Parallel and Distributed Computing

CS3006

Lecture 3

Flynn's Taxonomy

12th February 2024

Dr. Rana Asif Rehman

Agenda

- **A Quick Review**
- **Flynn's Taxonomy**
  - SISD
  - MISD
  - SIMD
  - MIMD
- **Physical Organization of Parallel Platforms**
  - PRAM
- **Routing techniques and Costs**

Quick Review to the Previous Lecture

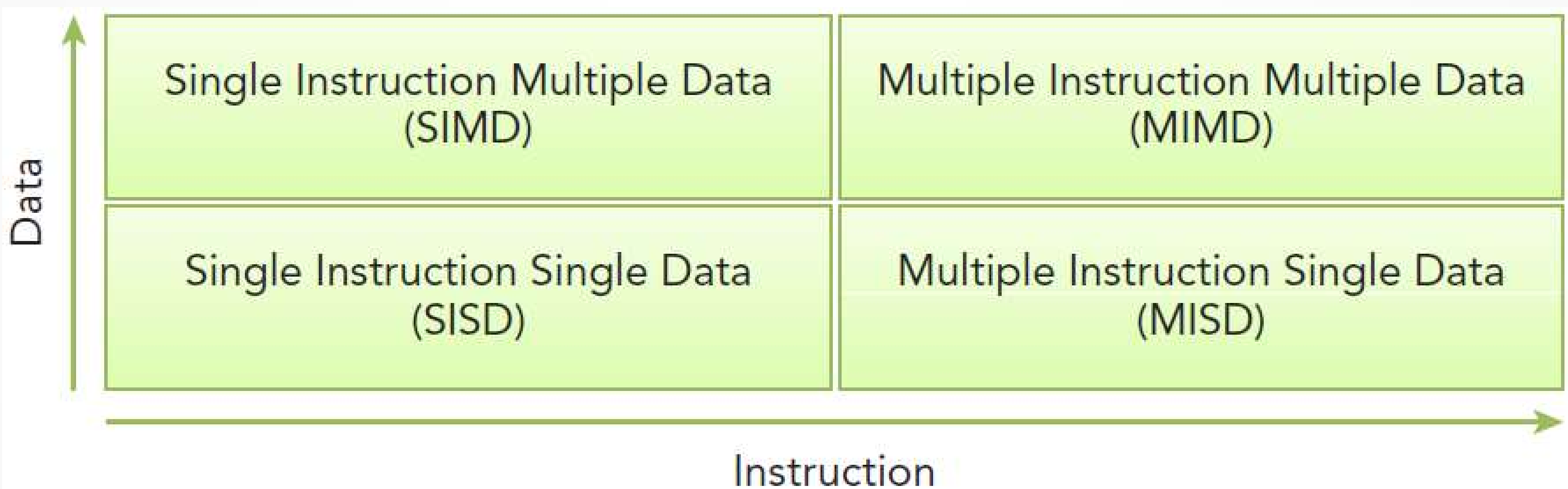
- **Amdahl's Law of Parallel Speedup**
  - Purpose, derivation, and examples
- **Karp-Flatt Metric**
  - Finding sequential fraction in the given parallel setup
- **Types of Parallelism**
  - Data-parallelism
    - Same operation on different data elements
  - Functional-parallelism
    - Different independent tasks with different operations on different data elements can be parallelized
  - Pipelining
    - Overlapping the instructions in a single instruction cycle to achieve parallelism

Quick Review to the Previous Lecture

- **Multiprocessor**
  - Centralized multiprocessor
  - Distributed multiprocessor
  - Shared address space(NUMA) vs Shared memory(UMA)
- **Multicomputer**
  - Asymmetrical
  - Symmetrical
- **Cluster vs Network of Workstations**
- **Assigned Reading**
  - Cache Coherence and snooping
  - Branch prediction and issues while pipelining the problem

Flynn's Taxonomy

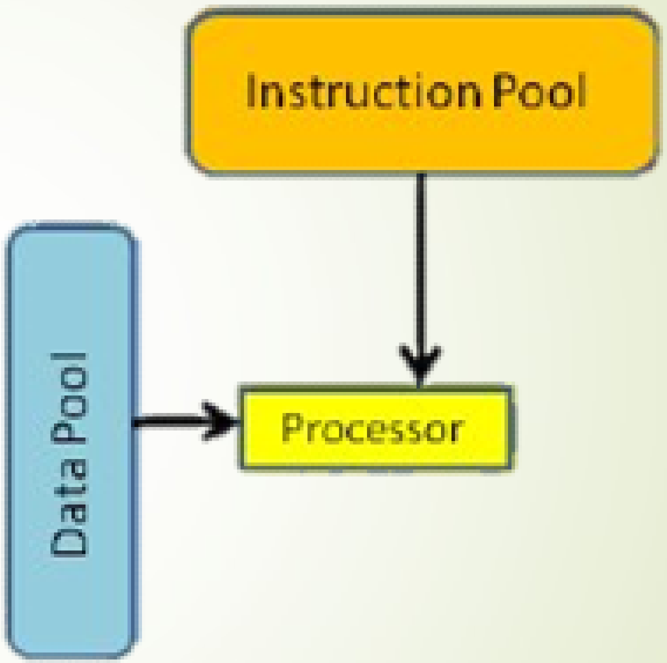
- Widely used architectural classification scheme
- Classifies architectures into four types
- The classification is based on how data and instructions flow through the cores.



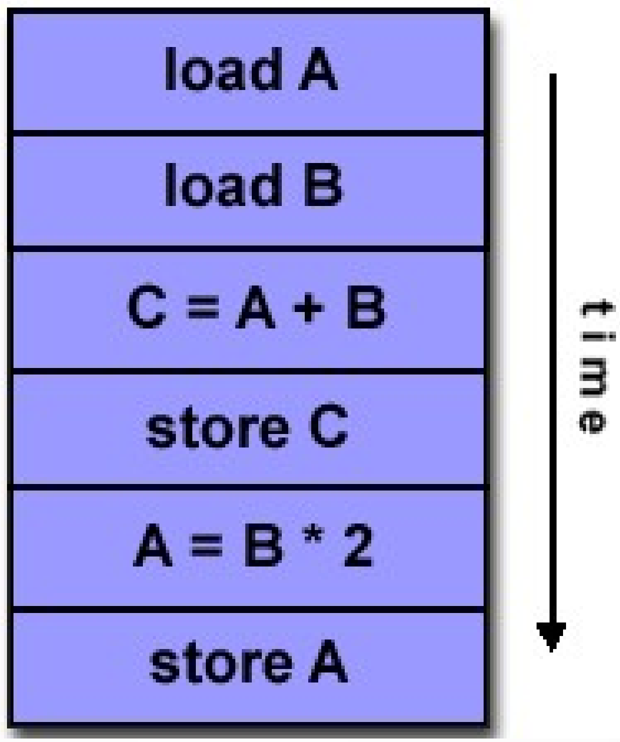
Flynn’s Taxonomy

SISD (Single Instruction Single Data)

- Refers to 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



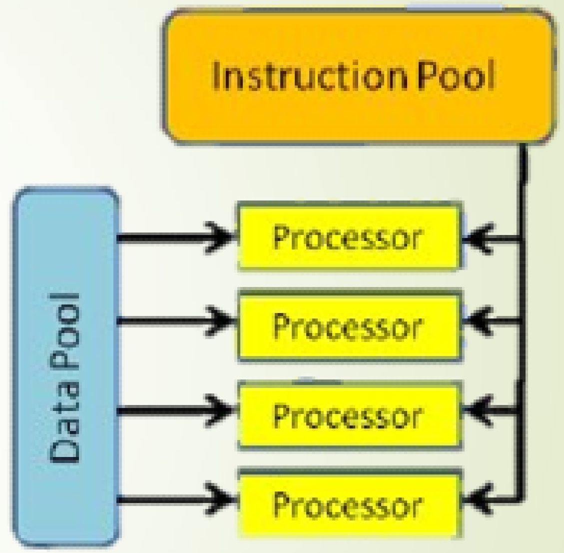
Example of SISD:



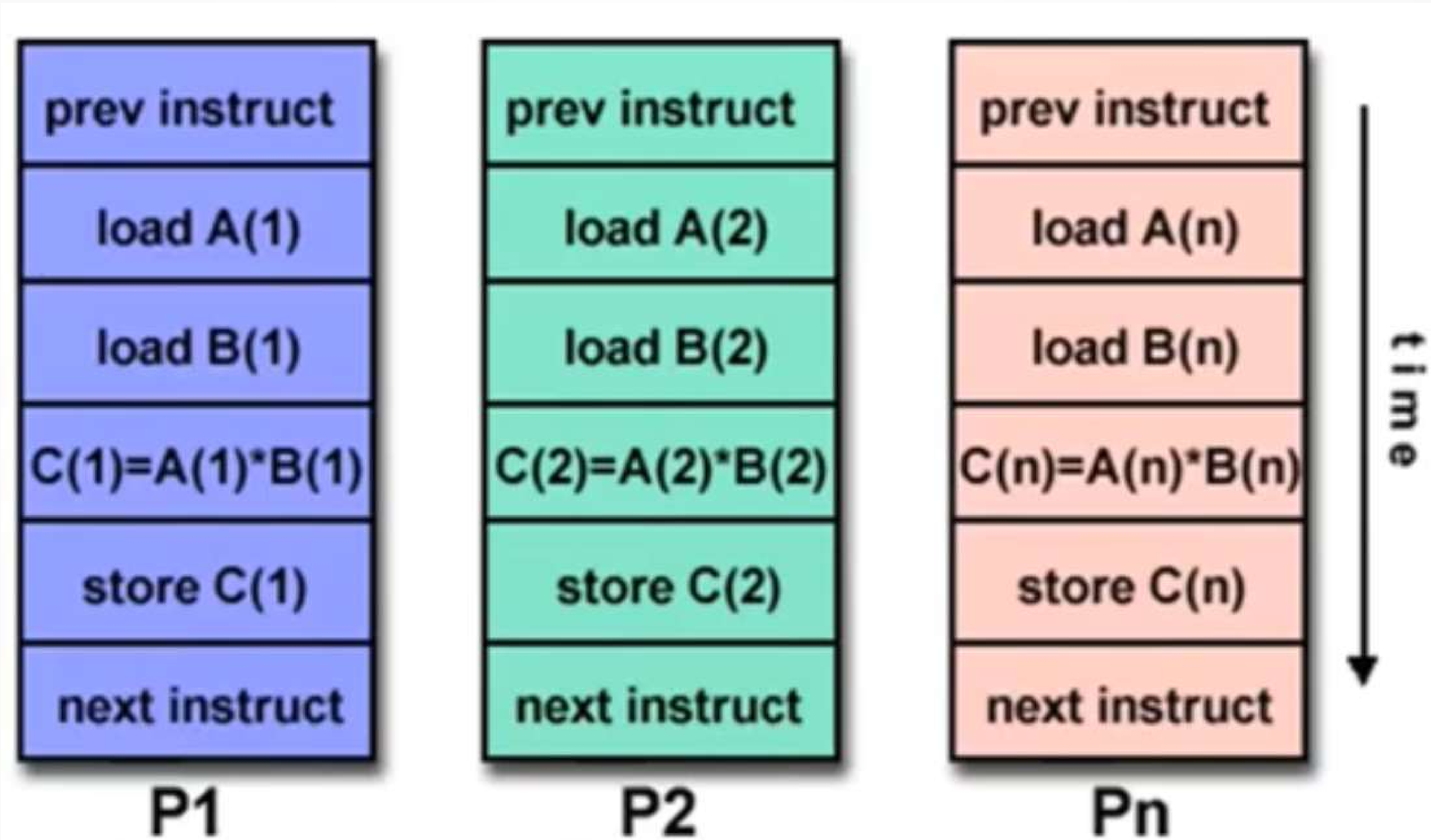
Flynn’s Taxonomy

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 the each.
- Well-suited for the 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



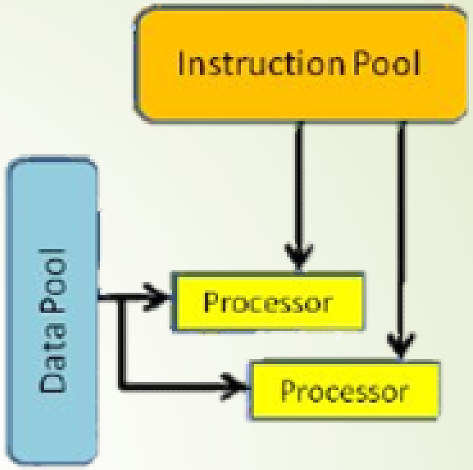
Example of SIMD:

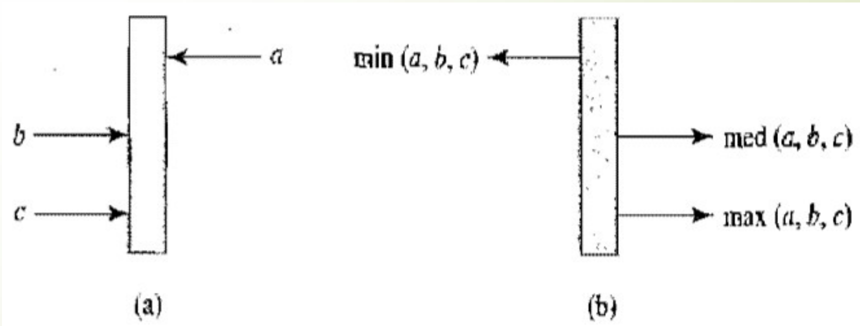


Flynn's Taxonomy

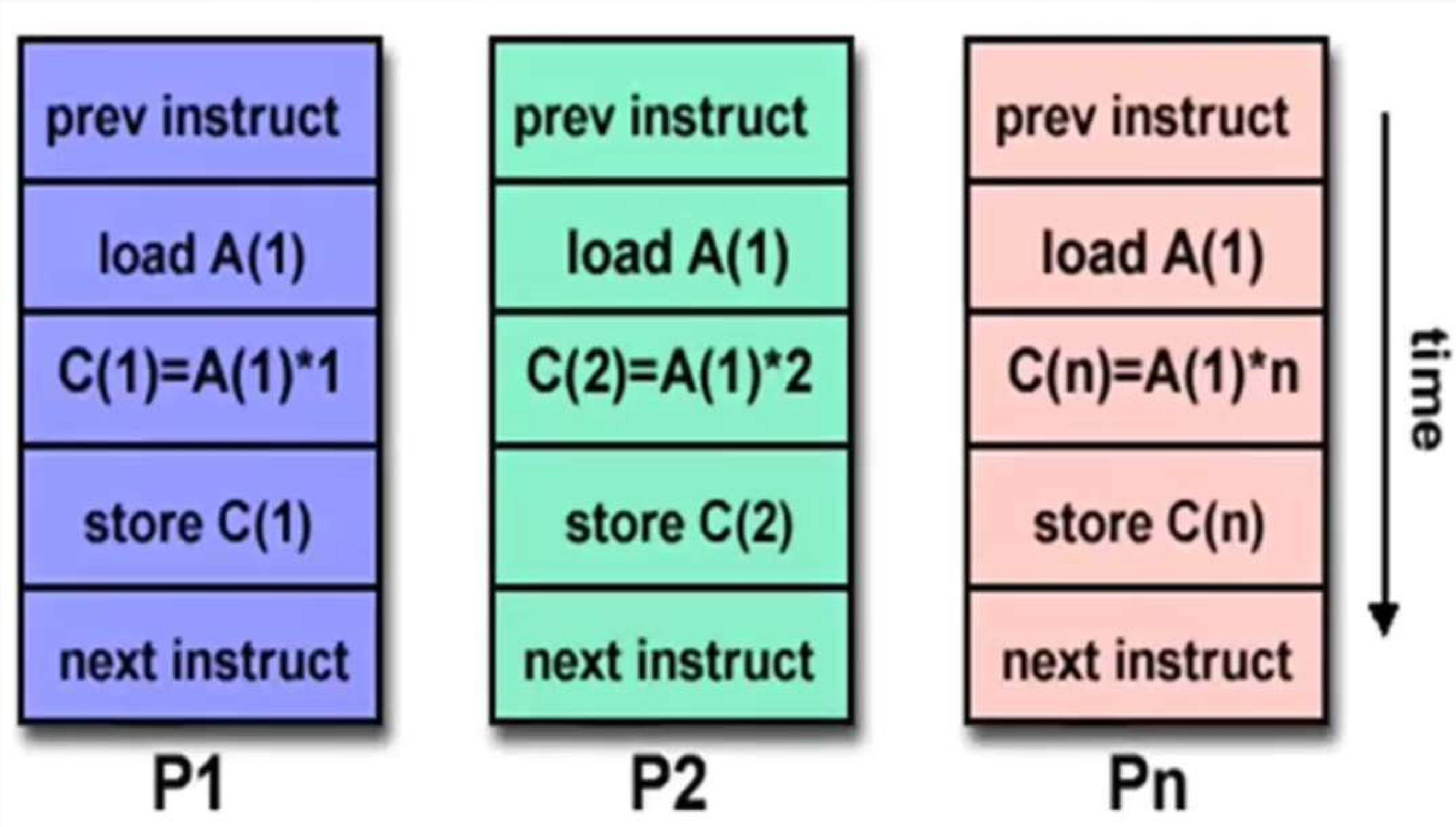
MISD (Multiple Instructions Single Data)

- 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.





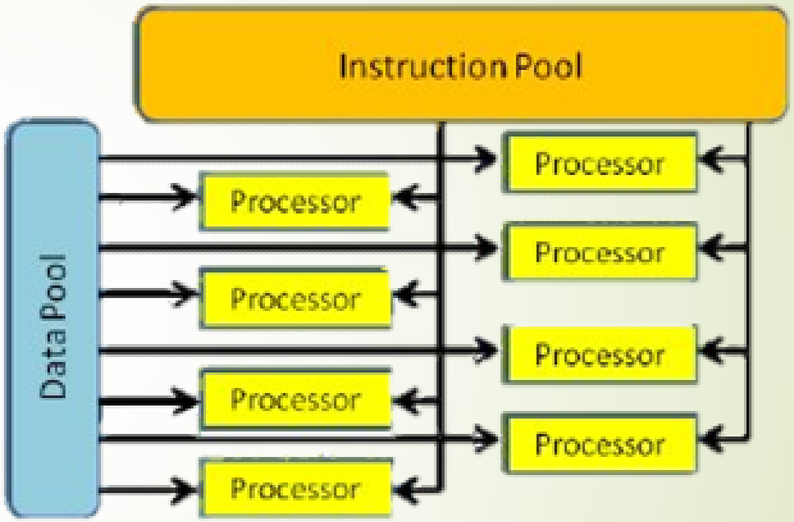
Example of MISD:



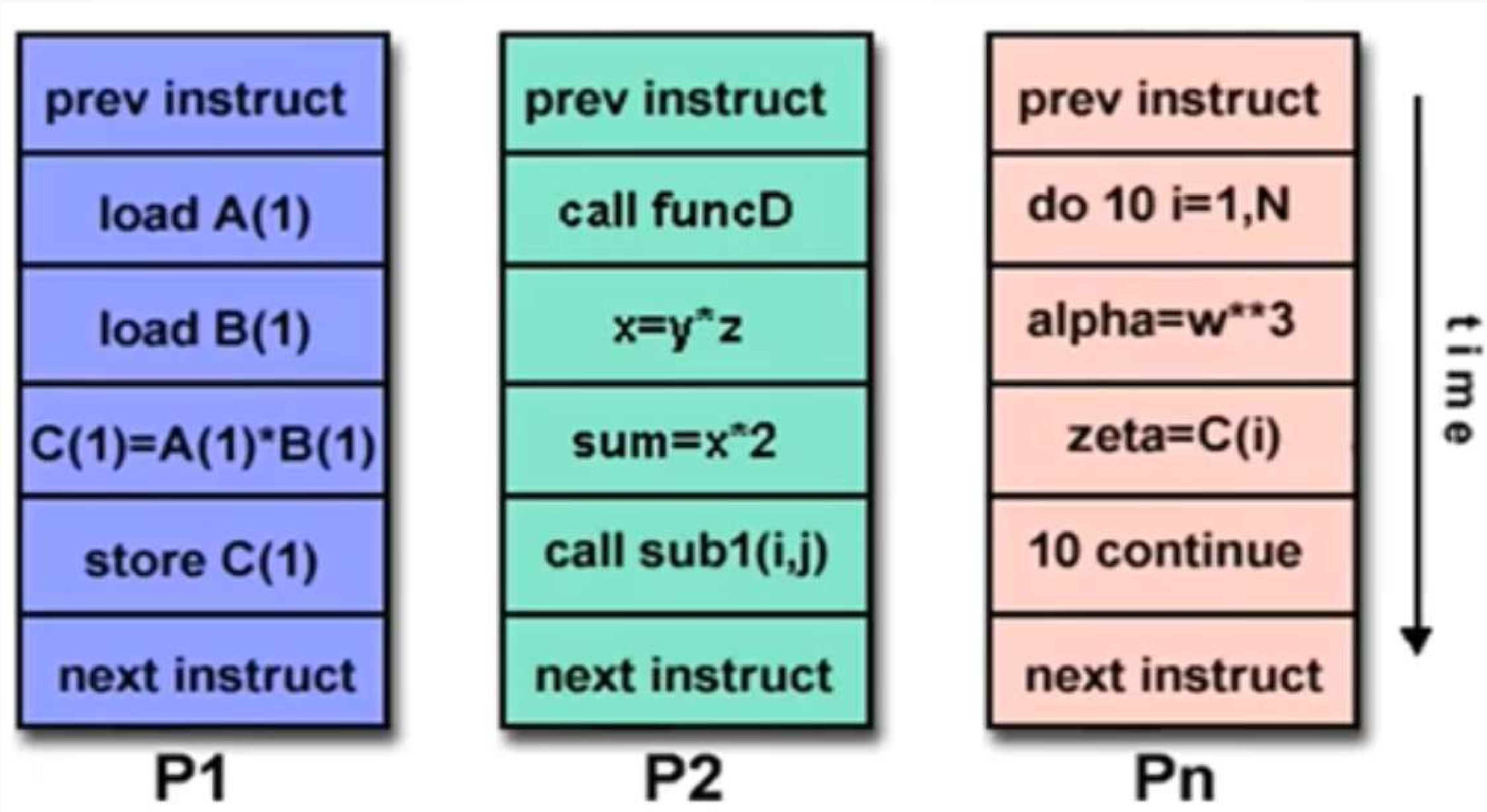
Flynn's Taxonomy

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.

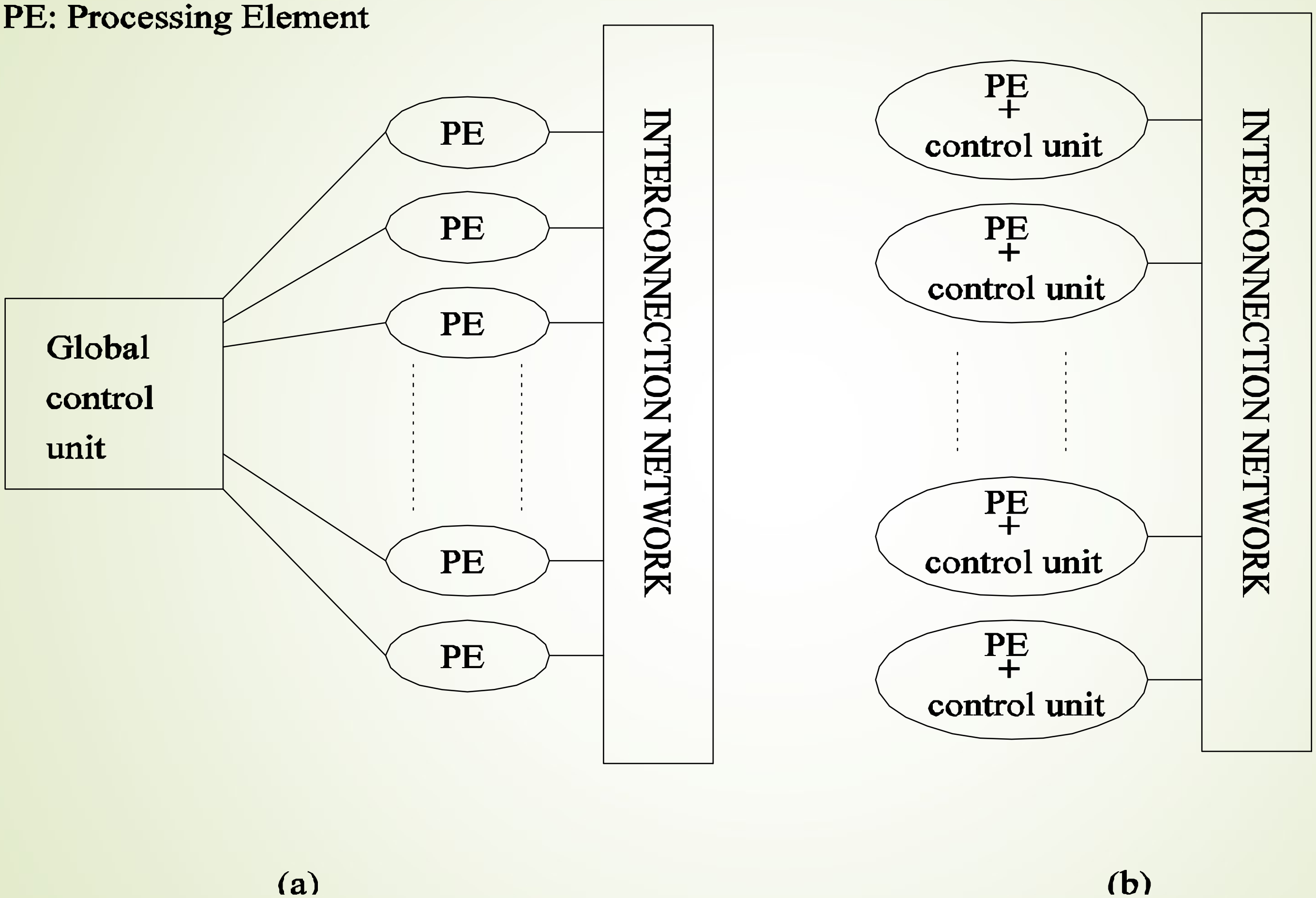


Example of MIMD:



Flynn's Taxonomy

PE: Processing Element



A typical SIMD architecture (a) and a typical MIMD architecture (b).

SIMD-MIMD Comparison

- SIMD computers require less hardware than MIMD computers (single control unit).
- However, since SIMD processors are specially designed, they tend to be expensive and have long design cycles.
- Not all applications are naturally suited to SIMD processors.
- In contrast, platforms supporting the SPMD (Same Program Multiple Data) paradigm can be built from inexpensive off-the-shelf components with relatively little effort in a short amount of time.
  - The Term SPMD is close variant of MIMD

Physical Organization of Parallel Platforms

Old Architecture

Random Access Machine (RAM)

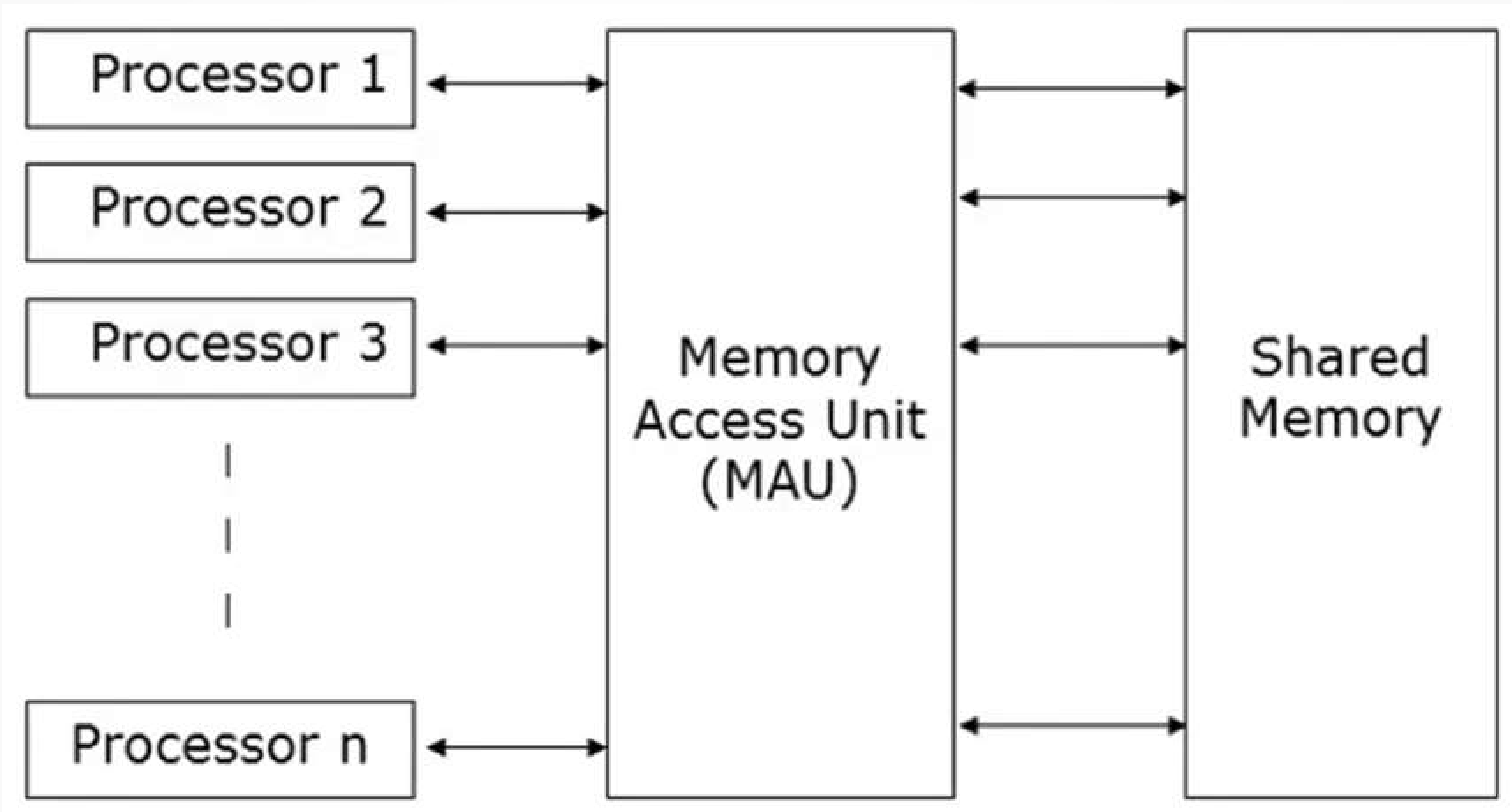
- RAM is a favorite model of a sequential computer
  - Computation unit with a user defined program
  - Read-only input tape and write-only tape
  - Unbounded number of local memory cells
  - Execution starts with the first instruction and ends when a HALT instruction is executed

Architecture of an Ideal Parallel Computer

Parallel Random Access Machine (PRAM)

- An extension to ideal sequential model: random access machine (RAM)
- Earlier and best-known model of parallel computation
- PRAMs consist of ***p*** processors
- A global memory
  - Unbounded size
  - Uniformly accessible to all processors with same address space
- Processors share a common clock but may execute different instructions in each cycle.
- Based on simultaneous memory access mechanisms, PRAM can further be classified.

Graphical representation of PRAM:



Architecture of an Ideal Parallel Computer

Parallel Random Access Machine (PRAM)

- PRAMs can be divided into four subclasses.

1. Exclusive-read, exclusive-write (EREW) PRAM
   - No two processors can perform read/write operations concurrently
   - Weakest PRAM model, provides minimum memory access concurrency
2. Concurrent-read, exclusive-write (CREW) PRAM
   - All processors can read concurrently but can't write at same time
   - Multiple write accesses to a memory location are serialized
3. Exclusive-read, concurrent-write (ERCW) PRAM
   - No two processors can perform read operations concurrently, but can write
4. Concurrent-read, concurrent-write (CRCW) PRAM
   - Most powerful PRAM model

Architecture of an Ideal Parallel Computer

Parallel Random Access Machine (PRAM)

- Concurrent reads do not create any semantic inconsistencies
- But, What about concurrent write?
- Need of an arbitration(mediation) mechanism to resolve concurrent write access

Architecture of an Ideal Parallel Computer

Parallel Random Access Machine (PRAM)

- Mostly used arbitration protocols: -
  - **Common:** write only if all values that the processors are attempting to write are identical
  - **Arbitrary:** write the data from a randomly selected processor and ignore the rest.
  - **Priority:** follow a predetermined priority order. Processor with highest priority succeeds and the rest fail.
  - **Sum:** Write the sum of the data items in all the write requests. The sum-based write conflict resolution model can be extended for any of the associative operators, that is defined for data being written .

Architecture of an Ideal Parallel Computer

Physical Complexity of an Ideal Parallel Computer

- Processors and memories are connected via switches.
- Since these switches must operate in $O(1)$ time at the level of words, for a system of p processors and memory m words, the switch complexity is $O(mp)$.
- Clearly, for meaningful values of p and m, a true PRAM is not realizable.

Communication Costs in Parallel Machines

Communication Costs in Parallel Machines

- Along with idling (doing nothing) and contention (conflict e.g., resource allocation), **communication** is a major overhead in parallel programs.
- The communication cost is usually dependent on a number of features including the following:
  - Programming model for communication
  - Network topology
  - Data handling and routing
  - Associated network protocols
- Usually, distributed systems suffer from major communication overheads.

Message Passing Costs in Parallel Computers

- The total time to transfer a message over a network comprises of the following:
  - ***Startup time (t_s):*** Time spent at sending and receiving nodes (preparing the message[adding headers, trailers, and parity information] , executing the routing algorithm, establishing interface between node and router, etc.).
  - ***Per-hop time (t_h):*** This time is a function of number of hops (steps) and includes factors such as switch latencies, network delays, etc.
    - Also known as **node latency**.
  - ***Per-word transfer time (t_w):*** This time includes all overheads that are determined by the length of the message. This includes bandwidth of links, and buffering overheads, etc.

Message Passing Costs in Parallel Computers

Store-and-Forward Routing

- A message traversing multiple hops is completely received at an intermediate hop before being forwarded to the next hop.
- The total communication cost for a message of size ***m*** words to traverse ***l*** communication links is

$$t_{comm} = t_s + (mt_w + t_h)l.$$

- In most platforms, $\boldsymbol{t_h}$ is small and the above expression can be approximated by

$$t_{comm} = t_s + mlt_w.$$

Message Passing Costs in Parallel Computers

Packet Routing

- Store-and-forward makes poor use of communication resources.
- Packet routing breaks messages into packets and pipelines them through the network.
- Since packets may take different paths, each packet must carry routing information, error checking, sequencing, and other related header information.
- The total communication time for packet routing is approximated by: $t_{comm} = t_s + t_h l + t_w m.$
- Here factor t_w also accounts for overheads in packet headers.

Message Passing Costs in Parallel Computers

Cut-Through Routing

- Takes the concept of packet routing to an extreme by further dividing messages into basic units called **flits** or flow control digits.
- Since flits are typically small, the header information must be minimized.
- This is done by forcing all flits to take the same path, in sequence.
- A tracer message first programs all intermediate routers. All flits then take the same route.
- Error checks are performed on the entire message, as opposed to flits.
- No sequence numbers are needed.

Message Passing Costs in Parallel Computers

Cut-Through Routing

- The total communication time for cut-through routing is approximated by:

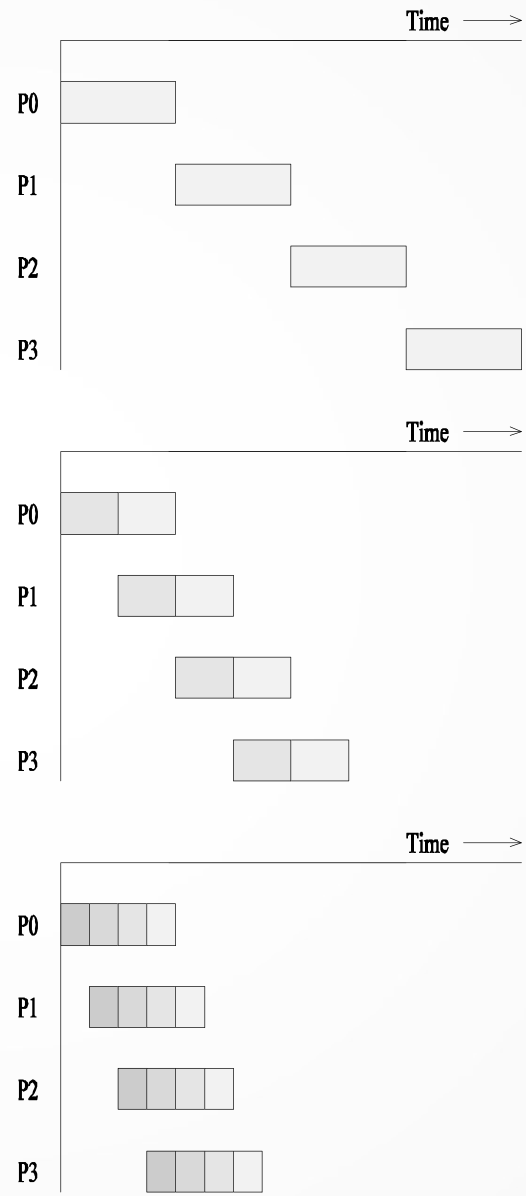
$$t_{comm} = t_s + t_h l + t_w m.$$

- This is identical to packet routing, however, t_w is typically much smaller.

Message Passing Costs in Parallel Computers

(a) through a store-and-forward communication network;

b) and (c) extending the concept to cut-through routing.



(a) A single message sent over a store-and-forward network

(b) The same message broken into two parts and sent over the network.

(c) The same message broken into four parts and sent over the network.

Message Passing Costs in Parallel Computers

Simplified Cost Model for Communicating Messages

- The cost of communicating a message between two nodes l hops away using cut-through routing is given by

$$t_{comm} = t_s + lt_h + t_w m.$$

- In this expression, t_h is typically smaller than t_s and t_w. For this reason, the second term in the RHS does not show, particularly, when m is large.

- For these reasons, we can approximate the cost of message transfer by

$$t_{comm} = t_s + t_w m.$$

Message Passing Costs in Parallel Computers

Simplified Cost Model for Communicating Messages

- It is important to note that the original expression for communication time is valid for only uncongested networks.

- Different communication patterns congest different networks to varying extents.

- It is important to understand and account for this in the communication time accordingly.

Questions

References

1. Flynn, M., “Some Computer Organizations and Their Effectiveness,” IEEE Transactions on Computers, Vol. C-21, No. 9, September 1972.
2. Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). *Introduction to parallel computing* (Vol. 110). Redwood City, CA: Benjamin/Cummings.
3. Quinn, M. J. Parallel Programming in C with MPI and OpenMP,(2003).