MODULE I

Parallel computer models – Evolution of Computer Architecture, System attributes to performance, Amdahl's law for a fixed workload. Multiprocessors and Multicomputers, Multivector and SIMD computers, Architectural development tracks, Conditions of parallelism.

1.PARALLEL COMPUTER MODELS

- Parallel processing has emerged as a key enabling technology in modern computers, driven by the ever-increasing demand for higher performance, lower costs, and sustained productivity in real-life applications.
- Concurrent events are taking place in today's high-performance computers due to the common practice of multiprogramming, multiprocessing, or multicomputing.

- Parallelism appears in various forms, such as pipelining, vectorization, concurrency, simultaneity, data parallelism, partitioning, interleaving, overlapping, multiplicity, replication, time sharing, space sharing, multitasking, multiprogramming, multithreading, and distributed computing at different processing levels.

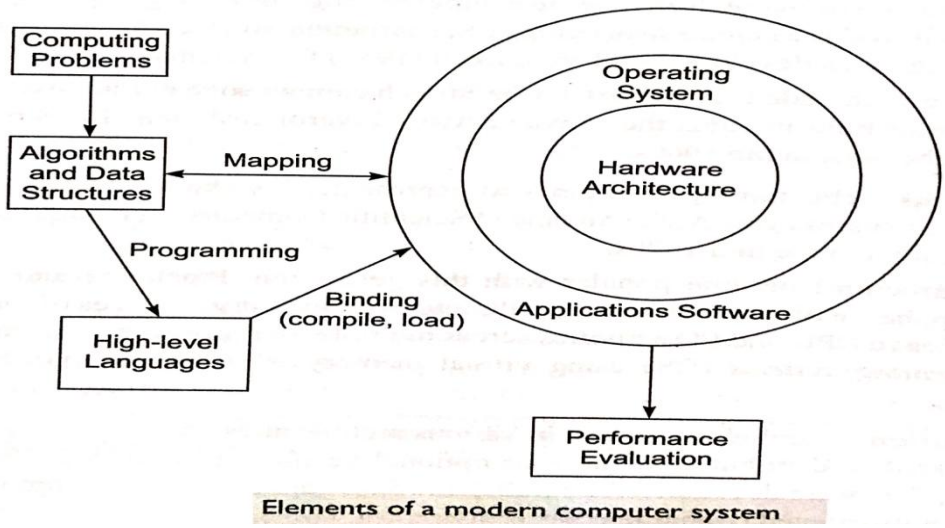
1.1THE STATE OF COMPUTING
1.1.1 Five Generation of Computers

Qn:Explain the five generations of computers?

Five Generations of Electronic Computers

Generation	*Technology and Architecture*	*Software and Applications*	*Representative Systems*
First (1945–54)	Vacuum tubes and relay memories, CPU driven by PC and accumulator, fixed-point arithmetic.	Machine/assembly languages, single user, no subroutine linkage, programmed I/O using CPU.	ENIAC, Princeton IAS, IBM 701.
Second (1955–64)	Discrete transistors and core memories, floating-point arithmetic, I/O processors, multiplexed memory access.	HLL used with compilers, subroutine libraries, batch processing monitor.	IBM 7090, CDC 1604, Univac LARC.
Third (1965–74)	Integrated circuits (SSI/-MSI), microprogramming, pipelining, cache, and lookahead processors.	Multiprogramming and time-sharing OS, multiuser applications.	IBM 360/370, CDC 6600, TI-ASC, PDP-8.
Fourth (1975–90)	LSI/VLSI and semiconductor memory, multiprocessors, vector supercomputers, multicomputers.	Multiprocessor OS, languages, compilers, and environments for parallel processing.	VAX 9000, Cray X-MP, IBM 3090, BBN TC2000.
Fifth (1991–present)	Advanced VLSI processors, memory, and switches, high-density packaging, scalable architectures.	Superscalar processors, systems on a chip, massively parallel processing, grand challenge applications, heterogeneous processing.	

1.1.2 Elements of Modern Computer



Elements of a modern computer system

1.1.3 Evolution of Computer Architecture

Qn:Describe the evolution of parallel computer architecture?

Qn:Explain the term look ahead parallelism?

The study of computer architecture involves both programming/software requirements and hardware organization. Therefore the study of architecture covers both **instruction set architectures** and **machine implementation organizations.**

As shown in figure below, Evolution Started with the **von Neumann architecture** built as a **sequential machine executing scalar data** . The sequential computer was improved from bit-serial to word—parallel operations, and from fixed—point to floating point operations. The von Neumann architecture is slow due to sequential execution of instructions in programs.

Lookahead , parallelism, and pipelining: Lookahead techniques were introduced to prefetch instructions in order to **overlap I/E** (instruction fetch/ decode and execution) operations and to enable **functional parallelism** .

Functional parallelism was supported by two approaches: One is to use **multiple functional units** simultaneously, and the other is to practice **pipelining** at various processing levels.

The latter includes pipelined instruction execution, pipelined arithmetic computations, and memory-access operations. Pipelining has proven especially attractive in performing identical operations repeatedly over **vector data strings**.

A vector is one dimensional array of numbers. A vector processor is CPU that implements an instruction set containing instructions that operate on one dimensional arrays of data called vectors.

Vector operations were originally carried out **implicitly** by software-controlled looping using scalar pipeline processors.

Explicit vector instructions were introduced with the appearance of vector processors. A vector processors equipped with multiple vector pipelines that can be concurrently used under hardware or firmware control.

There are two Families of pipelined vector processors:

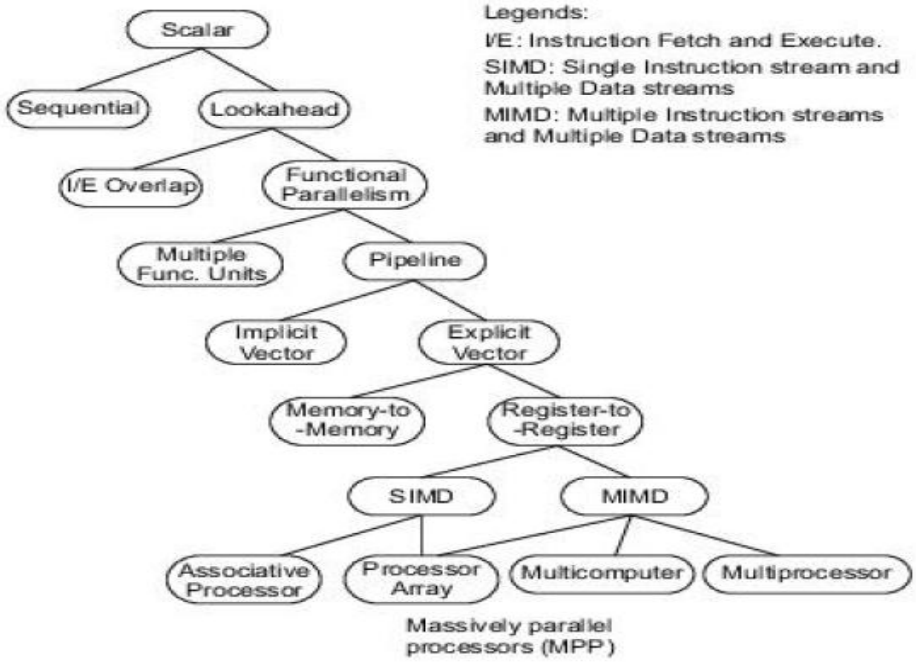
- **Memory –to-memory- architecture** supports the pipelined flow of vector operands directly from the memory to pipelines and then back to the memory.
- **Register-to register** architecture uses vector registers to interface between the memory and functional pipelines.

Another important branch of the architecture tree consists of the **SIMD** computers for synchronized vector processing. An SIMD computer exploits spatial parallelism rather than temporal parallelism as in a pipelined computer .SIMD computing is achieved through the use of an **array of processing elements [PEs]** synchronised by the same controller. Associative memory can be used to build SIMD **associative processors.**

Intrinsic parallel computers are those that execute programs in **MIMD mode**.
There are two major classes of parallel computers, namely, **Shared memory multiprocessors** and **message passing multicomputer**. The major distinction between multiprocessors and multicomputer lies in memory sharing and the mechanisms used for interprocessor communication.

The processors in a **multiprocessor system** communicate with each other through **shared variables in a common memory**.

Each computer node in a **multicomputer system** has a local memory, unshared with other nodes. lnterprocessor communication is done through **message passing** among the nodes.



Tree showing architectural evolution from sequential scalar computers to vector processors and parallel computers

Flynn's Classification (classified into 4)

Qn:Describe briefly about the operational model of SIMD computer with an example?

Qn: Characterize the architectural operations of SIMD and MIMD computers?

Describe briefly about Flynn's classification?

- Michael Flynn (1972) introduced a classification of various computer architectures based on notions of instruction and data streams. **Stream** denote a sequence of items (instructions or data) as executed or operated upon by a single processor. Two types of information flow into a processor: **instructions and data**. The **instruction stream** is defined as the sequence of instructions performed by the processing unit. The **data stream** is defined as the data traffic exchanged between the memory and the processing unit.

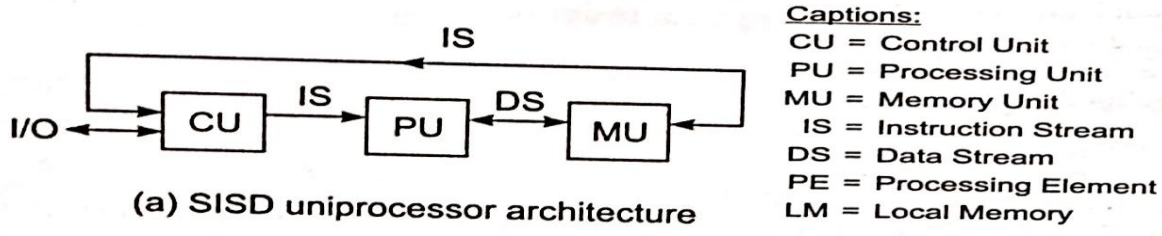
- Both instructions and data are fetched from the memory modules. Instructions are decoded by the control unit, which sends the decoded instruction to the processor units for execution. Data streams flow between the processors and the memory bidirectionally. Each instruction stream is generated by an independent control unit.

According to Flynn's classification, either of the instruction or data streams can be single or multiple. Computer architecture can be classified
into the

- **single-instruction single-data streams (SISD);**
- **single-instruction multiple-data streams (SIMD);**
- **multiple-instruction single-data streams (MISD); and**
- **multiple-instruction multiple-data streams (MIMD).**

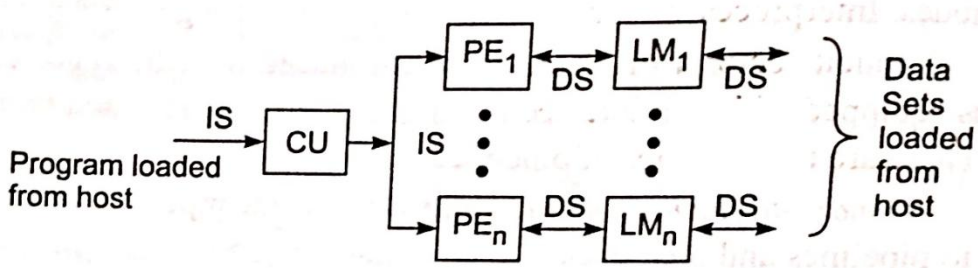
1. **SISD(Single Instruction Single Data Stream)**

- Conventional sequential machines are called SISD -[single instruction stream over single data stream] computers. Instructions are executed sequentially but may be overlapped in their execution stages (pipelining).

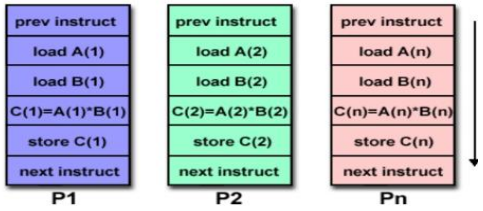


(a) SISD uniprocessor architecture

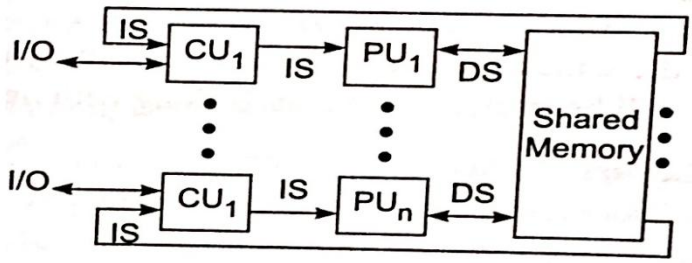
2. **SIMD(Single Instruction Multiple Data Stream)** – Represents vector computers/array processors equipped with scalar and vector hardware.There are multiple processing elements supervised by the same control unit. All PEs receive the same instruction broadcast from the control unit but operate on different data sets from distinct data streams.



(b) SIMD architecture (with distributed memory)



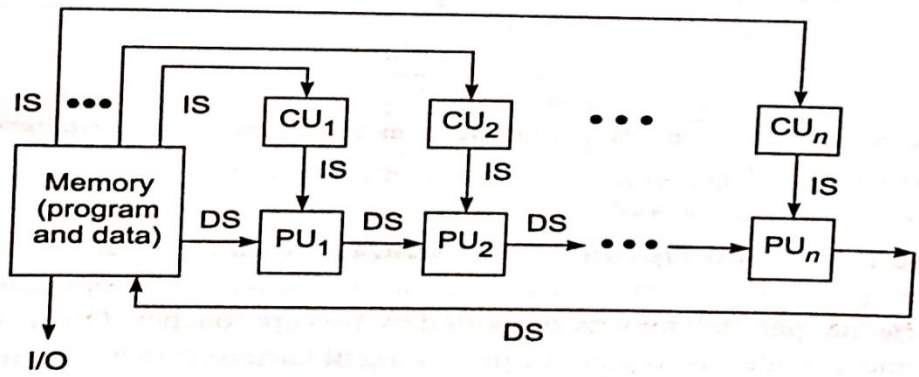
3. **<u>MIMD(multiple instructions over multiple data stream)</u>** – most popular model. parallel computers ters are reserved for MIMD



(c) MIMD architecture (with shared memory)

4. **<u>MISD(multiple instruction over single data stream)</u>**

The same data stream flows through a linear array of processors executing different instruction streams. This architecture is also known as systolic arrays For pipelined execution of specific algorithms.



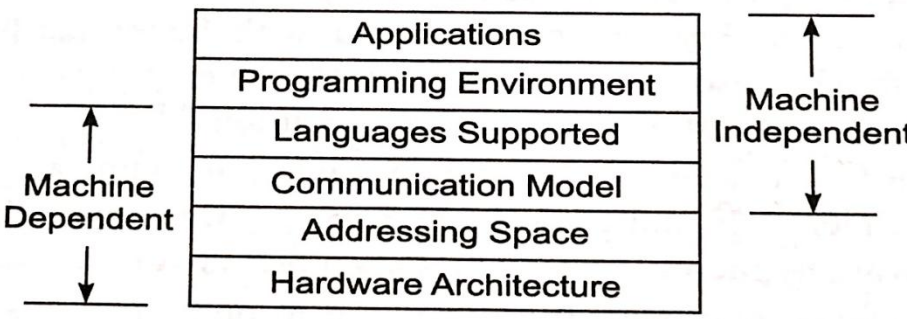
(d) MISD architecture (the systolic array)

Of the four machine models, most parallel computers built in the past assumed the MIMD model for general purpose computations. The SIMD and MISD models are more suitable for special-purpose

computations. For this reason, MIMD is the most popular model, SIMD next, and MISD the least popular model being applied in commercial machines.

Six Layers for Computer System Development

Qn:describe Six layers of computer system development?



Six layers for computer system development (

A layered development of parallel computers is illustrated in Fig. above, based on a classification by Lionel Ni [I990].

- **Hardware configurations** differ from machine to machine, even those of the same model.

- The **address space** of a processor in a computer system varies among different architectures. It depends on the memory organization, which is **machine—dependent.** These features are up to the designer and should match the target application domains.

- On the other hand, we want to develop application **programs and programming environments** which are **machine-independent.** Independent of machine architecture, the user programs can be ported to many computers with minimum conversion costs.

- **High- level languages and communication models** depend on the Architectural choices made in a computer system. From a programmer's viewpoint, these **two layers should be architecture-transparent.**

- **Programming languages** such as Fortran, C, C++, Pascal, Ada, Lisp and others can be supported by most computers. However, the **communication models**, shared variables versus message passing, are mostly machine-dependent.

Challenges in Parallel Processing: Major challenge is on the software and application side.It is still difficult to program parallel and vector computers.High performance computers should provide fast and accurate solutions to scientific ,engineering , business,social and defense problems.

2.SYSTEM ATTRIBUTES TO PERFORMANCE

Qn:Define the terms a)clock rate, b)CPI, MIPS rate, Throughput rate?

Qn:List out the matrics affecting scalability of a computer system for a given application?(Ic,p,m,k t)

Qn:List and explain four system attributes affecting the performance of CPU?(

instruction-set architecture, compiler technology, CPU implementation and control, and cache and memory hierarchy)

System Attributes versus Performance Factors

The ideal performance of a computer system requires a perfect match between machine capability and program behaviour.

Machine capability can be enhanced with better hardware technology, however program behaviour is difficult to predict due to its dependence on application and run-time conditions.

Below are the five fundamental factors for projecting the performance of a computer.

CPU is driven by a clock of constant clock with a **cycle time (τ). T**he inverse of cycle time is the **clock rate (f=1/ τ)**

Size of the program is determined by the ***Instruction Count(Ic).*** Different instructions in a particular program may require different number of clock cycles to execute. So,

Cycles per instruction (CPI):-is an important parameter for measuring the time needed to execute an instruction .

Execution Time/CPU Time (T): Let **Ic** be Instruction Count or total number of instructions in the program. The Execution Time or CPU time (T) will be:

$$T = Ic \times CPI \times \tau \qquad \text{Eq.1.1}$$

- The execution of an instruction involves the instruction fetch, decode, operand fetch, execution and store results.

Only instruction decode and execution phases are carried out in CPU. The remaining three operations may require access to memory

The CPI of an instruction type can be divided into two component terms corresponding to the total processor cycles and memory cycles needed to complete the execution of the instruction. That is :-

CPI = Instruction Cycle = Processor Cycles +Memory Cycles. ie

CPI = Instruction cycle = p + m + k

where

m = number of memory references

P = number of processor cycles

k = latency factor (how much the memory is slow w.r.t to CPU)

Therefore, Equation 1.1 can be rewrite as follows:-

$$T = Ic \times (p+m+k) \times \tau \qquad \text{Eq.1.2}$$

From the above equation the five factors affecting performance are:-Ic ,p,m,k, τ

System Attributes: The above five performance factors (**Ic ,p,m,k, τ**) are influenced by four system attributes**: instruction-set architecture, compiler technology, CPU implementation and control, and cache and memory hierarchy**, as specified in Table 1.2 below.

The instruction-set architecture affects the program length (1,) and processor cycles needed (p). The compiler technology affects the values of Ic, p, and the memory reference count (m). The CPU implementation and control determine the total processor time (p * τ) needed. Finally, the memory technology and hierarchy design affect the memory access latency (k * τ). The above CPU time can be used as a basis in estimating the execution rate of a processor.

Table 1.2 *Performance Factors versus System Attributes*

System Attributes	Performance Factors				
	Instr. Count, I_c	Average Cycles per Instruction, CPI			Processor Cycle Time, τ
		Processor Cycles per Instruction, p	Memory References per Instruction, m	Memory-Access Latency, k	
Instruction-set Architecture	✓	✓			
Compiler Technology	✓	✓	✓		
Processor Implementation and Control		✓			✓
Cache and Memory Hierarchy				✓	✓

MIPS Rate. The processor speed is often measured in terms of million instructions per second (MIPS). We simply call it the MIPS rate of a given processor.

Let C be the total number of cycles needed to execute a given program (ie **C=Ic * CPI**).
Then the **CPU time** in Eq. 1.2 can be estimated as

$$T = C * \tau$$
$$= C/f .$$

Furthermore, **CPI = C/Ic and**

$$T = Ic * CPI * \tau$$
$$= Ic * CPI/f.$$

MIPS rate $= Ic/(T*10^6)$
$= f/(CPI*10^6)$
$= (f*Ic)/(C*10^6)$

Eq.1.3

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{\text{CPI} \times 10^6} = \frac{f \times I_c}{C \times 10^6}$$

Or

Now Based on Equation 1.2 and 1.3

$$\textbf{CPU Time ,T} = (IC*10^{-6})/MIPS$$

MFLOPS:

Most compute intensive applications in science and engineering make heavy use of floating point operations. For such applications a more relevant measure of performance is floating point operations per second abbreviated as mflops. With prefix mega(10^6),giga(10^9) tera(10^12) or peta(10^15). Floating-point performance is expressed as millions of floating-point operations per second (MFLOPS), defined as follows ,only with floating-point instructions.

Number of executed floating-point operations in a program

$$MFLOPS = \frac{}{\text{execution time} * 10^6}$$

Throughput Rate:- Number of programs executed per unit time is called system throughput w_s(in programs per second).In a multiprogrammed system, the system throughput is often lower than CPU throughput Wp defined by

$$\mathbf{Wp = \frac{f}{Ic * CPI}} \qquad \textbf{Eq.1.4}$$

W = 1 / T
OR
W =(MIPS*10^6)/Ic

Problems:-

1 **A benchmark program is executed on a 40MHz processor. The benchmark program has the following statistics.**

Instruction Type	Instruction Count	Clock Cycle Count
Arithmetic	45000	1
Branch	32000	2
Load/Store	15000	2
Floating Point	8000	2

Calculate average CPI, MIPS rate & execution time for the above benchmark program

Given Clock speed of the processor= 40 MHz=40*10^6 Hz

Average CPI= C/Ic

=Total cycles to execute the program/Instruction count

=(45000 *1 + 32000*2 + 1500*2 + 8000*2) /(45000 + 3200 + 15000 + 8000)

=155000/100000

=1.55

Execution Time ,T= C / f

T = 150000 / 40 *10^6

T = 0.155 / 40

T = .003875 s

T = 3.875 ms (since 1s=1000ms)

MIPS rate = Ic / T * 10^6

MIPS rate = 100000/(0.003875*$10^{6)}$

= 25.8

2. Consider the execution of an object code with 2 *10^6 instructions on a 400 MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles [CPI] needed for each instruction type are given below based on the result of a program trace experiment:

Instruction type	CPI	Instruction mix
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

(a) Calculate the average CPl when the program is executed on a uniprocessor with the above trace results.
(b) Calculate the corresponding MIPS rate based on the CPI obtained in part (a).

Answer:
average CPI = 0.6 +(2 *0.18) +(4* 0.12) +(8 *0.1) =2.24.
MIPS rate = f/(CPI*$10^{6)}$
=(400* 10^6) /(2.24 *10^6) =178

Programming Environments:

Qn: Difference between Implicit Parallelism and Explicit forms of Parallelism?

The programmability of a computer depends on the programming environment provided to the users. conventional uniprocessor computers are programmed in a ***sequential environment*** in which instructions are executed one after another in a sequential manner. Parallel computers employs parallel environment where parallelism is automatically exploited.. Based on the programming environments required parallelism can be of two types:

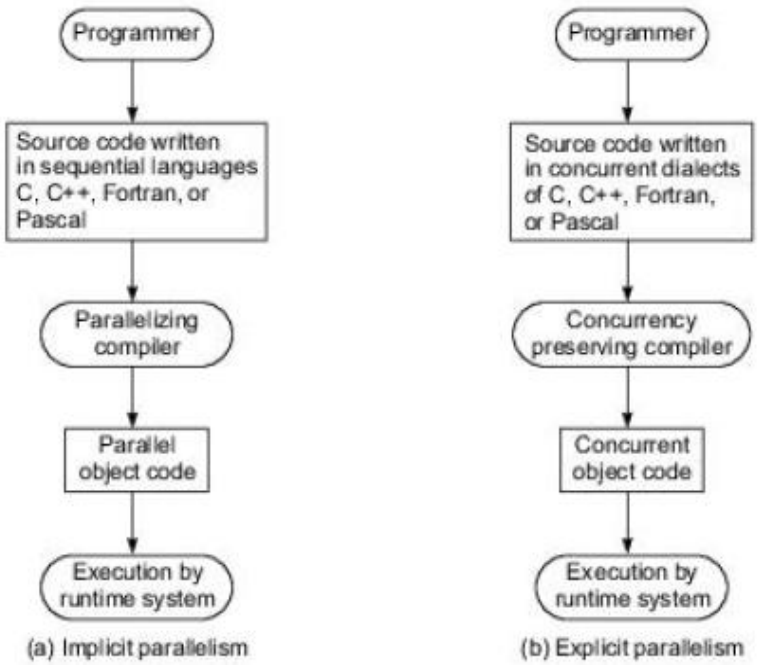


Fig. 1.5 Two approaches to parallel programming (Courtesy of Charles Seitz; adapted with permission from "Concurrent Architectures", p. 51 and p. 53, *VLSI and Parallel Computation*, edited by Suaya and Birtwistle, Morgan Kaufmann Publishers, 1990)

IMPLICIT PARALLELISM	EXPLICIT PARALLELISM
1. In computer science, implicit parallelism is a characteristic of a programming language that allows a **compiler or interpreter** to automatically exploit the parallelism inherent to the computations expressed by some of the language's constructs.	1. In computer programming, explicit parallelism is the representation of concurrent computations by means of primitives in the form of special-purpose directives or function calls. Most parallel primitives are related to process synchronization, communication or task partitioning.
2. Uses conventional languages such as C, C++, Fortran or Pascal to write source program	2. Requires more effort by programmers to develop a source program using parallel dialects like C, C++, Fortran and Pascal.
3. The **sequentially coded** source program is translated into parallel object code by a **parallelizing compiler.**	3. Parallelism is explicitly specified in the user programs.
4. Compiler detects parallelism and assigns target machine resources.	4. Burden on compiler is reduced as parallelism specified explicitly.
5. Success relies on intelligence of parallelizing compiler. Requires less effort from programmers.	5. Programmer's effort is more- special s/w tools needed to make environment more friendly to user groups.
6. Applied in shared memory multiprocessors.	6. Applied in Loosely coupled Multiprocessors to tightly coupled VLIW

3. MULTIPROCESSORS AND MULTICOMPUTERS

Qn. Distinguish between multiprocessor and multicomputers based on their structure, resource sharing, and interprocessor communication)?

Qn:List differences between UMA, NUMA,COMA Models?

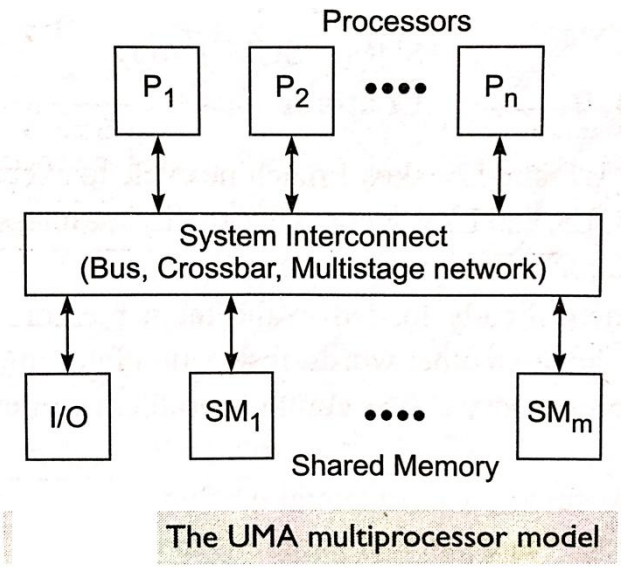
Qn:Describe in detail the types of shared memory multiprocessors models?

- 2 categories of parallel computer
- Distinguished by having shared memory(multiprocessors) or unshared distributed memory(multi computers)

Multiprocessors	Multicomputer
1. Single computer with multiple processors	1. Multiple autonomous computers
2. Each PE's(CPU/processing elements) do not have their own individual memories – memory and I/O resources are shared – Thus called **Shared Memory Multiprocessors**	2. Each PE's has its own memory and resources – no sharing – Thus called **Distributed Memory Multicomputers**
	3. Communication between PE's not

3. Communication between PE's a must 4. Tightly coupled – due to high degree of resource sharing 5. Use Dynamic Network – thus communication links can be reconfigured 6. Ex: Sequent Symmetry S-81 7. 3 Types - **UMA model** - **NUMA model** **COMA model**	mandatory 4. Loosely coupled as there is no resource sharing 5. Use Static Network – connection of switching units is fixed 6. Ex: Message Passing Multicomputer **7. NORMA model/ Distributed-Memory Multicomputer**

THE UMA MODEL



The UMA multiprocessor model

- Physical memory is uniformly shared by all processors
- All processors (PE1….PEn) take equal access time to memory – Thus its termed as **Uniform Memory Access Computers**
- Each PE can have its own private Cache
- High degree of resource sharing(memory and I/O) – Tightly Coupled
- Interconnection Network can be – Common bus, cross bar switch or Multistage n/w (discussed later)
- When all PE's have equal access to all peripheral devices – **Symmetric Multiprocessor**
- In **Asymmetric multiprocessor** only one subset of processors have peripheral access. Master Processors control Slave (attached) processors.

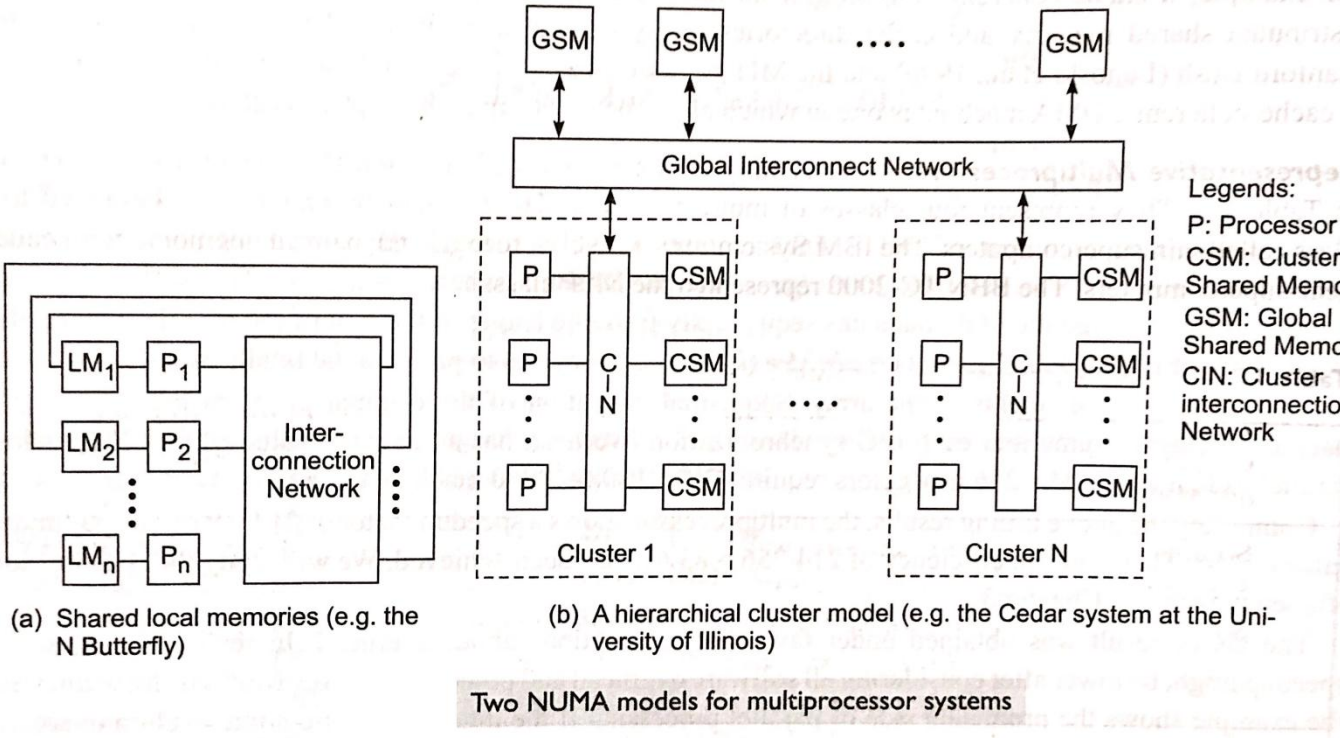
Applications of UMA Model

- Suitable for general purpose and time sharing application by multiple users
- Can be used to speed up execution of a single program in time critical application

DISADVANTAGES

- Interacting process cause simultaneous access to same locations – cause problem when an update is followed by read operation (old value will be read)
- Poor Scalability – as no: of processors increase –shared memory area increase-thus n/w becomes bottleneck.
- No: of processors usually in range(10-100)

THE NUMA MODEL (Ex: BBN Butterfly)



Two NUMA models for multiprocessor systems

- Access time varies with location of memory
- Shared memory is distributed to all processors – Local memories
- Collection of all local memories forms global memory space accessible by all processors.
- Its faster to access content within local memory of a processor than to access remote memory attached to another processor (delay through interconnection) – Thus named as **NON-Uniform Memory access** – because access time depends on whether data available in local memory of the processor itself or not.

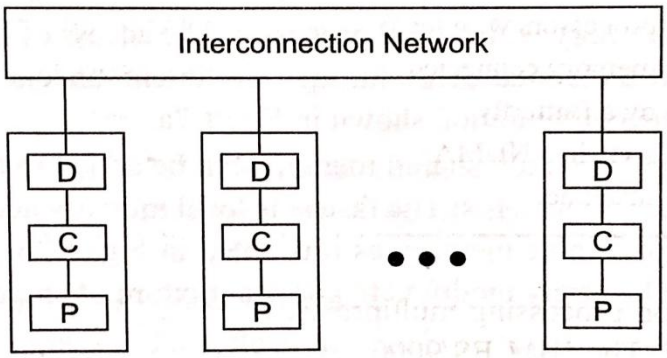
Advantage

- – reduces n/w bottleneck issue that occurs in UMA as prcessors have a direct access path to attached local memory

3 types of Memory access pattern

I. Fastest to access – local memory of PE itself
II. Next fastest – global memory shared by PE/Cluster
III. Slowest to access – remote memory(local memory of PE from another Cluster)

THE COMA MODEL



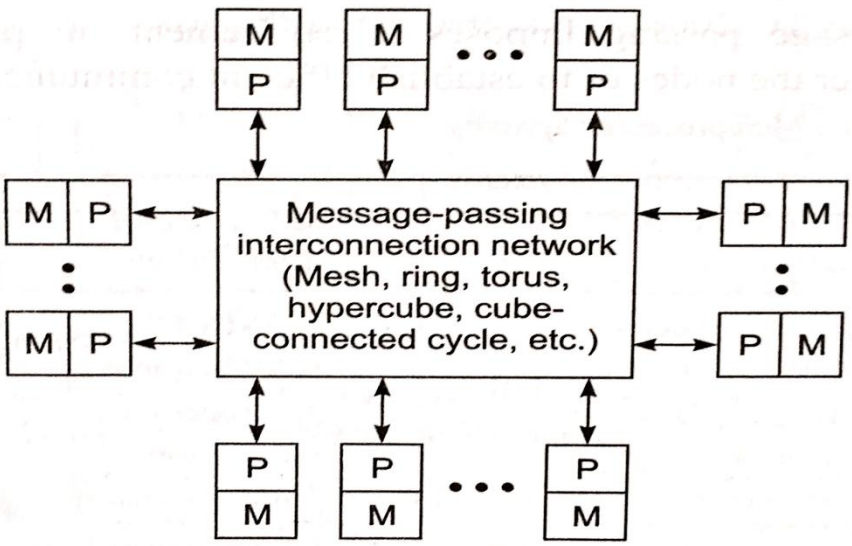
The COMA model of a multiprocessor (P: Processor, C: Cache, D: Directory;

- Multiprocessor + Cache Memory = COMA Model
- **Multiprocessor using cache only memory**
- Ex: Data diffusion Machine , KSR-1 machine
- Special case of NUMA machine in which distributed main memories are converted to caches – all caches together form a global address space
- Remote cache access is assisted by Distributed Cache directories (D in fig abv)

Application of COMA

- General purpose multiuser applications

DISTRIBUTED MEMORY / NORMA / MULTICOMPUTERS (Intel Paragon, nCUBE, SuperNode1000)



Generic model of a message-passing multicomputer

- Consists of multiple computers(called nodes)
- A node is an autonomous computer consisting of processor, local memory attached disks or I/O peripherals.
- Nodes interconnected by a message passing network – can be Mesh, Ring, Torus, Hypercube etc (discussed later)
- All interconnection provide point to point static connection among nodes