



Lecture 2

Hardware Architectures

Institute of Computer Science & Information Technology,
Faculty of Management & Computer Sciences,
The University of Agriculture, Peshawar, Pakistan.

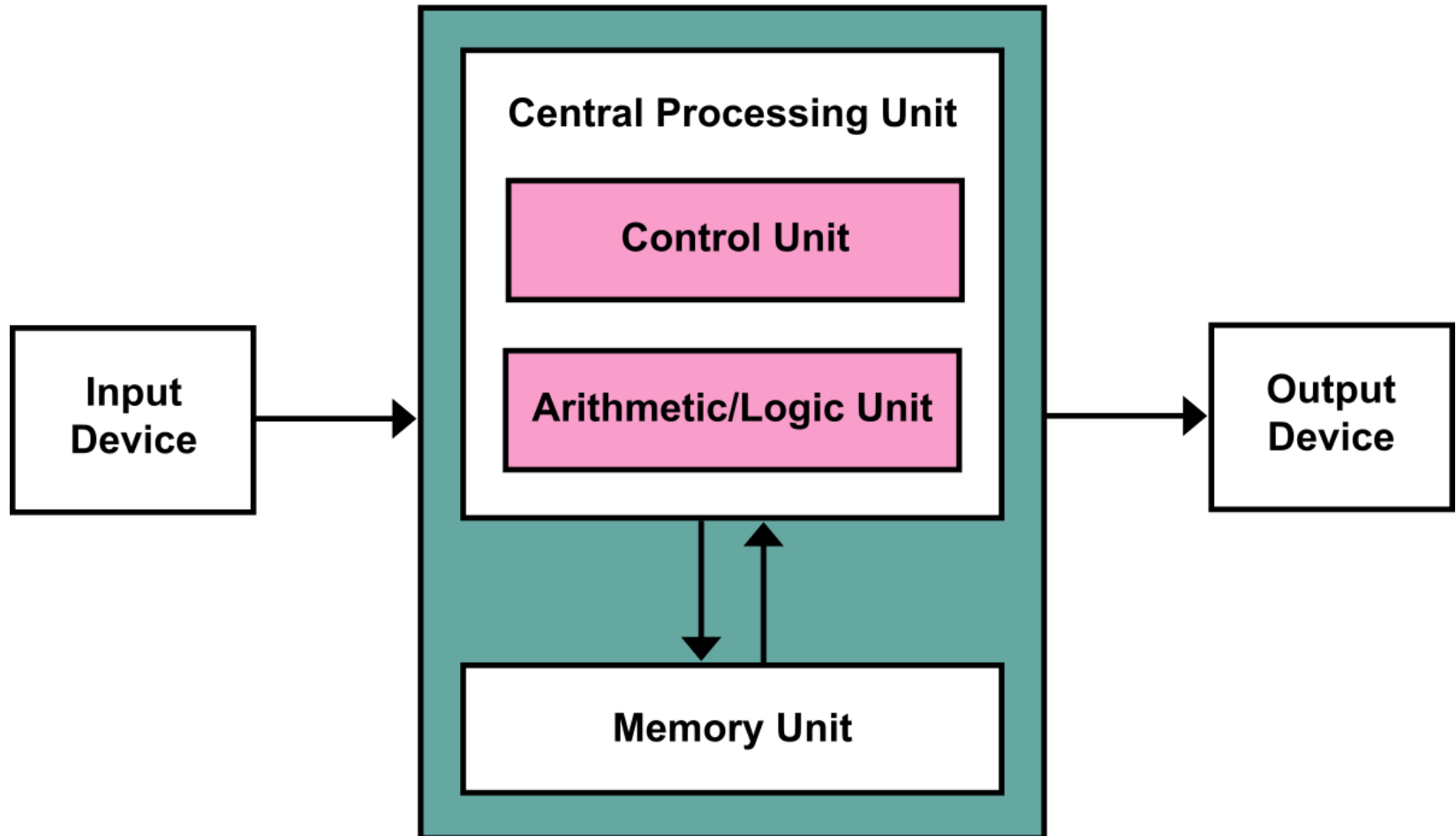
Hardware Architectures

- It refers to the identification of system's **physical components** and their **interrelationships**.
- This description, often called a **hardware design model**, allows hardware designers to understand how their components fit into a system architecture and provides to software component designers important information needed for software development and integration.

Hardware Architectures(I)

- It is set of rules and methods that describe the functionality, organization and implementation of computer.
- It is also concerned with balancing the performance, efficiency, cost and reliability of a computer system.

Hardware Architectures(2)



Flynn's Taxonomy

- Micheal J. Flynn (born May 20, 1934) is an American Professor at Stanford university.
- He proposed the Flynn's taxonomy, a method of classifying digital computer architectures, in 1966.
- Two types of Information flow into a processor:
 1. Instruction Stream: The sequence of **instructions** from **memory** to the control unit.
 2. Data Stream: The sequence of **data** from **memory** to the control unit.

Flynn's Classification

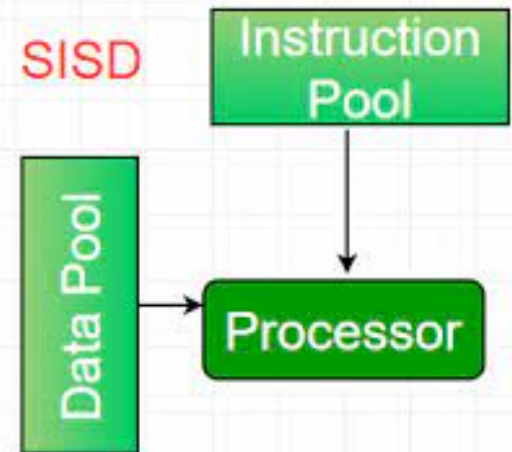
- Number of concurrent instructions (or control) streams and data stream available in the architecture can be single or multiple.
 1. Single Instruction Stream, Single Data Stream (SISD)
 2. Single Instruction Stream, Multiple Data Streams (SIMD)
 3. Multiple Instruction Streams, Single Data Stream (SISD)
 4. Multiple Instruction Streams, Multiple Data Streams (MIMD)

Flynn's Classification(I)

		Instruction Streams	
		one	many
Data Streams	one	SISD traditional von Neumann single CPU computer	MISD May be pipelined Computers
	many	SIMD Vector processors fine grained data Parallel computers	MIMD Multi computers Multiprocessors

Single-Instruction Single Data (SISD)

- It is uni-processor machine which is capable of executing a single instruction, operating on single data stream.
- Machine instructions are processed in a sequential manner in which single control unit fetches single instruction from memory.
- The control unit then generate appropriate control signals to direct single processing element to operate on single data steam i.e., one operation at a time
- The speed of the processing element in the SISD model is limited by the rate at which the computer can transfer information internally.

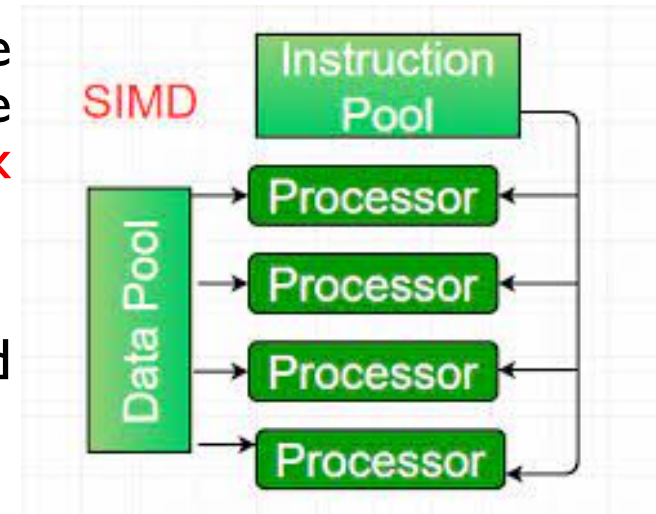


Single-Instruction Single Data (SISD)(I)

- It require less **power**
- There is no issue of **complex communication protocol** between multiple cores.
- The **speed** of SISD architecture **is limited** just like single-core processors.
- Dominant representative of SISD systems are IBM PC, Workstations

Single-Instruction Multiple Data (SIMD)

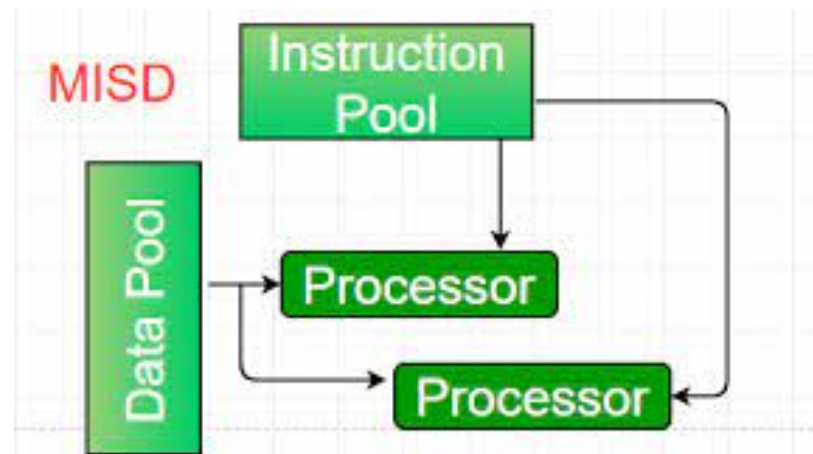
- It is a **multiprocessor** machine capable of executing the same instruction on all the CPUs but operating on **different data streams**.
- Machines based on an SIMD model are suited to **scientific computing** since they involve lots of **vector and matrix operations**.
- So that the information can be passed to all the Processing Elements (PE).
- Organized data elements of vectors can be divided into multiple sets and each PE can process one data set.



Single-Instruction Multiple Data (SIMD)(I)

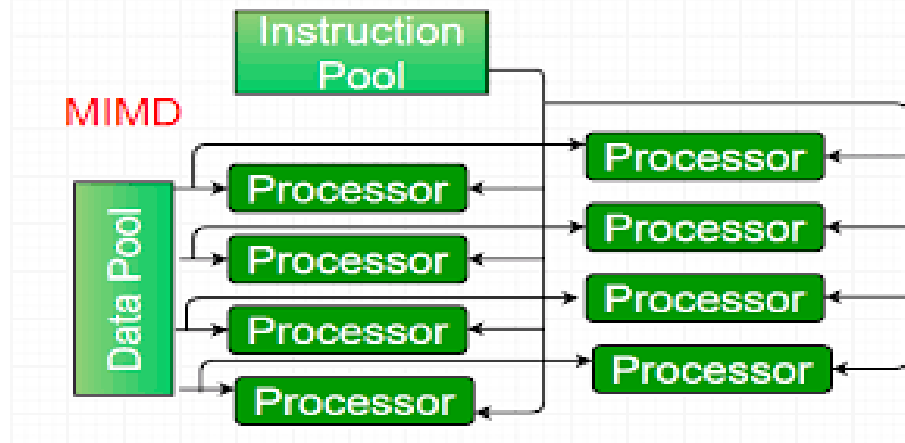
- Instructions can be executed **sequentially**, such as by **pipelining**, or in **parallel** by multiple functional units.
- Throughput of the system can be **increased** by increasing the **number of cores** of the processor.
- Processing **speed** is **higher** than SISD architecture.
- There is complex communication between number of cores of processor.

Multiple-Instruction Single Data (MISD)



- In this computing, microprocessor is capable of executing **different instructions** on PE but all of them **operating** on the **one data stream**.
- It has number of processing units performing **different operations** by executing **different instruction** on the same dataset.
- This model is not useful in general for most of the applications.

Multiple-Instruction Multiple Data (MIMD)

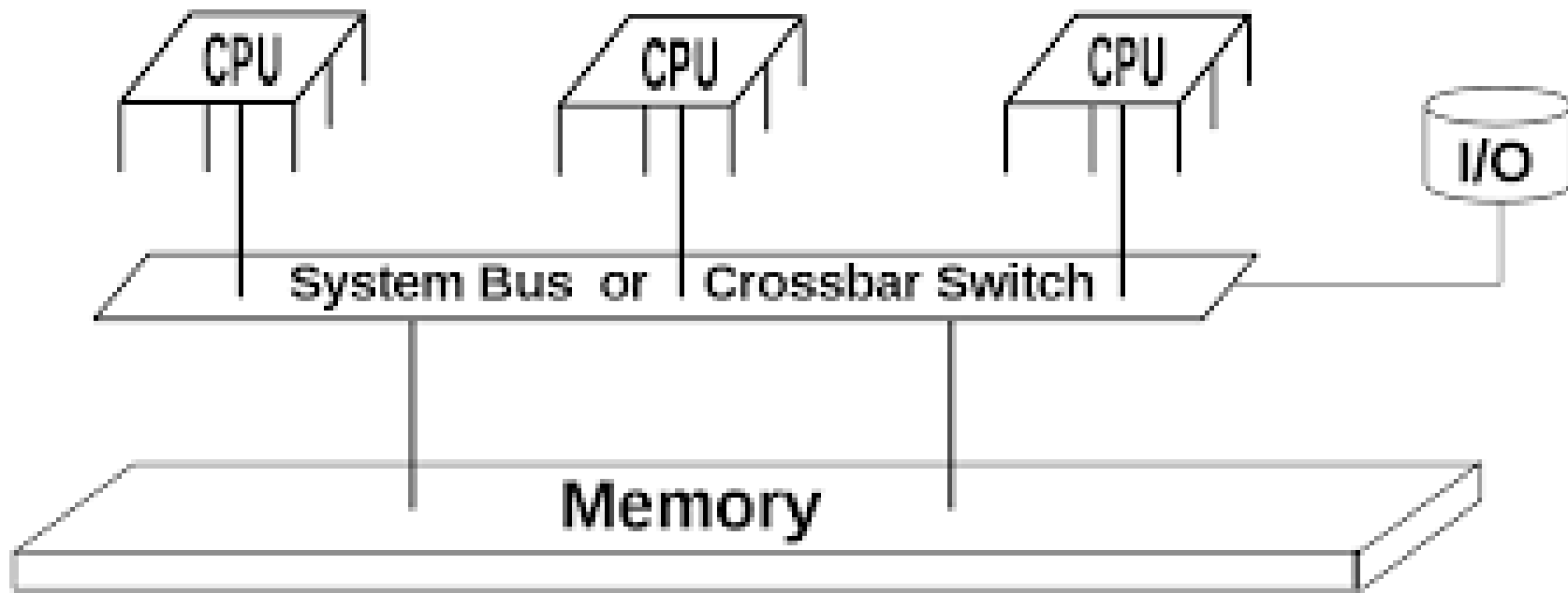


- This is model used for **multiprocessor**, which is capable of executing **multiple instructions** on **multiple data sets**.
- Each PE in the MIMD has separate **instructions** and **data streams**; therefore machines built using this model are capable to any kind of applications.
- Unlike SIMD and MISD machines, PEs in MIMD machines work **asynchronously**.

Shared Memory

- It is a memory that simultaneously accessed by multiple programs with an intent to **provide communication among them** or **avoid redundant copies**.
- Shared memory is an efficient means of passing data between programs.
- Depending on context, programs may run on **single** processor or on **multiple** separate processor.

Shared Memory(I)



Shared Memory(2)

- The concept of shared memory not only existed at hardware but also in software.
- In computer hardware, shared memory refers to a block of **Random Access Memory(RAM)** that can be accessed by several different **CPUs** in a **multiprocessor** systems.

Shared Memory(3)

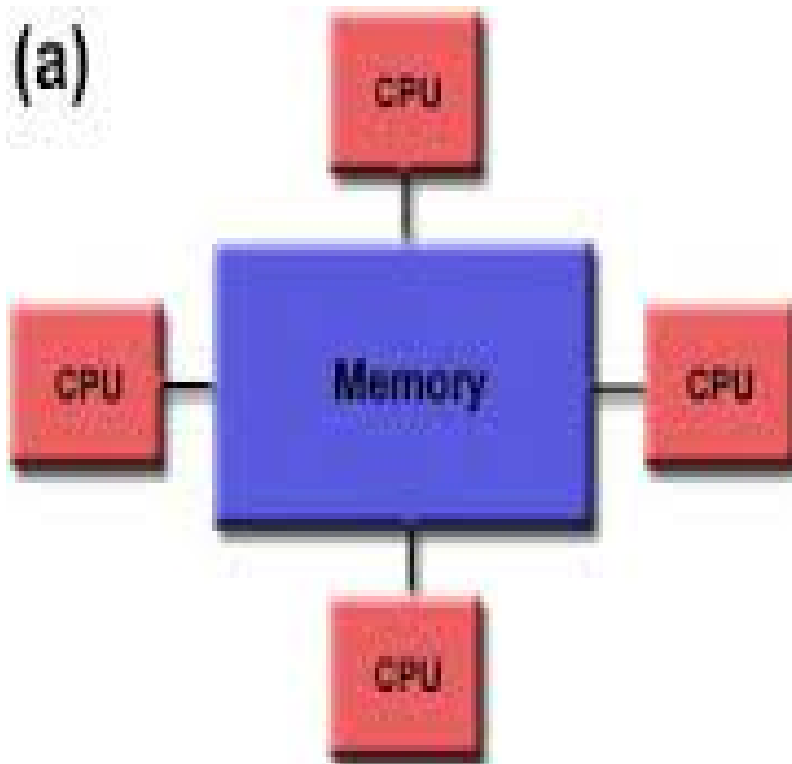
- In shared memory systems following different approaches may followed:

1. **Uniform Memory Access (UMA):** All the processors share the **physical** memory **uniformly** (e.g., access time is same etc.)

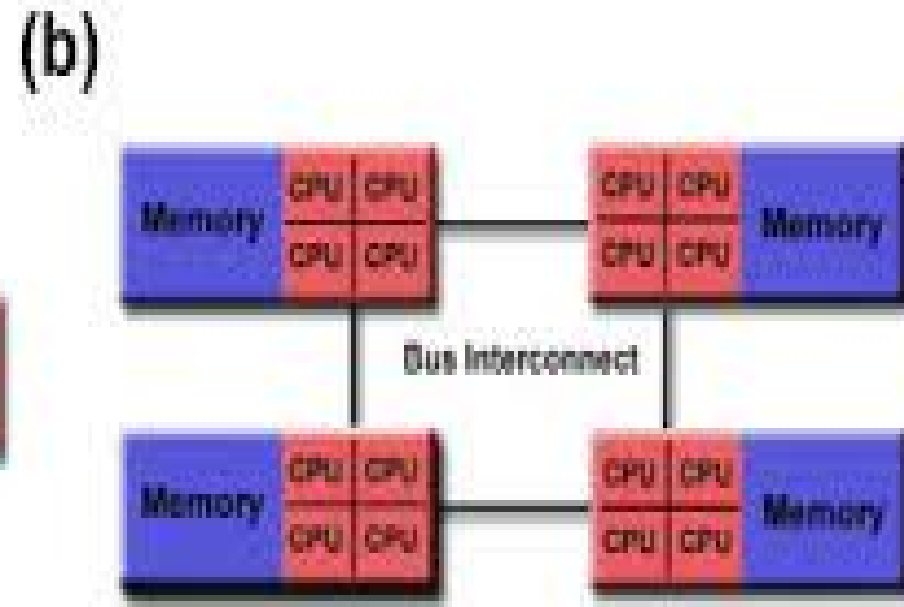
2. **Non-Uniform Memory Access (NUMA):** memory access time **depends on the memory location** relative to a processor;

3. **Cache-only Memory Architecture (COMA):** the **local memories** for the processor at each node is used as cache instead of as actual main memory.

Shared Memory(4)



Uniform Memory Access (UMA)



Non-Uniform Memory Access (NUMA)

Advantage of Shared Memory

- Shared memory system is relatively easy to program since all processors share a **single view** of data and the communication between processors can be as **fast as memory accesses** to a same location.

Challenge in Shared Memory

- The issue with shared memory systems is that **many CPUs need fast access to memory** and will likely cache memory, which have **two complications**:
 - I. **Access Time Degradation**: When several processors try to access same memory location it causes contention

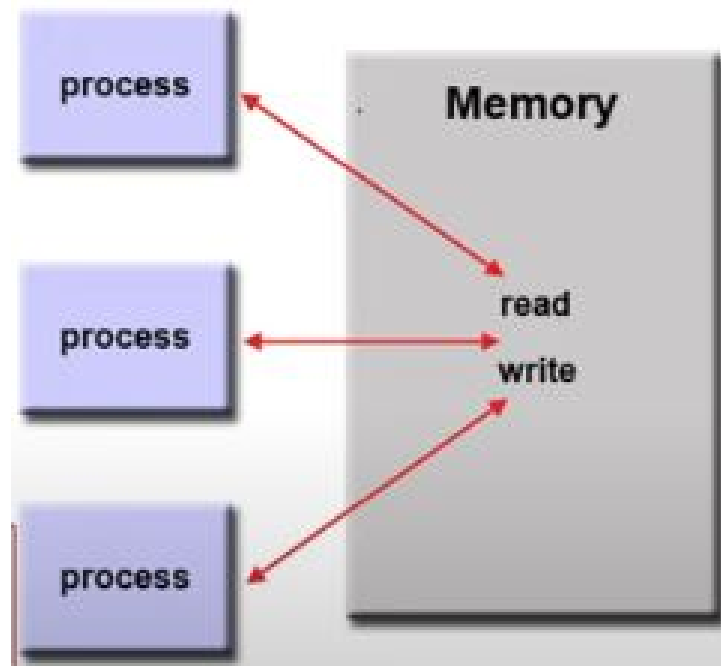
Challenge in Shared Memory (I)

- Various mechanisms can be used to aid the access time degradation:

1. Locks: Two values of lock can be possible, either 0 or 1. A process want to be in the **critical section** first check the lock variable.

2. Semaphores used for the synchronization of **concurrent processes**.

3. Mutex is a locking mechanism used to synchronize access to a resources. Only one task can acquire mutex.




Challenge in Shared Memory (2)

Lack of Data Coherence:

- Whenever one cache is updated with information that may be used by other processors, the change needs to be reflected to the processors, otherwise the different processor will be working with incoherent data.

Shared Memory (Software Level)

- At software level, shared memory can be managed by either:
 1. A method of **Inter-Process Communication (IPC)**, i.e., a way of exchanging data between programs running at the same time.
 2. A method of **conserving memory space** by directing accesses to what would ordinarily be copies of a piece of data to a single instance instead, by using virtual memory mappings or with explicit support of the program in question.

- 
- Since both process can access the shared memory area like regular working memory, this is a very fast way of communication.
 - On the other hand, it is less scalable, and care must be taken to avoid issues if processes sharing memory are running on separate CPUs and the underlying architecture is not cache coherent.