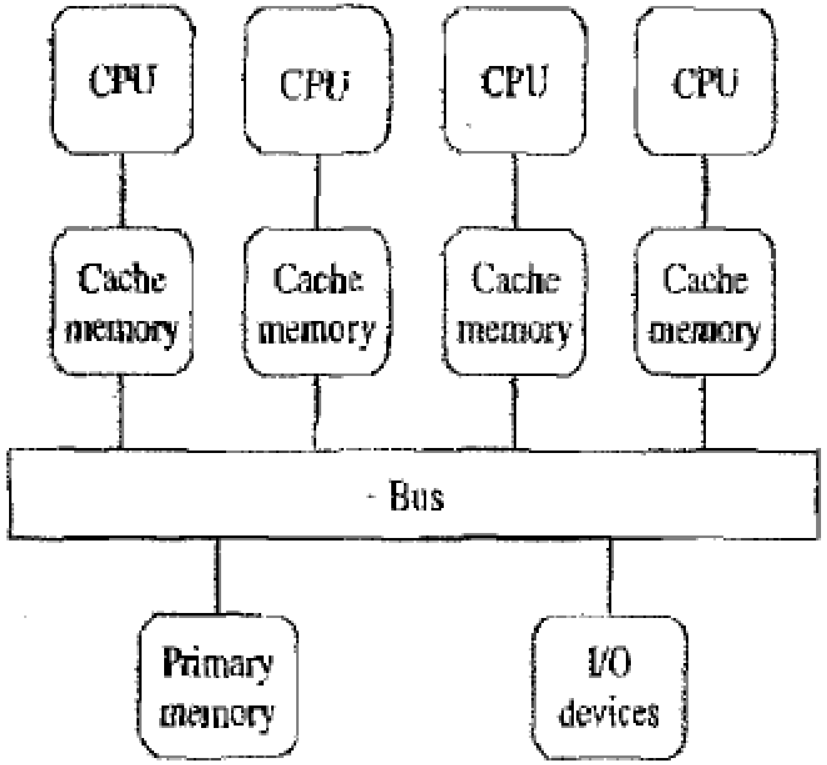
- The same address on two different CPUs refers to the same memory location.

- **Generally two categories:-**
  1. Centralized Multi-processors
  2. Distributed Multi-processor

## Multi-Processor

**i. Centralized Multi-processor**

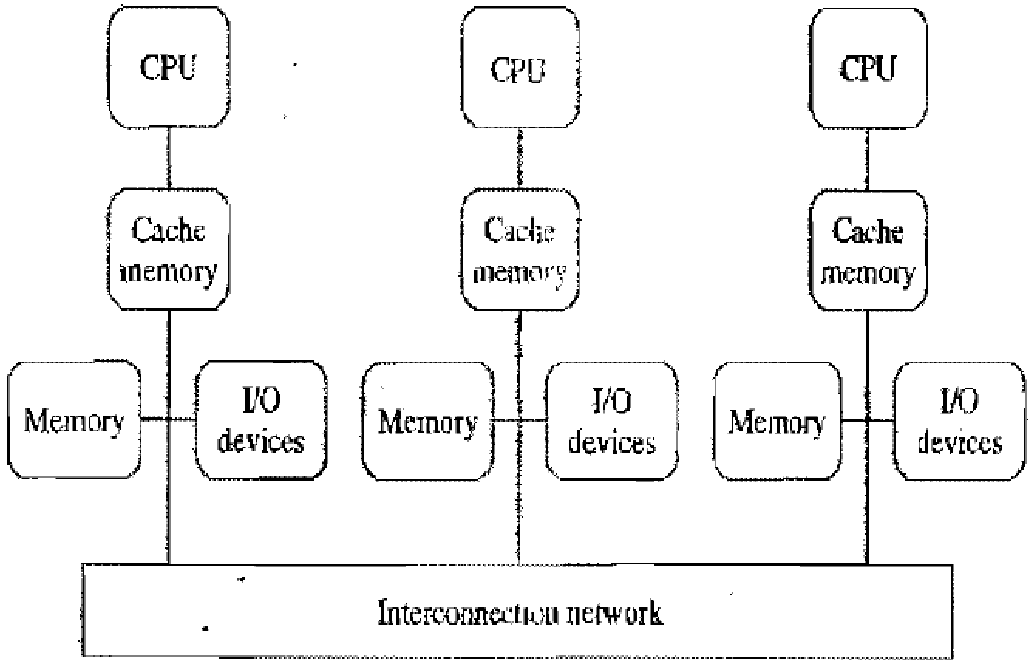
- Additional CPUs are attached to the system bus, and all the processors share the same primary memory
- All the memory is at one place and has the same access time from every processor
- Also known as **UMA** (Uniform Memory Access) multi-processor or **SMP** (symmetrical Multi-processor )



## Multi-Processor

**ii. Distributed Multi-processor**

- Distributed collection of memories forms one logical address space
- Again, the same address on different processors refers to the same memory location.
- Also known as non-uniform memory access (**NUMA**) architecture
- Because, memory access time varies significantly, depending on the physical location of the referenced address



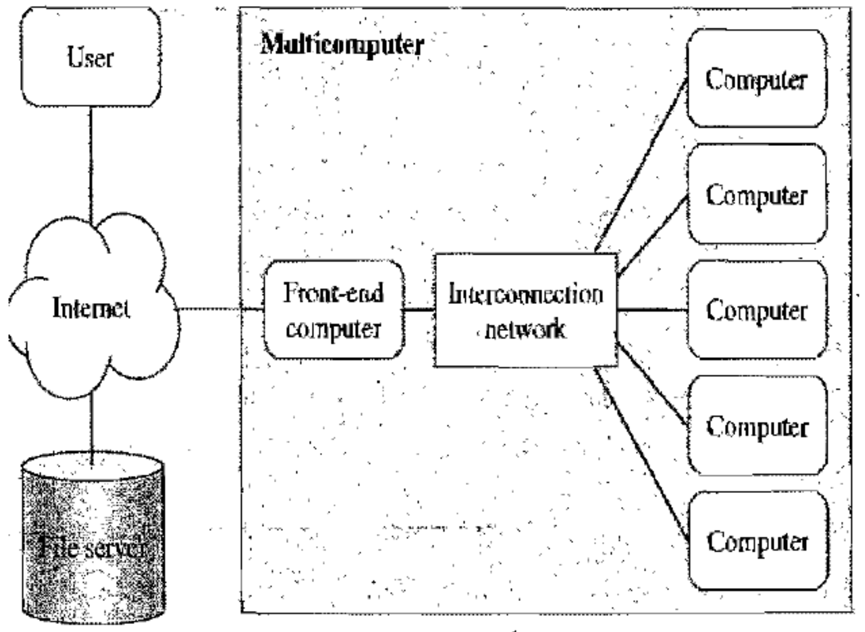
## Multi-Computer

- Distributed-memory, multi-CPU computer
- Unlike **NUMA** architecture, a multicomputer has disjoint local address spaces
- Each processor has direct access to their local memory only.
- The same address on different processors refers to two different physical memory locations.
- Processors interact with each other  through passing messages

## Multi-Computer

**Asymmetric Multi-Computers**

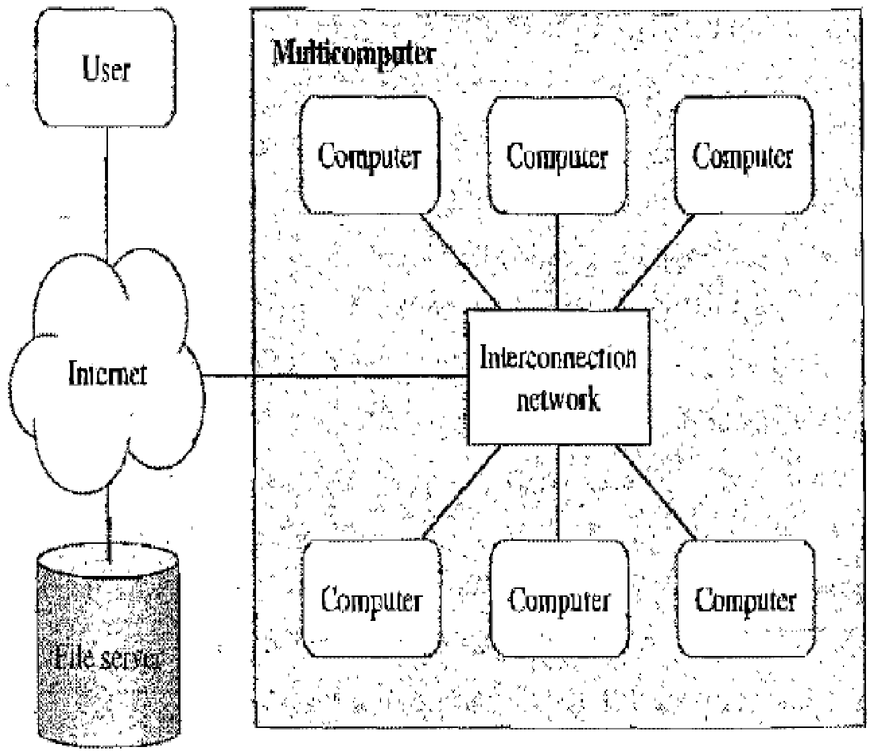
- A front-end computer that interacts with users and I/O devices
- The back-end processors are dedicatedly used for "number crunching"
- Front-end computer executes a full, multi-programmed OS and provides all functions needed for program development
- The backends are reserved for executing parallel programs



# Multi-Computer

## Symmetric Multi-Computers

- Every computer executes same OS
- Users may log into any of the computers
- This enables multiple users to concurrently login, edit and compile their programs.
- All the nodes can participate in execution of a parallel program



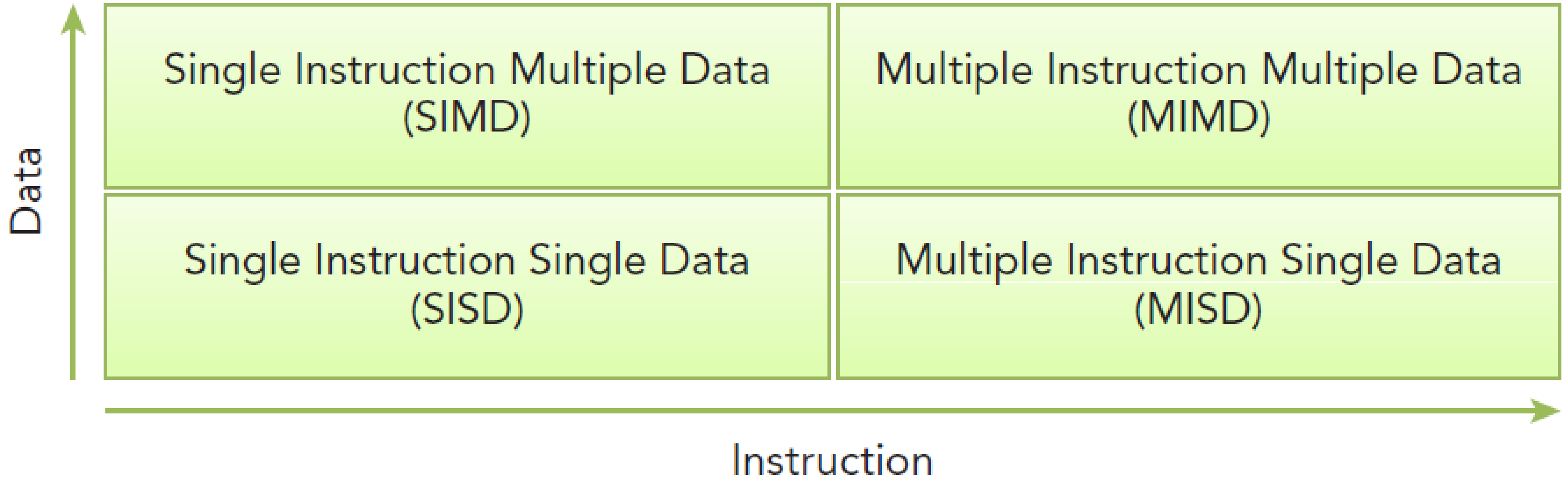
## Network of Workstations
## vs
## Cluster

| Cluster | Network of workstations |
|---|---|
| Usually a co-located collection of low-cost computers and switches, dedicated to running parallel jobs.<br>All computer run the same version of operating system. | A dispersed collection of computers. Individual workstations may have different Operating systems and executable programs |
| Some of the computers may not have interfaces for the users to login | User have the power to login and power off their workstations |
| Commodity cluster uses high speed networks for communication such as fast Ethernet@100Mbps, gigabit Ethernet@1000 Mbps and Myrinet@1920 Mbps. | Ethernet speed for this network is usually slower. Typically in the range of 10 Mbps |

## Flynn’s Taxonomy

## Flynn's Taxonomy

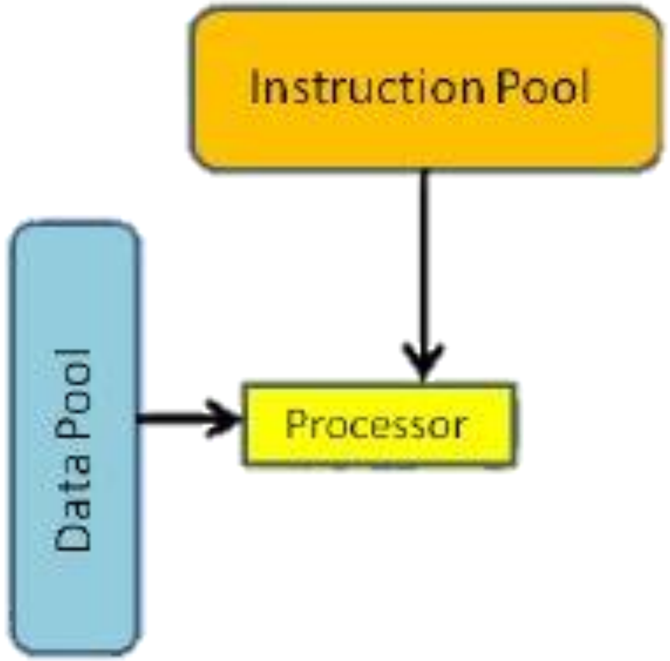
- Widely used architectural classification scheme
- Classifies architectures into four types
- The classification is based on how data and instructions flow through the cores.


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

# Flynn’s Taxonomy

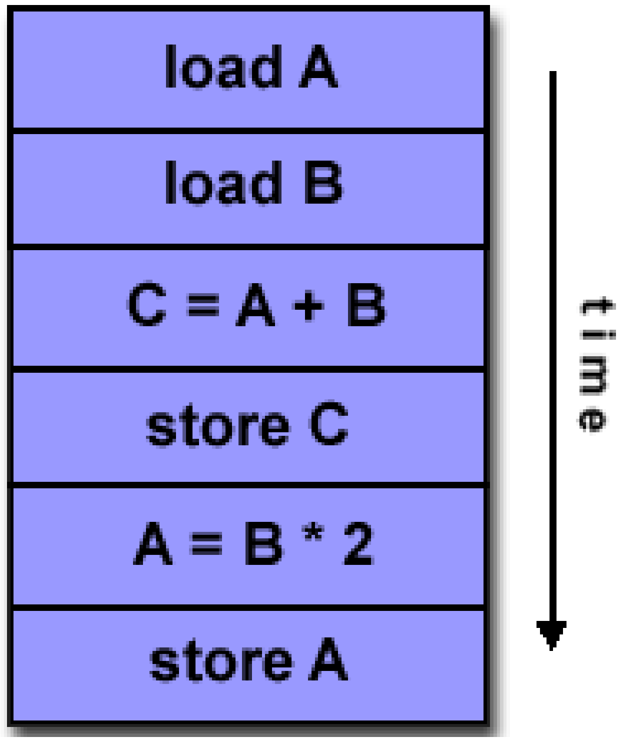
**SISD (Single Instruction Single Data)**

- Refers to the traditional computer: a serial architecture
- This architecture includes single core computers
- Single instruction stream is in execution at a given time
- Similarly, only one data stream is active at any time



*Not parallel, classical Von Neumann architecture*
*Parallelism can be introduced using pipelining*

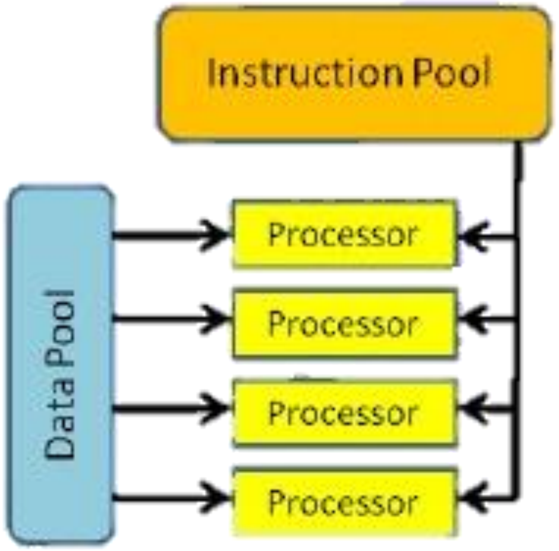
## Example of SISD:



# Flynn's Taxonomy

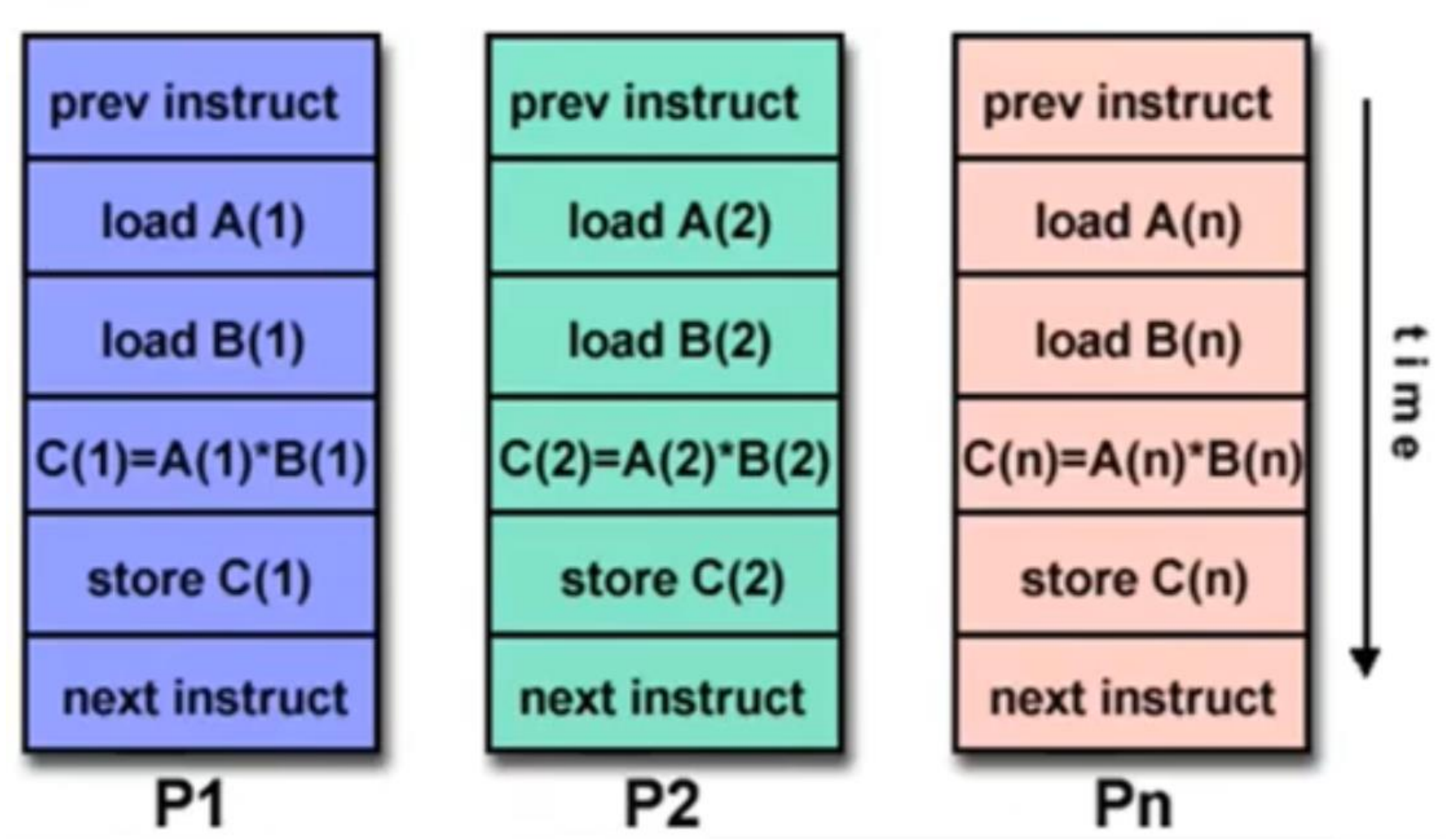
**SIMD (Single Instruction Multiple Data)**

- Refers to parallel architecture with multiple cores
- All the cores execute the same instruction stream at any time but, data stream is different for each.
- Well-suited for scientific operations requiring large matrix-vector operations
- Vector computers (Cray vector processing machine) and Intel co-processing unit 'MMX' fall under this category.
- Used with array operations, image processing and graphics



Array: same operations on different array elements.
Replaces the loops
Image: Applying same operation on different pixels

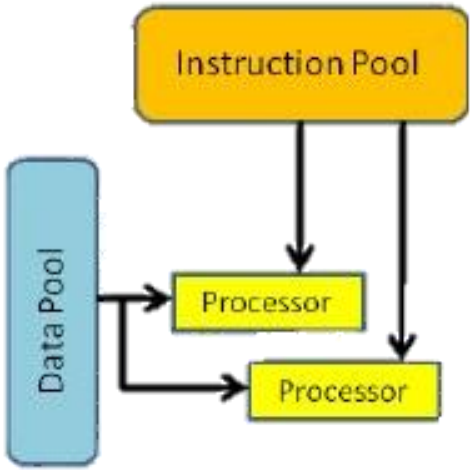
## Example of SIMD:

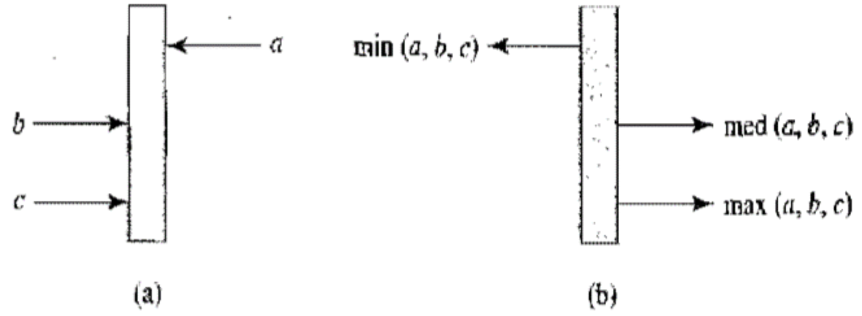


# Flynn’s Taxonomy

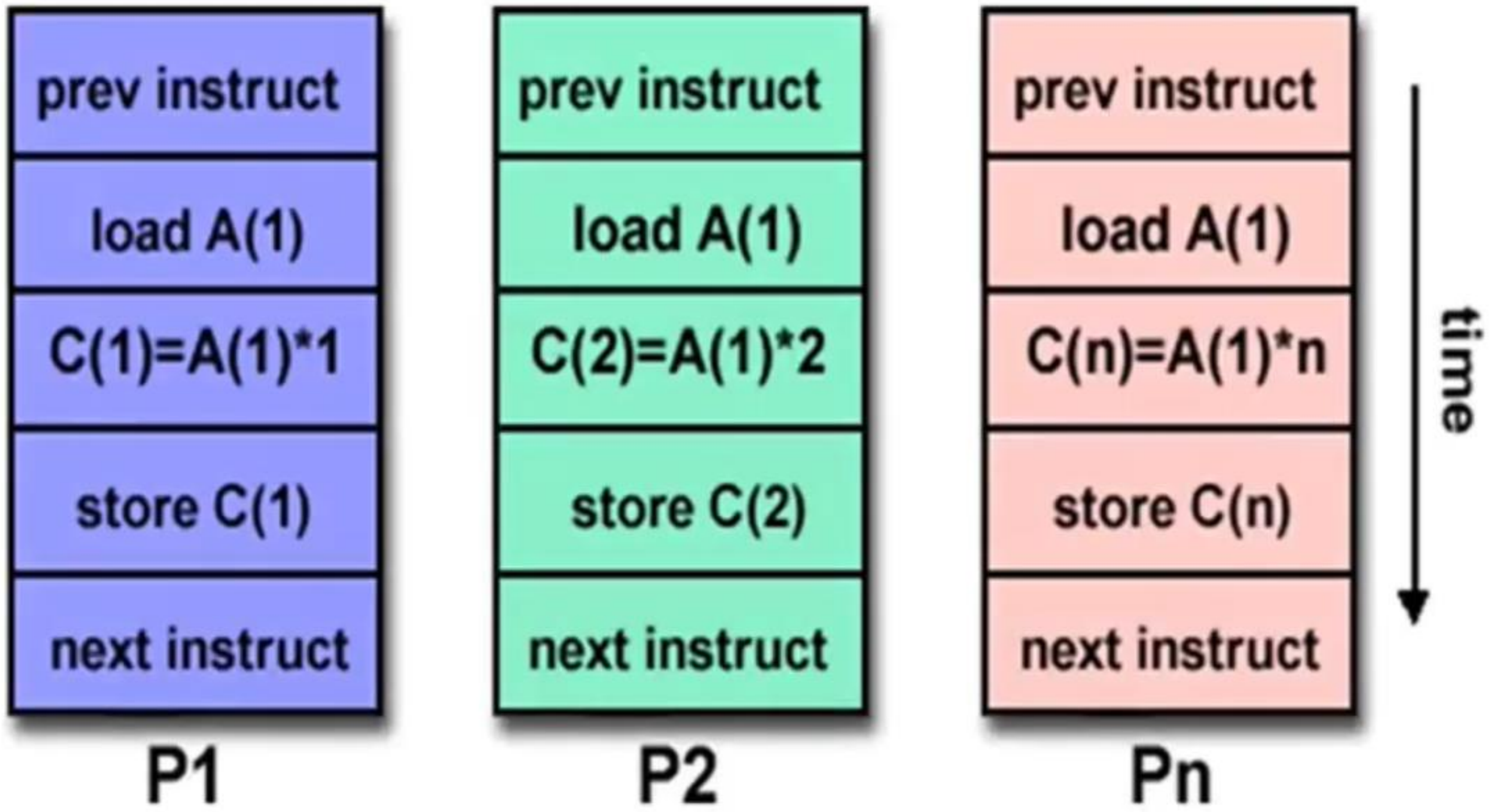
**MISD (Multiple Instructions Single Data)**

- Multiple instruction stream and single data stream
    - A pipeline of multiple independently executing functional units
    - Each operating on a single stream of data and forwarding results from one to the next
- Rarely used in practice
- E.g., Systolic arrays : network of primitive processing elements that pump data.





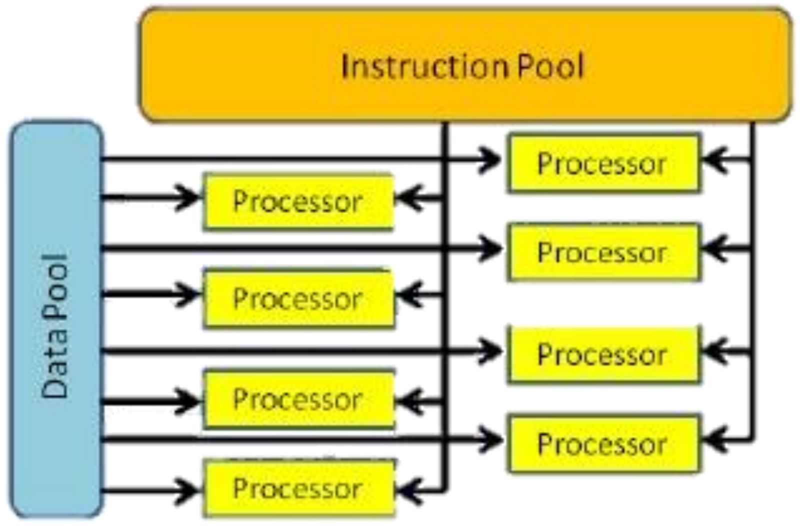
# Example of MISD:



# Flynn’s Taxonomy

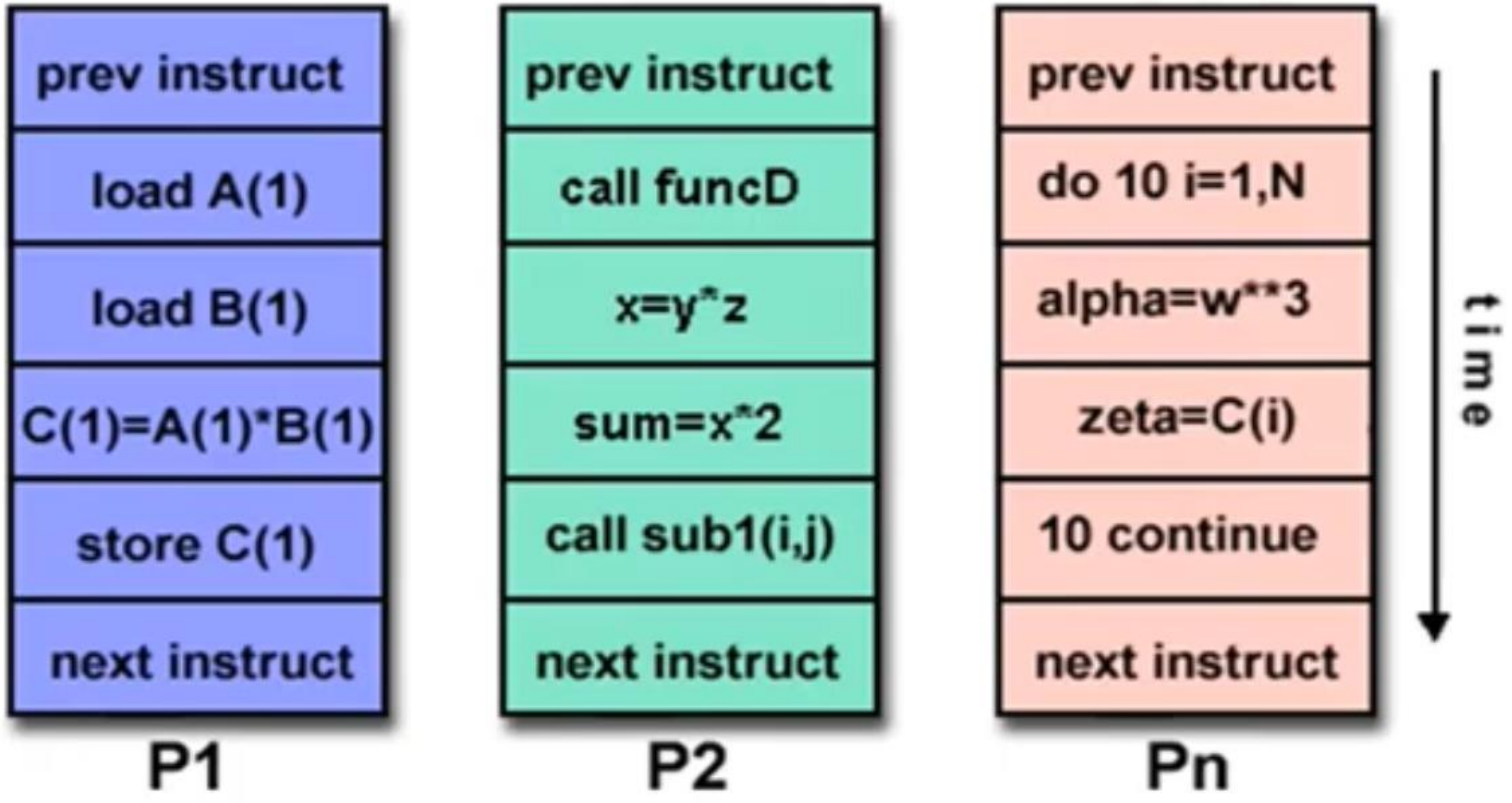
**MIMD (Multiple Instructions Multiple Data)**

- Multiple instruction streams and multiple data streams
- Different CPUs can simultaneously execute different instruction streams manipulating different data
- Most of the modern parallel architectures fall under this category e.g., **Multiprocessor** and **multicomputer** architectures
- Many MIMD architectures include SIMD executions by default.



e.g. super computers

## Example of MIMD:



## Reading Assignment

- Cache Coherence and Snooping
- Branch prediction and issues while pipelining the problem

## Sources

- Slides of Dr. Rana Asif Rahman, FAST