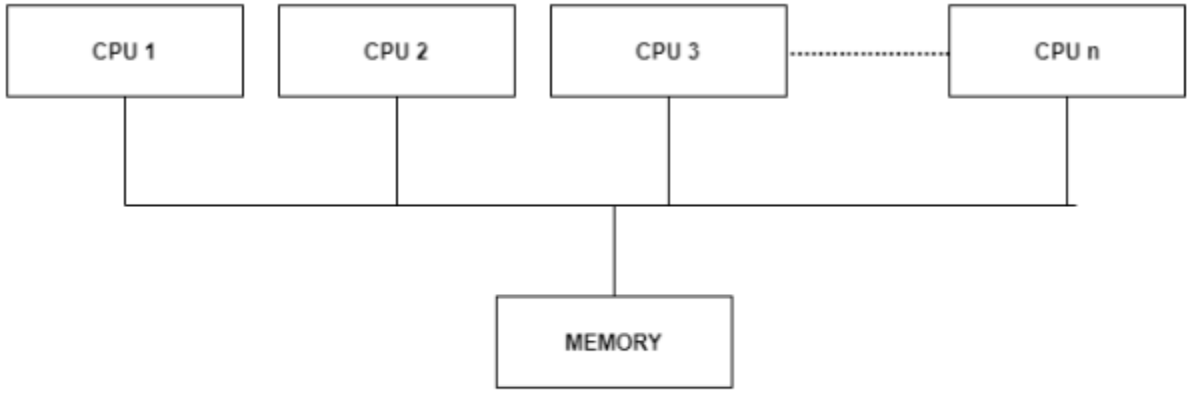
Multiprocessors

A shared-memory multiprocessor (or just multiprocessor henceforth) is a computer system in which two or more CPUs share full access to a common RAM. A program running on any of the CPUs sees a normal (usually paged) virtual address space. The only unusual property this system has is that the CPU can write some value into a memory word and then read the word back and get a different value (because another CPU has changed it).



Multiprocessing Architecture

Types of Multiprocessors

There are mainly two types of multiprocessors i.e. symmetric and asymmetric multiprocessors. Details about them are as follows:

- **Symmetric Multiprocessors**

➢In these types of systems, **each processor contains a similar copy of the operating system and they all communicate with each other**. All the processors are in a peer to peer relationship i.e. **no master - slave relationship** exists between them.

➢An example of the symmetric multiprocessing system is the Encore version of Unix for the Multimax Computer.

- **Asymmetric Multiprocessors**

➢In asymmetric systems, each processor is given a predefined task. There is a **master processor that gives instruction to all the other processors**. Asymmetric multiprocessor system **contains a master slave relationship**.

➢Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created. Now also, this is the cheaper option.

Advantages of Multiprocessor Systems

There are multiple advantages to multiprocessor systems. Some of these are:

- **More reliable Systems** In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example: If there are 5 processors in a multiprocessor system and one of them fails, then also 4 processors are still working. So the system only becomes slower and does not ground to a halt.
- **Enhanced Throughput** If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.
- **More Economic Systems** Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than have different computer systems with multiple copies of the data.

Disadvantages of Multiprocessor Systems

There are some disadvantages as well to multiprocessor systems. Some of these are:

- **Increased Expense** Even though multiprocessor systems are cheaper in the long run than using multiple computer systems, still they are quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.
- **Complicated Operating System Required** There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart resources to processes, than in single processor systems. Hence, a more complex and complicated operating system is required in multiprocessor systems.
- **Large Main Memory Required** All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

Coupling of Processors

Tightly Coupled System

- Tasks and/or processors communicate in a highly synchronized fashion
- Communicates through a common shared memory
- Shared memory system

Loosely Coupled System

- Tasks or processors do not communicate in a synchronized fashion
- Communicates by message passing packets
- Overhead for data exchange is high
- Distributed memory system

Interconnection Structure

The **components that form a multiprocessor system are CPUs, IOPs connected to input-output devices, and a memory unit**.

The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available

o Between the processors and memory in a shared memory system

o Among the processing elements in a loosely coupled system

There are several physical forms available for establishing an interconnection network.

o Time-shared common bus

o Multiport memory

o Crossbar switch

o Multistage switching network : These composed of processing elements (PEs) on one end of the network and memory elements (MEs) on the other end, connected by switching elements (SEs).

o Hypercube system : Nodes of one cube are connected with nodes of other cube

Time Shared Common Bus

A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit.

o Only one processor can communicate with the memory or another processor at any given time.

o As a consequence, the total overall transfer rate within the system is limited by the speed of the single path

A more economical implementation of a dual bus structure is depicted in Fig. below.

Part of the local memory may be designed as a *cache memory attached to the CPU.*

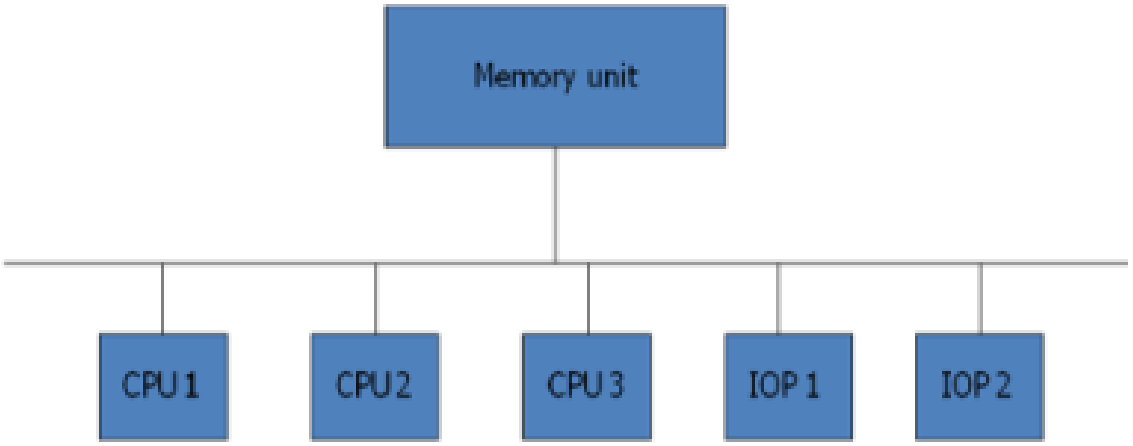


Fig: Time shared common bus organization

Multiport Memory

A multiport memory system employs separate buses between each memory module and each CPU.

The module must have **internal control logic to determine which port will have access to memory at any given time**.

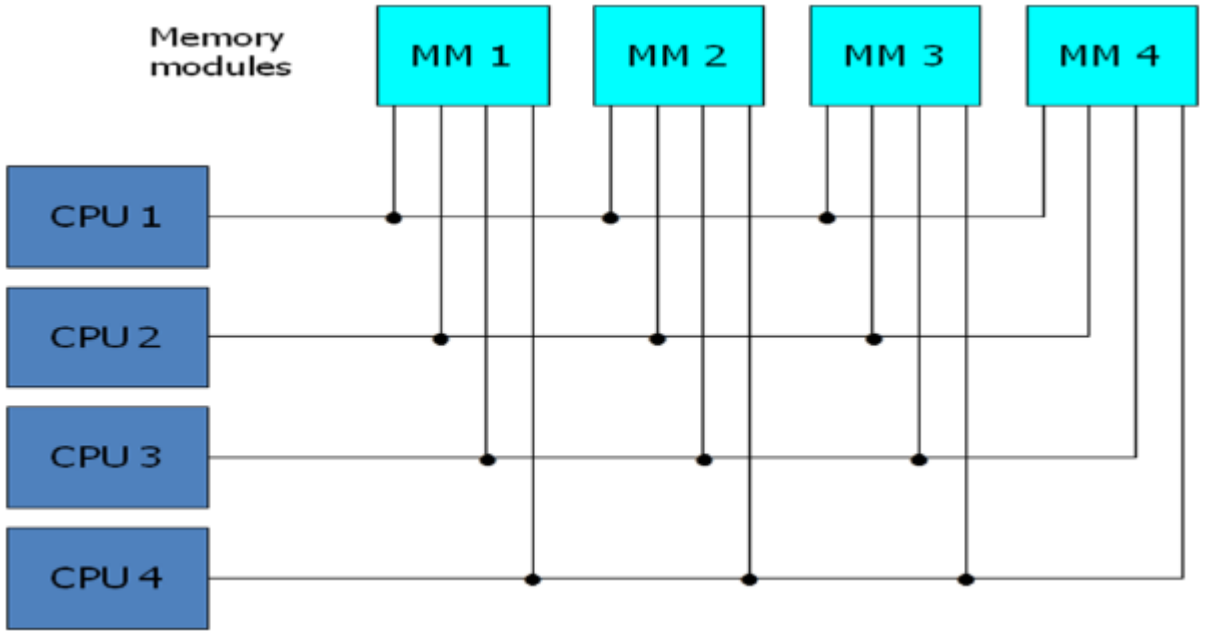
Memory access conflicts are resolved by assigning fixed priorities to each memory port.

Adv.:

o The high transfer rate can be achieved because of the multiple paths.

Disadv.:

o It requires expensive memory control logic and a large number of cables and connections



Cross Bar Switch

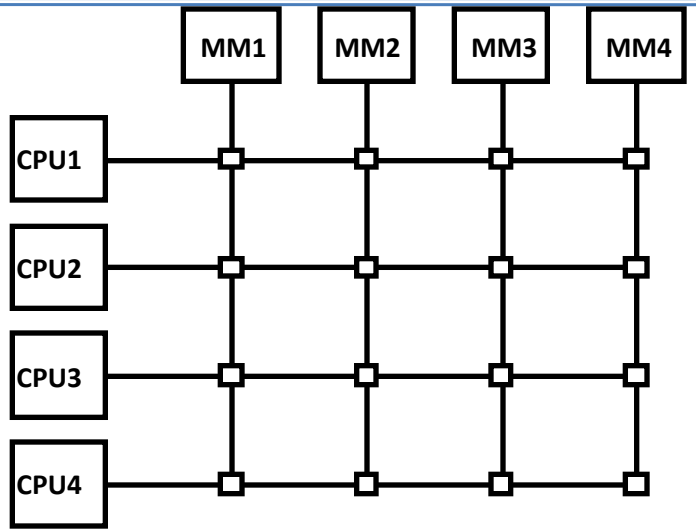


fig. shows the functional design of a crossbar switch connected to one memory module.

Consists of a number of *crosspoints that are placed at intersections between processor buses and memory module paths.*

The small square in each crosspoint is a *switch that determines the path from a processor to a memory module.*

Adv.:

o Supports simultaneous transfers from all memory modules

Disadv.:

o The hardware required to implement the switch can become quite large and complex.

Multiprocessor architectures

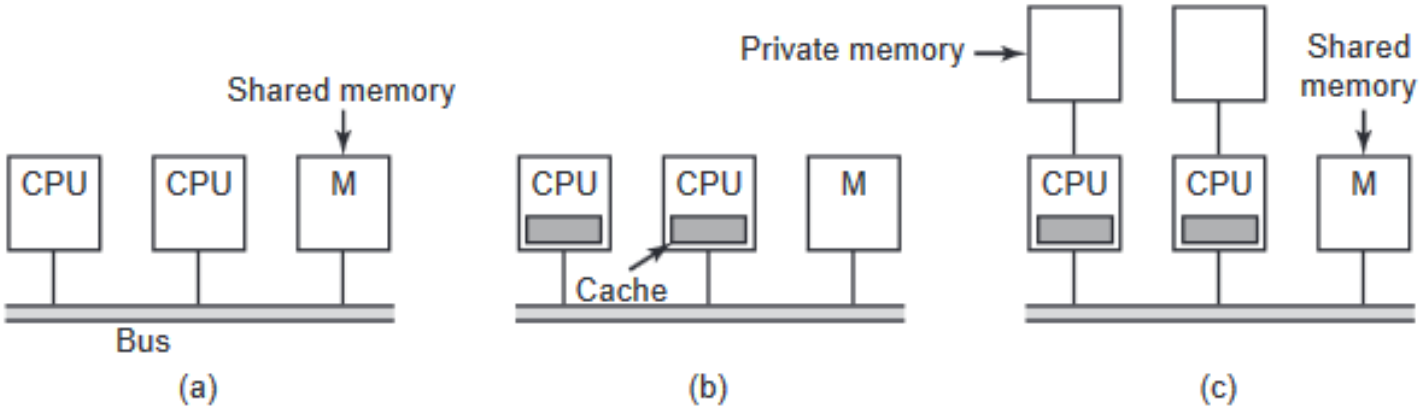


Figure 8-1. Three bus-based multiprocessors. (a) Without caching. (b) With caching. (c) With caching and private memories.

(a) Two or more CPUs and one or more memory modules all use the same bus for communication. When a CPU wants to read a memory word, it first checks to see if the bus is busy. If the bus is idle, the CPU puts the address of the word it wants on the bus, asserts a few control signals, and waits until the memory puts the desired word on the bus.

(b) Problem to previous approach, If the bus is busy when a CPU wants to read or write memory, the CPU just waits until the bus becomes idle. The cache can be inside the CPU chip, next to the CPU chip, on the processor board, or some combination of all three. Since many reads can now be satisfied out of the local cache, there will be much less bus traffic, and the system can support more CPUs.

(c) in this each CPU has not only a cache, but also a local, private memory which it accesses over a dedicated (private) bus. To use this configuration optimally, the compiler should place all the program text, strings, constants and other read-only data, stacks, and local variables in the private memories. The shared memory is then only used for writable shared variables.

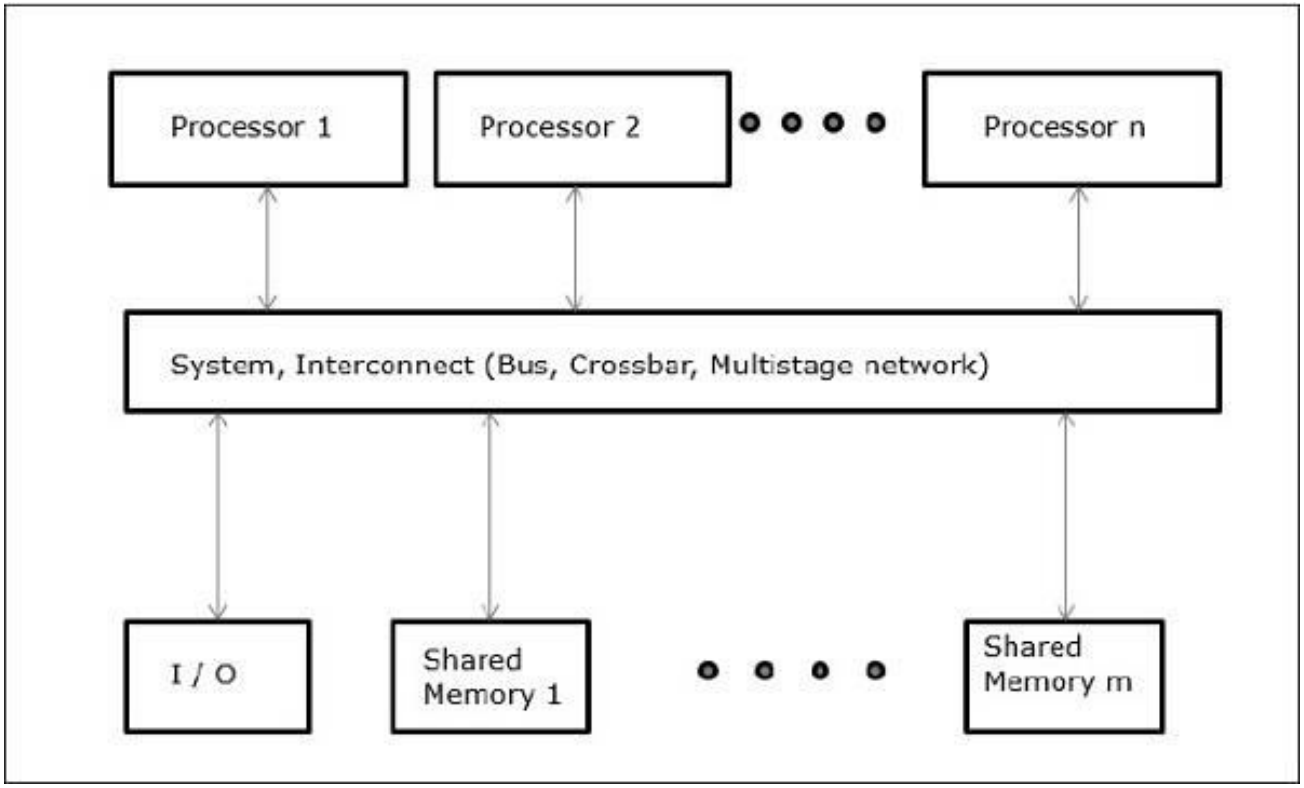
Shared-Memory Multicomputers

Three most common shared memory multiprocessors models are −

Uniform Memory Access (UMA)

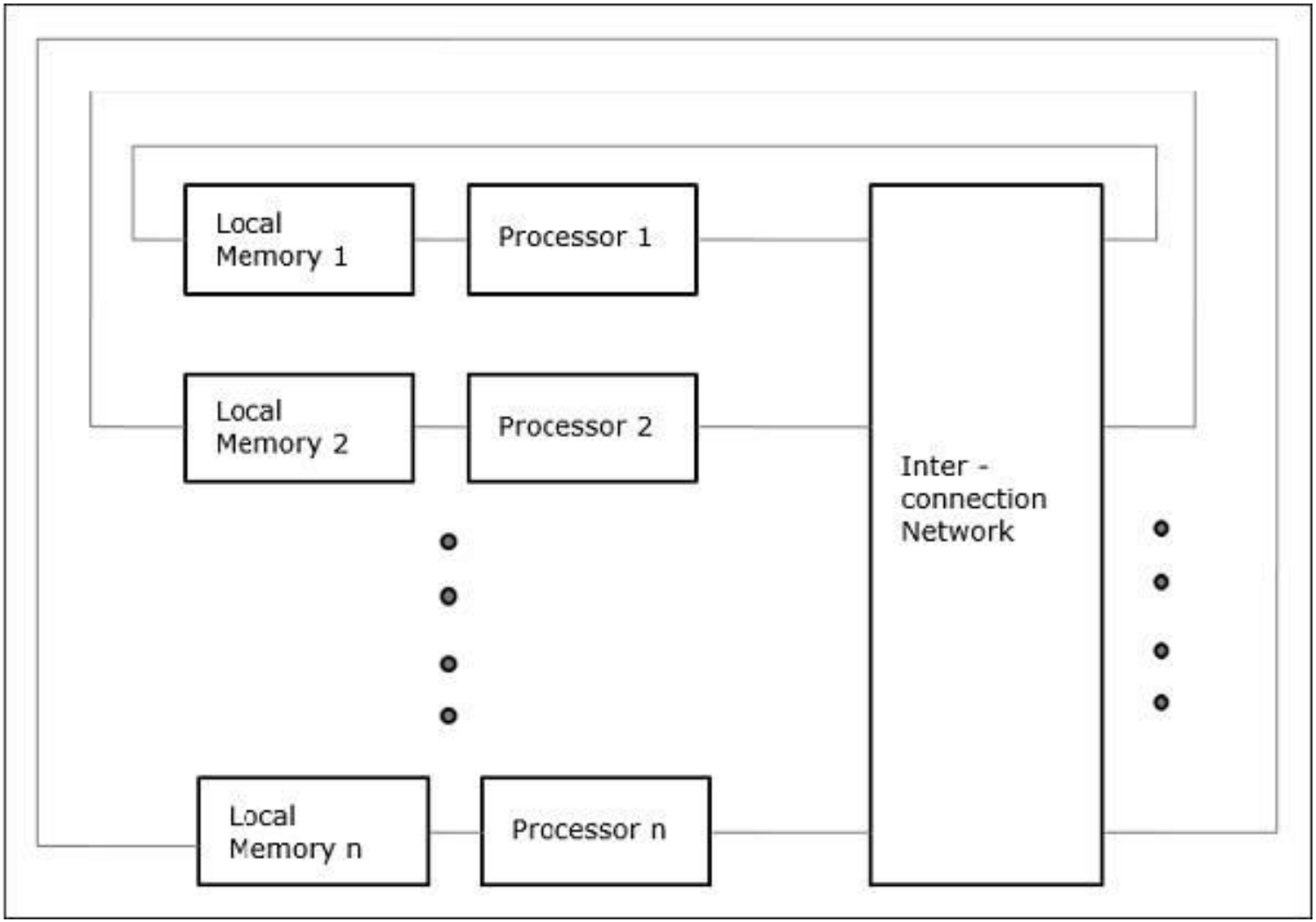
In this model, all the processors share the physical memory uniformly. All the processors have equal access time to all the memory words. Each processor may have a private cache memory. Same rule is followed for peripheral devices.

When all the processors have equal access to all the peripheral devices, the system is called a **symmetric multiprocessor**. When only one or a few processors can access the peripheral devices, the system is called an **asymmetric multiprocessor**.



Non-uniform Memory Access (NUMA)

In NUMA multiprocessor model, the access time varies with the location of the memory word. Here, the shared memory is physically distributed among all the processors, called local memories. The collection of all local memories forms a global address space which can be accessed by all the processors.



Key Differences Between UMA and NUMA

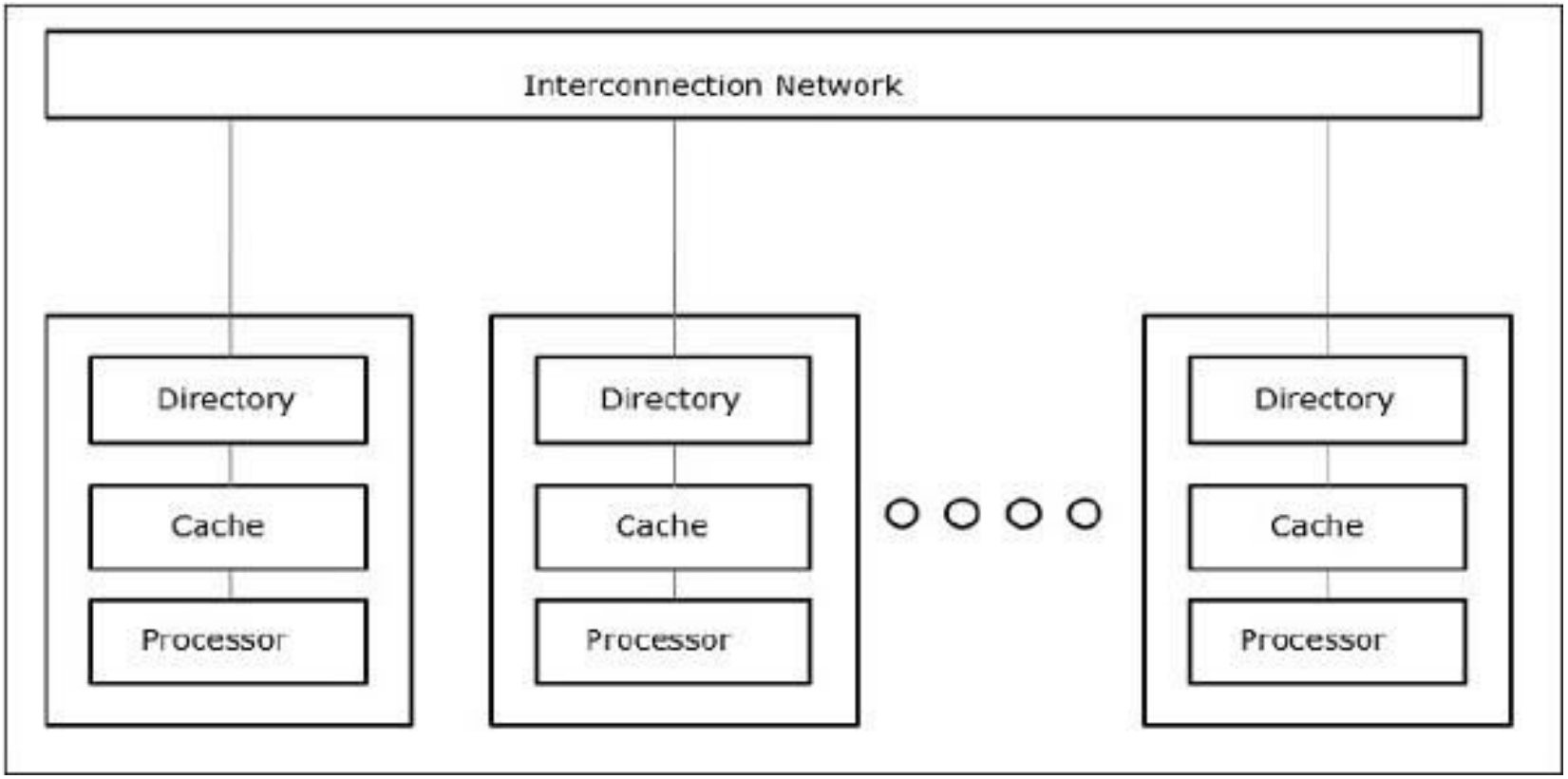
- The UMA (shared memory) model uses one or two memory controllers. As against, NUMA can have multiple memory controllers to access the memory.
- Single, multiple and crossbar busses are used in UMA architecture. Conversely, NUMA uses hierarchical, and tree type of busses and network connection.
- In UMA the memory accessing time for each processor is the same while in NUMA the memory accessing time changes as the distance of memory from the processor changes.
- General purpose and time-sharing applications are suitable for the UMA machines. In contrast, the appropriate application for NUMA is real-time and time-critical centric.
- The UMA based parallel systems work slower than the NUMA systems.
- When it comes to bandwidth UMA, have limited bandwidth. On the contrary, NUMA has bandwidth more than UMA.

Comparison Chart

Basis for comparison	UMA	NUMA
Basic	Uses a single memory controller	Multiple memory controller
Type of buses used	Single, multiple and crossbar.	Tree and hierarchical
Memory accessing time	Equal	Changes according to the distance of microprocessor.
Suitable for	General purpose and time-sharing applications	Real-time and time-critical applications
Speed	Slower	Faster
Bandwidth	Limited	More than UMA.

Cache Only Memory Architecture (COMA)

The COMA model is a special case of the NUMA model. Here, all the distributed main memories are converted to cache memories.



Parallel Processing

Execution of *Concurrent Events* in the computing process to achieve faster *Computational Speed*

- **The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, i.e. the amount of processing that can be accomplished during a given interval of time**

Levels of Parallel Processing

- **Job or Program level**
- **Task or Procedure level**
- **Inter-Instruction level**
- **Intra-Instruction level**

Lowest level : shift register, register with parallel load
Higher level : multiplicity of functional unit that perform identical /different task

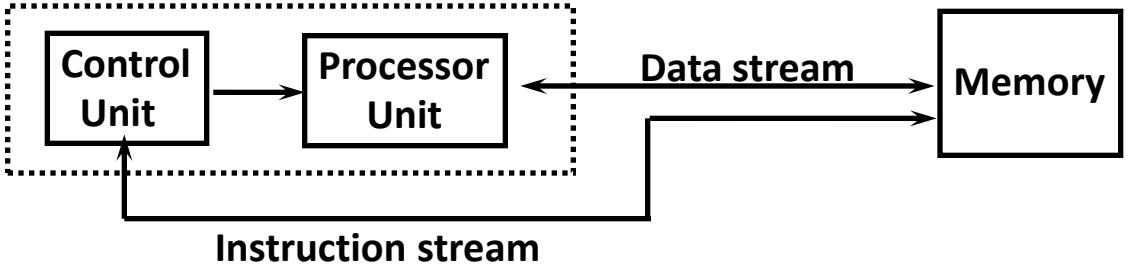
Parallel Computers

Architectural Classification

- **Flynn's classification**
  - **Based on the multiplicity of *Instruction Streams* and *Data Streams***
  - **Instruction Stream**
    - **Sequence of Instructions read from memory**
  - **Data Stream**
    - **Operations performed on the data in the processor**

		Number of *Data Streams*	
		Single	**Multiple**
Number of *Instruction Streams*	**Single**	**SISD**	**SIMD**
	Multiple	**MISD**	**MIMD**

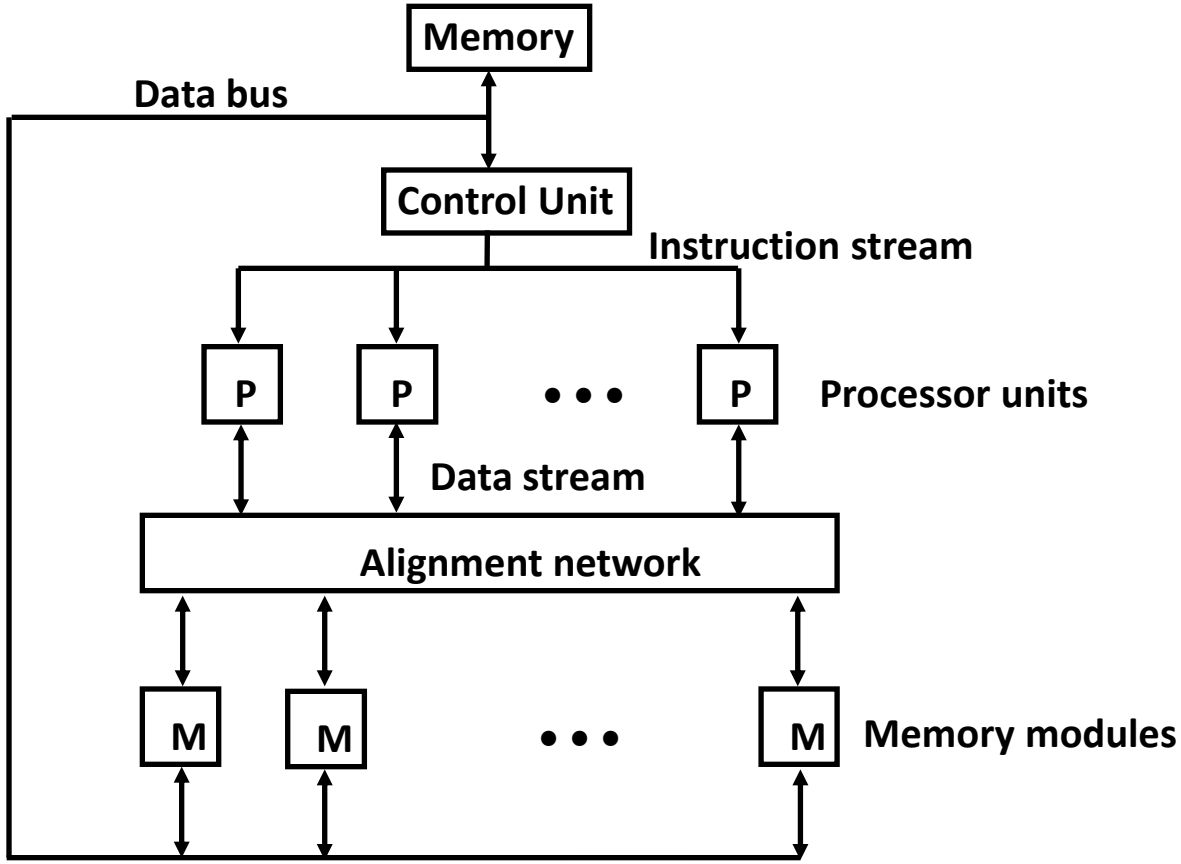
SISD



Characteristics

- **Single computer containing a control unit, processor and memory unit**
- **Instructions and data are stored in memory and executed sequentially**
- **may or may not have parallel processing**
- **parallel processing can be achieved by pipelining**

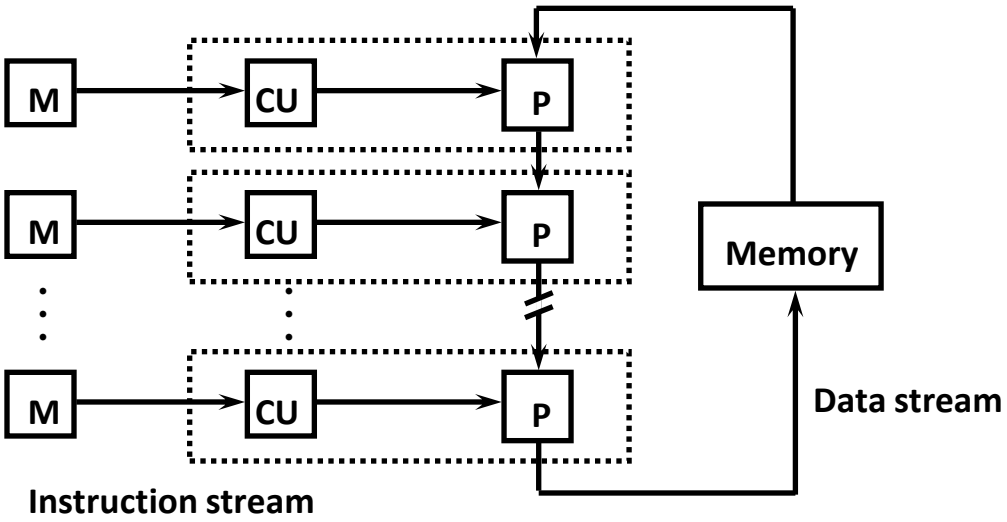
SIMD



Characteristics

- Only one copy of the program exists
- A single controller executes one instruction at a time

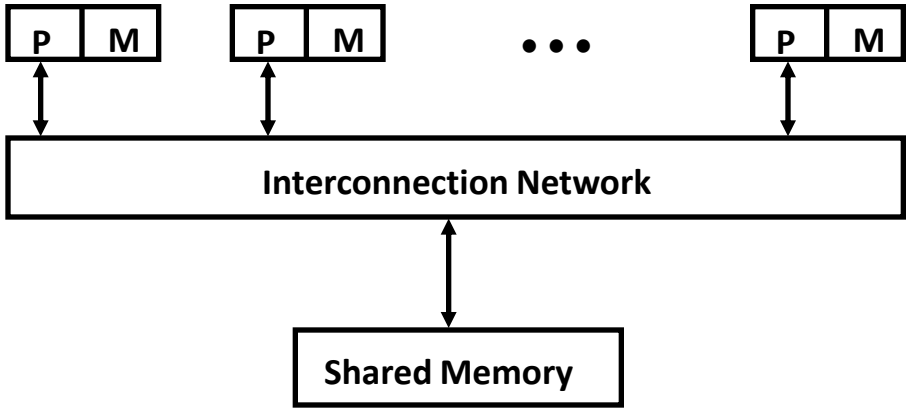
MISD



Characteristics

- There is no computer at present that can be classified as MISD

MIMD



Characteristics

- **Multiple processing units**
- **Execution of multiple instructions on multiple data**

Types of MIMD computer systems

- **Shared memory multiprocessors**
- **Message-passing multicomputers**

Pipelining

A technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.

- It is the characteristic of pipelining that several computations can be in progress in distinct segments at the same time.

- Each segment performs partial processing dictated by the way the task is dictated

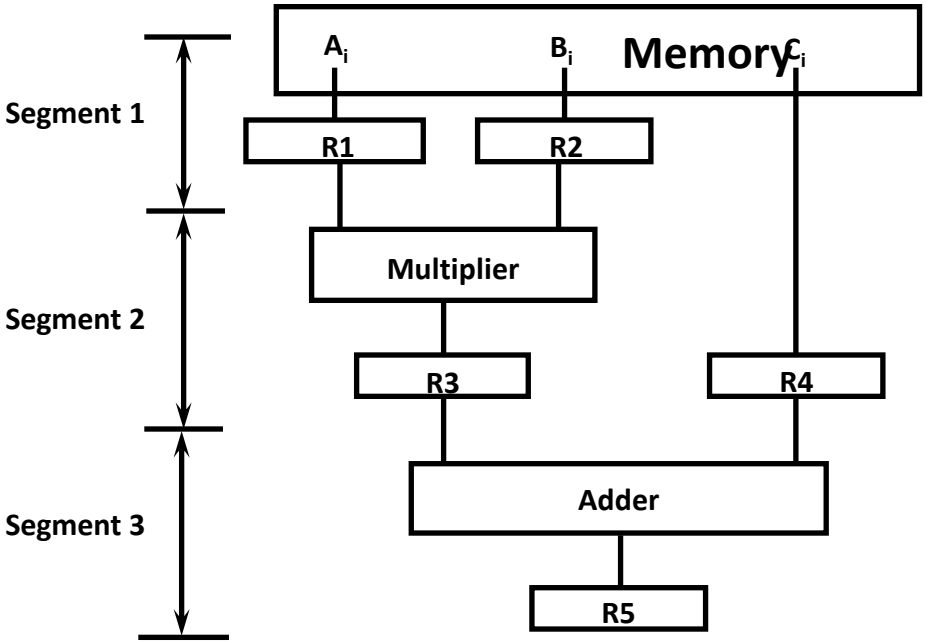
- The result obtained from computation is in each segment is transferred to next segment in the pipeline

- The final result is obtained after data has been passed through all segment

Pipelining

Simplest way to understand pipelining is to imagine that each segment consist of input register followed by combinational circuit. The o/p of combinational circuit in a segment is applied to i/p register of next segment

$A_i * B_i + C_i$ for i = 1, 2, 3, ... , 7



$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Load A_i and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and load C_i
$R5 \leftarrow R3 + R4$	Add

Design of a basic pipeline

•In a pipelined processor, a pipeline has two ends, the input end and the output end. Between these ends, there are multiple stages/segments such that output of one stage is connected to input of next stage and each stage performs a specific operation.

•Interface registers are used to hold the intermediate output between two stages. These interface registers are also called latch or buffer.

•All the stages in the pipeline along with the interface registers are controlled by a common clock.

Pipeline Stages

RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of RISC pipeline with their respective operations:

Stage 1 (Instruction Fetch)

In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

Stage 2 (Instruction Decode)

In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

Stage 3 (Instruction Execute)

In this stage, ALU operations are performed.

Stage 4 (Memory Access)

In this stage, memory operands are read and written from/to the memory that is present in the instruction.

Stage 5 (Write Back)

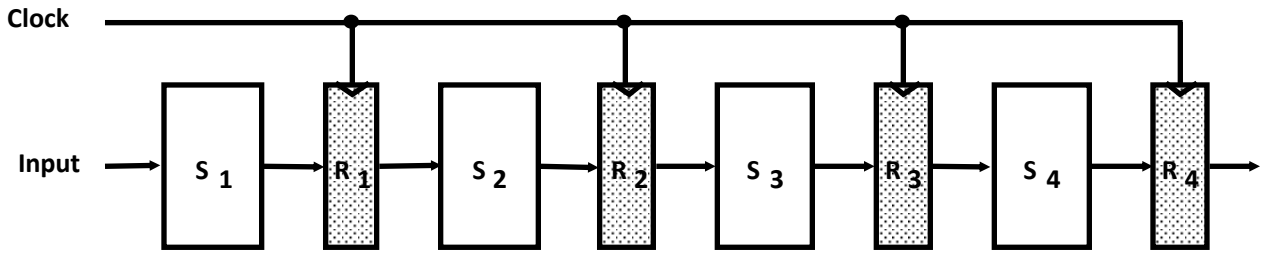
In this stage, computed/fetched value is written back to the register present in the instruction.

Operations in each Pipeline Stage

Clock Pulse Number	Segment 1		Segment 2			Segment 3
	R1	R2	R3	R4		R5
1	A1	B1				
2	A2	B2	A1 * B1	C1		
3	A3	B3	A2 * B2	C2	A1	* B1 + C1
4	A4	B4	A3 * B3	C3	A2	* B2 + C2
5	A5	B5	A4 * B4	C4	A3	* B3 + C3
6	A6	B6	A5 * B5	C5	A4	* B4 + C4
7	A7	B7	A6 * B6	C6	A5	* B5 + C5
8			A7 * B7	C7		A6 * B6 + C6
9						A7 * B7 + C7

General Pipeline

General Structure of a 4-Segment Pipeline



Space-Time Diagram

Segment	1	2	3	4	5	6	7	8	9
1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

Clock cycles

Pipeline SpeedUp

n: Number of tasks to be performed

Conventional Machine (Non-Pipelined)

t_n: Clock cycle (time to complete each task)

τ_1: Time required to complete the n tasks

$$\tau_1 = n * t_n$$

Pipelined Machine (k stages)

t_p: Clock cycle (time to complete each suboperation)

τ_k: Time required to complete the n tasks

$$\tau_k = (k + n - 1) * t_p$$

Speedup

S_k: Speedup

$$S_k = n*t_n / (k + n - 1)*t_p$$

Pipeline SpeedUp

As n becomes very larger than k-1 then k+n-1 approaches to n

Then : $S = t_n/t_p$

If we consider time taken to complete a task is same in both circuits then $t_n = kt_p$ and speedup reduces to

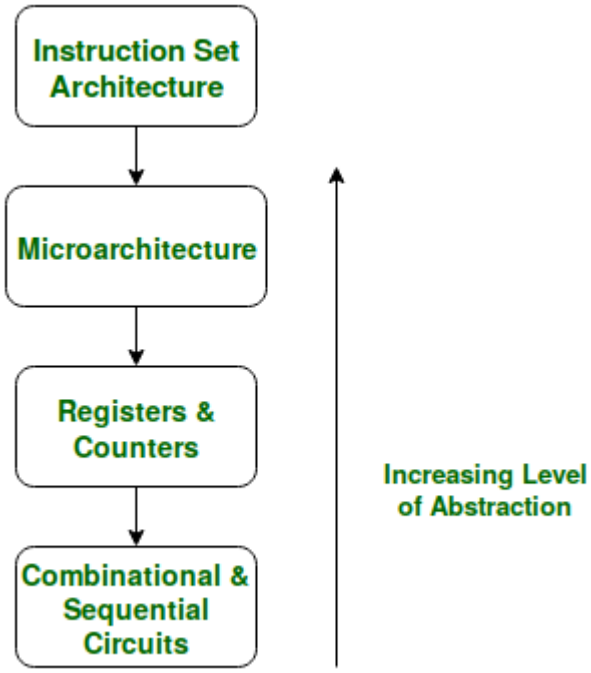
$S = kt_p/t_n = k$

i.e. maximum theoritical speedup pipeline can provide is k.

Micro-architecture

An **ISA** describes the **design of a Computer** in terms of the **basic operations** it must support. The ISA is not concerned with the implementation specific details of a computer. It is only concerned with the set or collection of basic operations the computer must support. For example the AMD Athlon and the Core 2 Duo processors have entirely different implementations but they support more or less the same set of basic operations as defined in the x86 Instruction Set.

Micro architectural level lies just below the **ISA** level and hence is concerned with the implementation of the basic operations to be supported by the Computer as defined by the **ISA**. Therefore we can say that the AMD Athlon and Core 2 Duo processors are based on the same ISA but have different microarchitectures with different performance and efficiencies.



Graphics Processing Unit

•Like a motherboard, a graphics card is a printed circuit board that houses a processor and RAM. It also has an input/output system (BIOS) chip, which stores the card's settings and performs diagnostics on the memory, input and output at startup.

•A graphics card's processor, called a graphics processing unit (GPU), is similar to a computer's CPU. A GPU, however, is designed specifically for performing the complex mathematical and geometric calculations that are necessary for graphics rendering. Some of the fastest GPUs have more transistors than the average CPU. A GPU produces a lot of heat, so it is usually located under a heat sink or a fan.

•A GPU however is more dedicated in function. It takes that same function that a CPU was processing and completes it all at once.

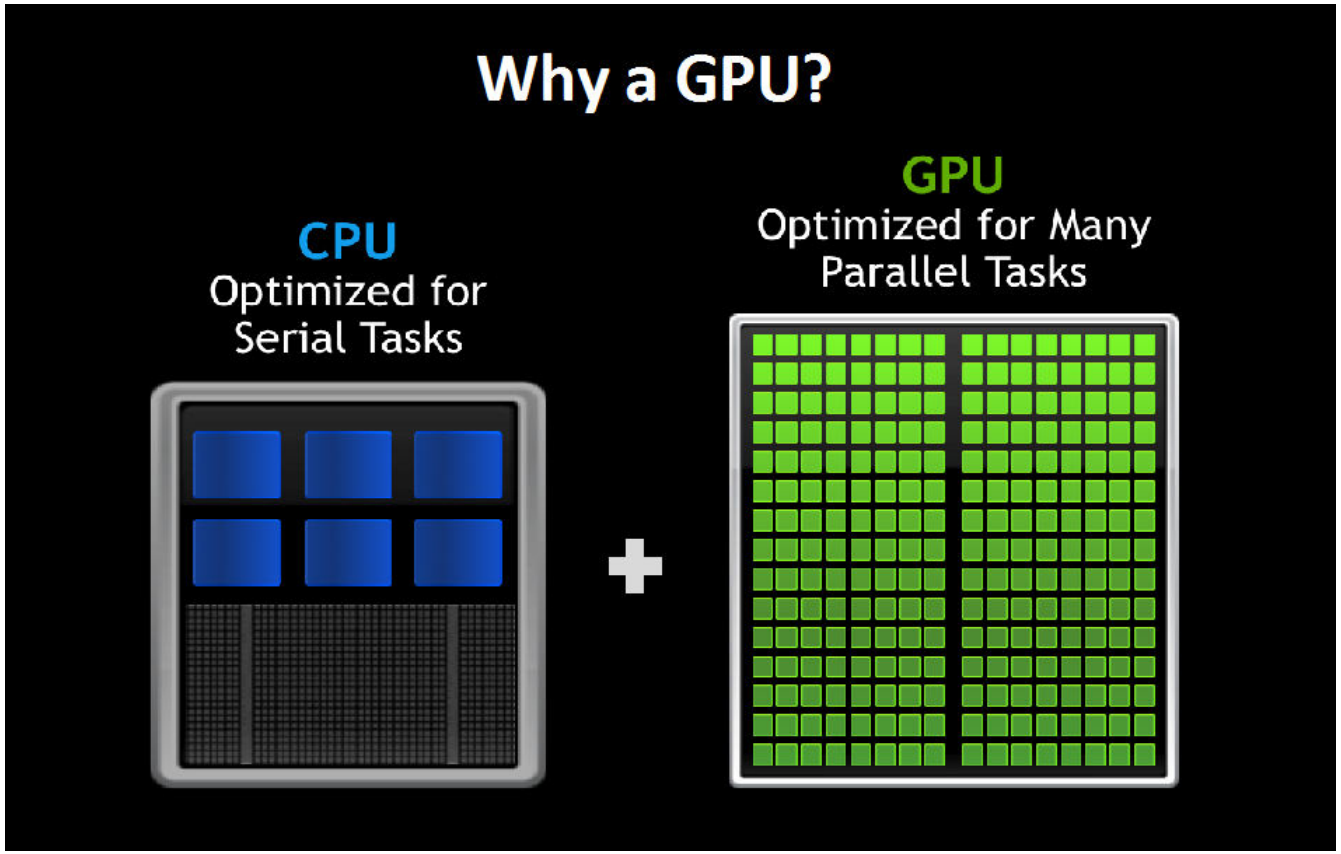
•The specified function requested of a GPU enters the GPU’s hundred’s of cores, and processes all at a single point in time, where it handles each process parallel to the next.

•The GPU has become the most powerful processing unit of the system, especially since organizations are beginning to rely more on the processing power of a GPU rather than a CPU for the single fact that it can process more at a single point in time faster than a CPU.

There are two different types of GPUs:

•**Integrated GPUs** are located on a PC's CPUand share memory with the CPU processor.

•**Discrete GPUs** live on their own card and have their own video memory (VRAM), so that the PC doesn't have to use its RAM for graphics.



Processors on Handheld devices

ARM processor

•An ARM processor is one of a family of CPUs based on the RISC (reduced instruction set computer) architecture developed by Advanced RISC Machines (ARM).

•ARM makes 32-bit and 64-bit RISC multi-core processors. RISC processors are designed to perform a smaller number of types of computer instructions so that they can operate at a higher speed, performing more millions of instructions per second (MIPS).

•By stripping out unneeded instructions and optimizing pathways, RISC processors provide outstanding performance at a fraction of the power demand of CISC (complex instruction set computing) devices.

•ARM processors are extensively used in consumer electronic devices such as smartphones, tablets, multimedia players and other mobile devices, such as wearables. Because of their reduced instruction set, they require fewer transistors, which enables a smaller die size for the integrated circuitry (IC).

•The ARM processor’s smaller size, reduced complexity and lower power consumption makes them suitable for increasingly miniaturized devices.

System on Chip (SoC)based architectures

•Mobile device processor architecture became simple with SOC designs. Real time responsiveness in mobile devices can be managed by using an enhanced DSP hybrid chip. Lowering the voltage of the chip enables low power operation in mobile devices.

•Qualcomm Snapdragon Processors Snapdragon is a family of mobile system on a chip (SoC) processor architecture provided by Qualcomm. Scorpion, the original snapdragon CPU had many features similar to ARM Cortex-A8 core based on ARMv7 instruction set, but with an added advantage of higher performance utilizing SIMD operations.

Smartphone Hardware Architecture

Every modern smartphone today uses a System on a Chip (SoC) Architecture with the following 3 primary components:

- Application processor executing the user's application software with instructions from the middleware and the operating system (OS)
- A baseband (or modem) processor with its own OS components performing baseband radio transmission and reception of audio, video and data
- Various peripheral devices for the user interface

Basic Smartphone architecture