relatively inexpensive off-the-shelf networked PCs, specialize parallel machines are not cost-effective to manufacture. The additionally require special compiler and other system support fo maximum throughput.

Q 13

The general idea for **parallel processing/computing** is t distribute computation among processors or split the job into task and execute these tasks concurrently on different processors.

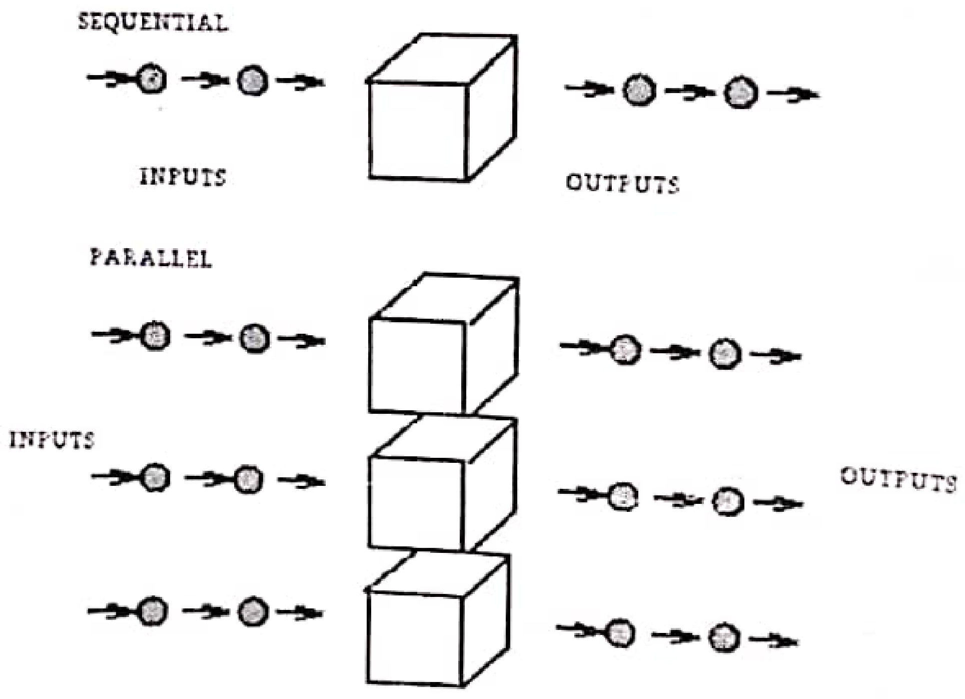


Figure 1.4: Sequential and parallel processing.

The parallel version has the potential of being 3 times as the sequential machine. So, it appears that the only way f is to use PARALLELISM. The idea here [illegible] if several ope can be performed simultaneously the [illegible] l computatio is reduced.

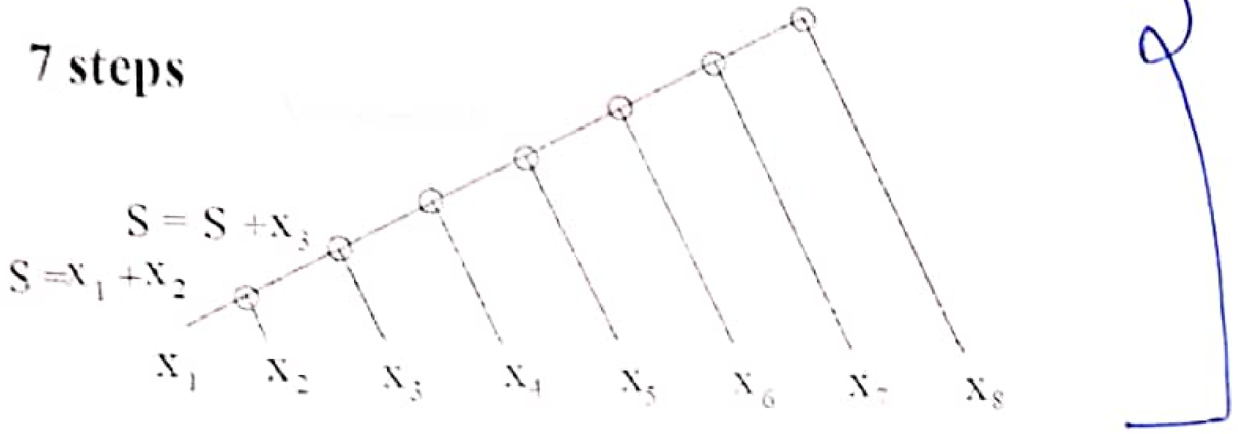
Example: Show how the following operation may be computed at a uniprocessor system and at a multiprocessor system.

$$S = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$$

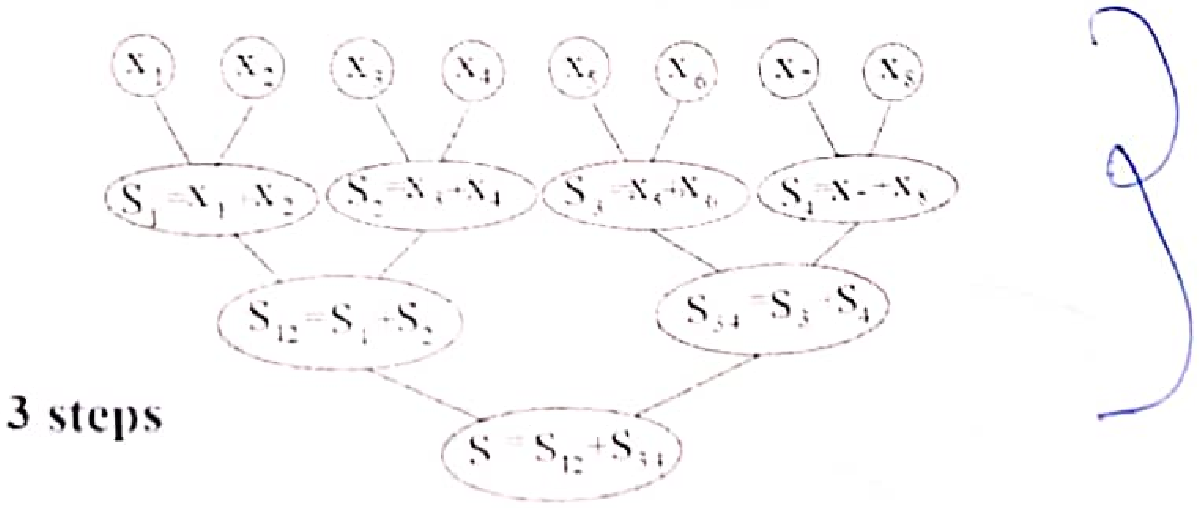
Solution:

Q1 Sheet 2

For **uniprocessor system** (sequential processing), 7 steps computations are done sequentially.



For **multiprocessor system** (parallel processing), 3 step computations are done.



collaboratively achieve a common
system and what is **distributed processing/com**

Distributed Systems:

Various definitions of **distributed systems** have been given in the literature. A distributed system can be defined as:

"A collection of autonomous computers interconnected by a computer network and equipped with distributed system software to form an integrated computing facility"

"A system in which hardware or software components located at networked computers that communicate and coordinate their actions only by message passing"

"A system consists of a collection of two or more independent computers which coordinate their processing through the exchange of synchronous or asynchronous message passing"

"A collection of autonomous computers linked by a network with software designed to produce an integrated computing facility"

"A collection of independent computers that appear to the users of the system as a single computer"

Sheet 2
Q2

In short, a distributed system is basically a collection of independent computers interconnected by a communication network and coordinate their actions only by message passing but appears to its users as a single coherent system.

A typical distributed system would look as shown in Figure 1.5. Each computer has a memory unit and a processing unit and the computers are connected by a communication network.

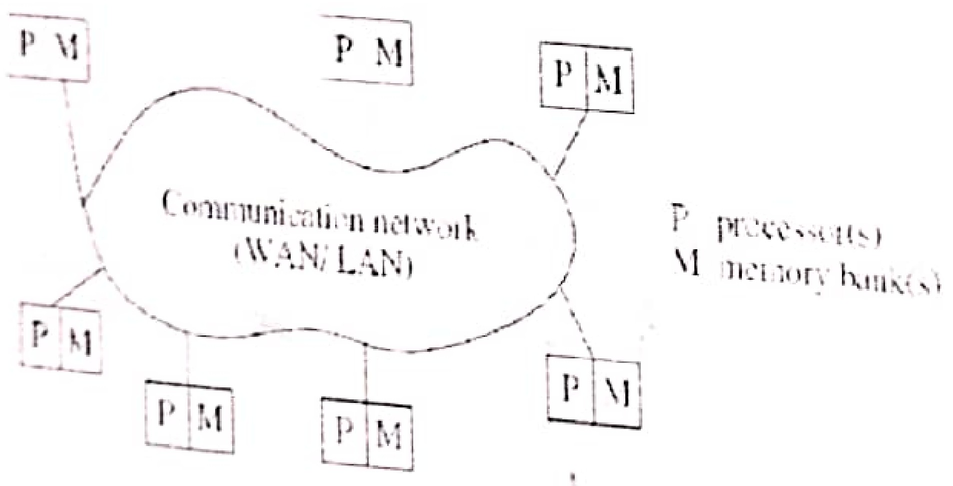


Figure 1.5: A distributed system connects multiple computers by a communication network.

The definition of distributed system has two essential aspects. The first one deal with ***hardware***: the machines are autonomous (independent). The second one deal with ***software***: the users think that they are dealing with a single system. Although the hardware issues of building such systems were fairly well understood, the major stumbling block at that time was the availability of adequate software for making these systems easy to *these points later in the*

Distributed Processing/Computing:

Distributed computing offers the ability to solve complex proble in reasonable times. Despite this, the full potential of large sc multiprocessor systems is still to be realized.

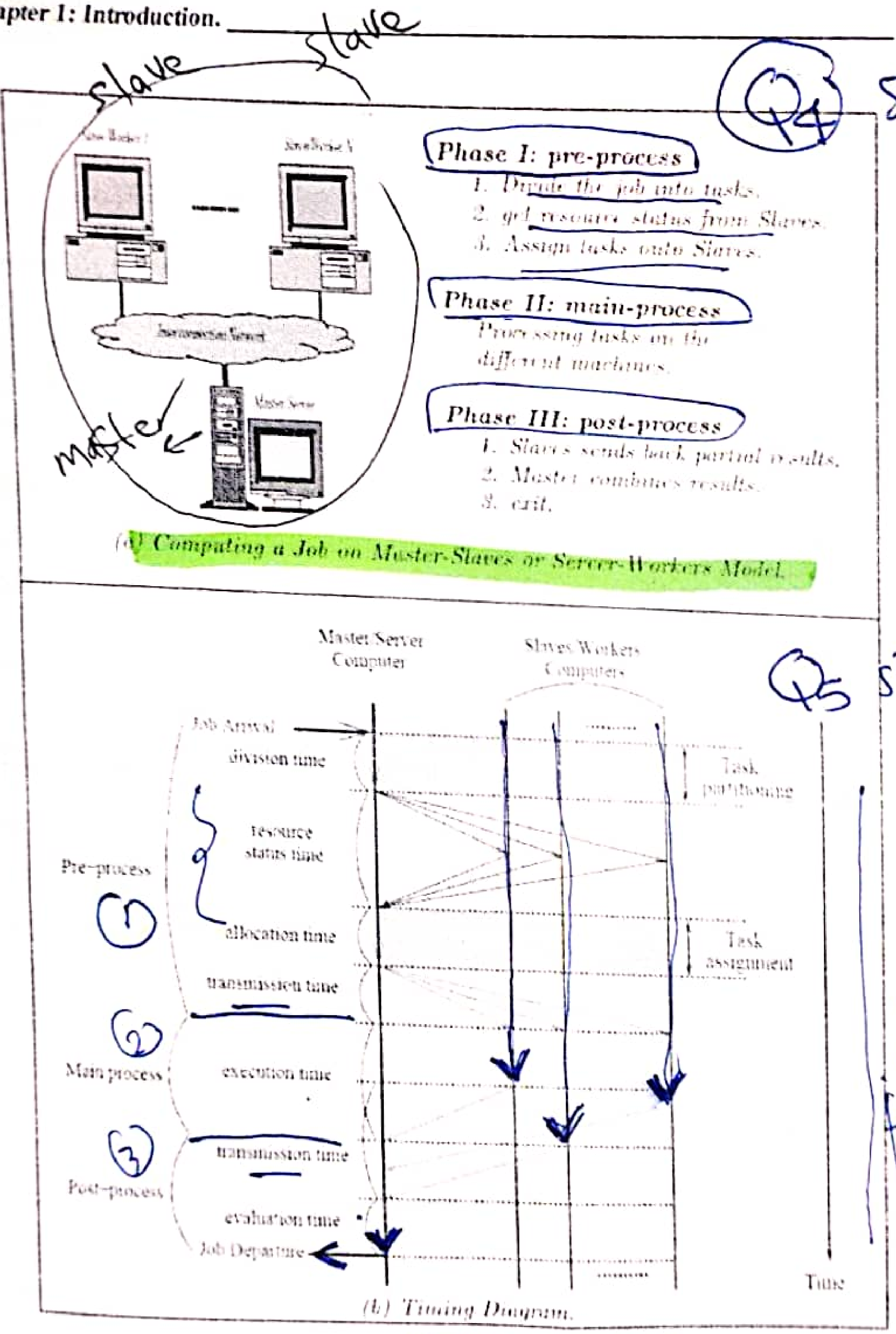
The goal of distributed processing is thus to solve a given proble more rapidly or to enable the solution of a problem that woul otherwise be impracticable by a single computer.

QB Sheet 2

Distributed computing is a science which solves a large problem b giving small parts of the problem to many computers to solv concurrently and then combining the solutions for the parts into a solution for the problem. Co-operation will always be necessary between computers during problem solution, even if this is a simple agreement on the division of labor. These ideas can be illustrated by a simple analogy of tacking the problem of emptying swimming pool using buckets.

Recent distrib

Q4 sheet2



(a) Computing a Job on Master-Slaves or Server-Workers Model.

Q5 sheet2

(b) Timing Diagram.

Figure 1.6: Distributed Computing.

Chapter 2

Parallel and Distribute
Systems Architecture

In this chapter, we first present some character
distributed systems and then examine the archite
parallel systems. Afterwards, we present a nun
common architectural classifications of the parallel s

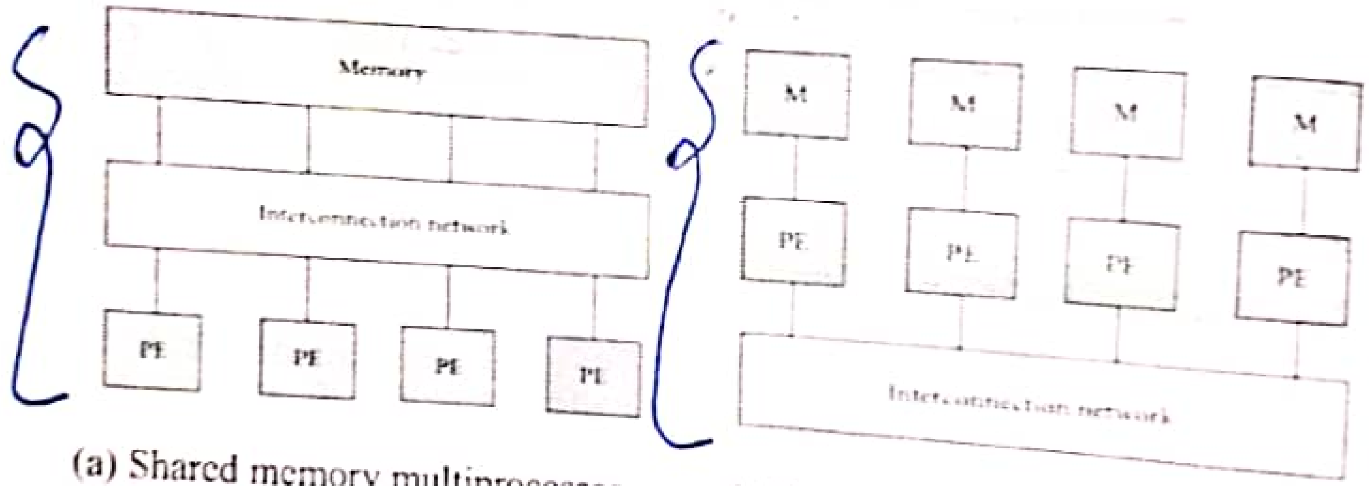
2.2 Architecture of parallel systems

A parallel system may be broadly belonging to one of three types: **multiprocessor system**, **multicomputer system** and **array processors**. ① ⇓ processing ②

2.2.1 Multiprocessor system:

A multiprocessor system is a parallel system in which the multiple processors have direct access to shared memory which forms a common address space. The architecture is shown in Figure 2.1(a).

The processors are usually of the same type, and are housed within the same box/container with a shared memory. Such processors usually do not have a common clock. The usually a multistage switch with a symmetric and regular design. In such parallel system, all the processors usually run the same operating system, and both the hardware and software are very tightly coupled.



(a) Shared memory multiprocessor (b) Distributed memory multicomputer

Figure 2.1: Two standard architectures for parallel systems.

2.2.2 Multicomputer system:

A multicomputer parallel system

2.3 Architectural Classifications

of Parallel Computers

Parallel computers can be classified by various aspects of their architecture. Here, we present **four** different classification schemes. In the first, parallel computers are distinguished by the way the processors are connected with the memory. The second scheme, called "Flynn's Classification Scheme", takes the number of instruction-streams and the number of data-streams into account. The third scheme, the Erlanger Classification Scheme (ECS), focuses on the number of control units, functional units, and the word-size of the computer. Finally, Johnson's classification focuses on the different memory access methods. First

2.4 Memory-Processor Organization

In terms of memory-processor organization, three main groups of architectures can be distinguished. These are

1 • Shared memory architectures,

2 • Distributed memory architectures, and

3 • Distributed shared memory architectures.

In the following these approaches are briefly described.

2.4.1 Shared Memory Architectures

In shared memory architecture, each CPU shares main memory and peripherals to execute instructions at the same time. It offers a single memory space for all the CPUs. The main memory consists of several memory modules, whose number is not necessarily equal to the number of processors in the computer. Indeed, the processors are connected to the memory modules via some kind of interconnection network, to be explained later, see Figure 2.2. The main property of shared memory architectures is that all processors in the system have access to the same memory; there is only one global address space. In such a system, communication (data transfer processors) and synchronization between the processors is done implicitly via shared variables.

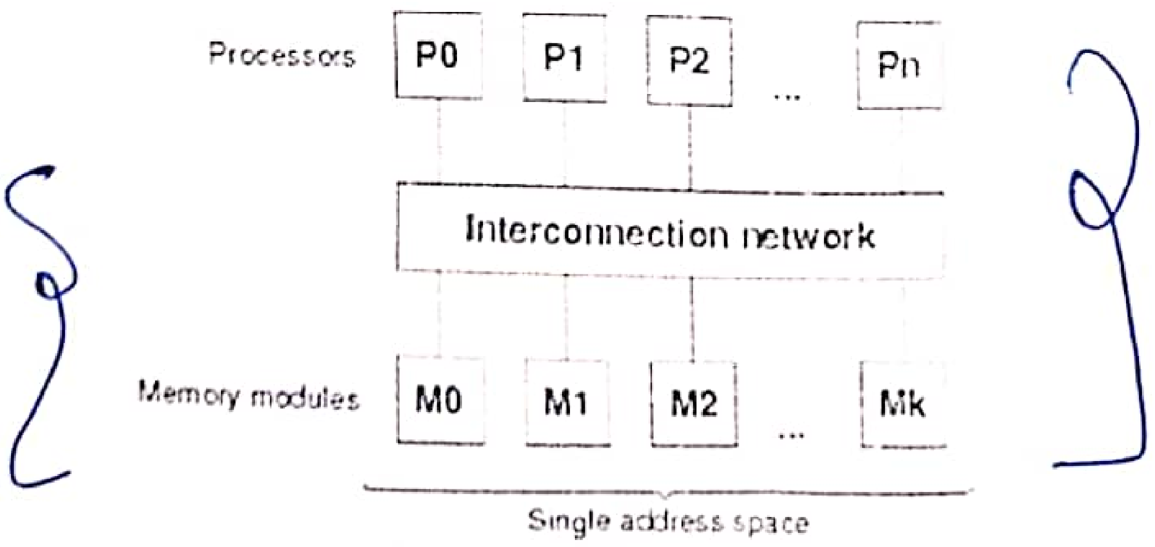


Figure 2.2: shared memory architecture.

This type of parallel computer is also called **UMA**, which stands for Uniform Memory Access, since all processors access every memory module in the same way concerning latency and bandwidth.

A big $\boxed{\textbf{advantage of shared memory}}$ computers is that programming a shared memory computer is very convenient (1) due to the fact that all data are accessible by all processors, such that there is no need to copy data. (2) Furthermore the programmer does not have to care for synchronization, since this is carried out by the operating system automatically disAdv (which makes the hardware more complex and hence more expensive). However, (1) it is very difficult to obtain high levels of parallelism with shared memory machines; (2) most systems do not have more than 64 processors. This limitation stems from the fact, that a centralized memory and the interconnection network are both difficult to scale once built.

2.4.2 Distributed Memory Architectures

In a distributed memory computer (in literature also called multicomputer), each processor has its own, private memory. There is no common address space, i.e. the processors can access only their own memories. Every CPU can operate on local data to perform computational tasks. If remote data is required, the CPU can communicate with other CPUs via the interconnection network. Communication and synchronization between the processors is done by exchanging messages over the interconnection network, *to be explained later.*

Figure 2.3 shows the organization of the processors and memory modules in a distributed memory computer. In contrary to shared memory architecture, a distributed memory machine scales very well, since all processors have their own local memory which means that there are no memory access conflicts. Using this architecture, Massively Parallel Processors (MPP) can be built, with up to several hundred or even thousands of processors.

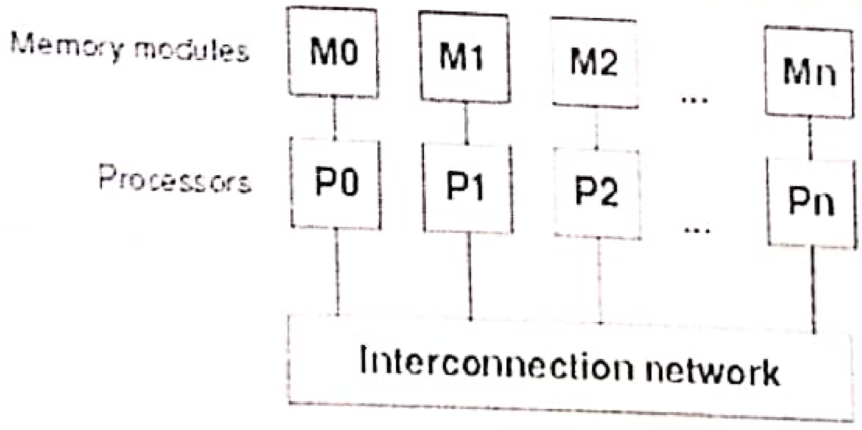


Figure 2.3: distributed memory architecture.

Typical representatives of pure distributed memory architecture are clusters of computers (a **type of distributed systems**), which become more and more important nowadays. In a cluster, each node is a complet computer, and these computers are connected through low-cost commodity network (e.g. Ethernet ... etc.). The bi advantage of clusters compared to MPPs is that they have much better cost/performance ratio. However, until no clusters have not reached the peak performance of the MPPs.

Distributed Memory	Shared Memory
Large number of processors (100's - 1000's)	modest number of processors (10's - 100's)
High theoretical power	Modest power
Unlimited expansion	Limited expansion
Difficult to fully utilise	Easy to fully utilise
Revolutionary parallel Programming	Evolutionary parallel Programming

Comparison Table

2.4.3 Distributed Shared Memory Architecture

To combine the advantages of the two architectures described above, i.e., ease of programming on the one hand and high scalability on the other hand, a third kind of architecture has been established: distributed shared memory machines. Here, each processor has its own local memory, but contrary to the distributed memory architecture, all memory modules form one common address space, i.e. each memory cell has a system-wide unique address. In order to avoid the disadvantage of shared memory computers, namely the low scalability, each processor uses a cache, which keeps the number of memory access conflicts and the network contention low (Figure 2.4). However, the usage of caches introduces a number of problems, for example how to keep the data in the memory and the copies in the caches up-to-date. This problem is solved by using sophisticated cache coherence and consistency protocols.

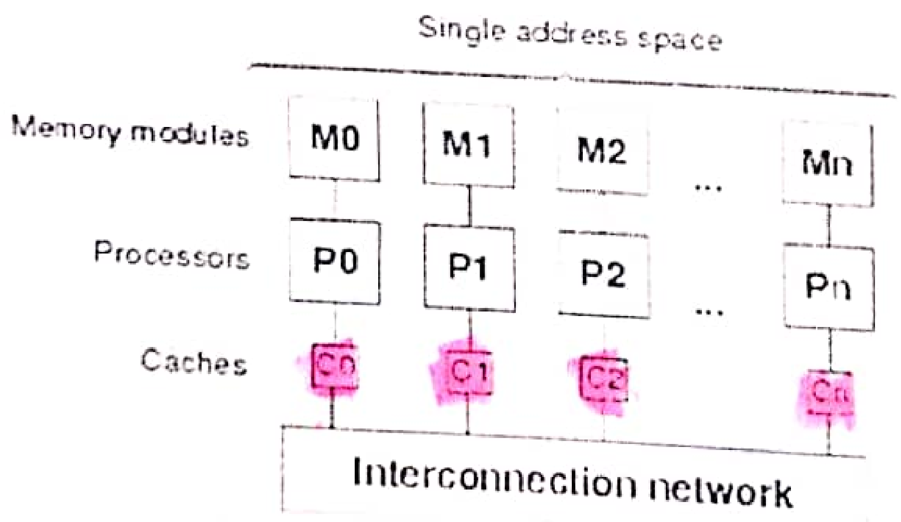


Figure 2.4: distributed shared memory architecture.

2.5 Flynn's Classification Scheme

Besides the informal classification presented above, another widely accepted and often used classification scheme was proposed by Michael Flynn in 1966. Flynn has identified two essential characteristics to classify multiple CPU computer systems: the number of instruction streams and the number of data streams, respectively.

Traditionally, any computer, whether sequential or parallel, operates by executing instructions on data. A stream of instructions (the algorithm) tells the computer what to do. A stream of data (the input) is affected by these instructions. Flynn identified four processing modes, based on whether the processors execute the same or different instruction streams at the same time, and whether or not the processors processed the same (identical) data at the same time. Depending on whether there is one or several of these streams, we have four classes of computers, as shown in Table 2.1. Hence, the following four classes are distinguished:

1. Single Instruction Stream, Single Data Stream: SISD.
2. Multiple Instruction Stream, Single Data Stream: MISD.
3. Single Instruction Stream, Multiple Data Stream: SIMD.
4. Multiple Instruction, Multiple Data Stream: MIMD.

Table 2-1: Flynn's Classification Scheme

	Single instruction stream (SI)	**Multiple instruction streams (MI)**
Single data stream (SD)	SISD	SIMD
Multiple data streams (MD)	MISD	MIMD

It is instructive to examine this classification to understand the range of options used for configuring systems.

2.5.1 SISD Computers

A⁺B

The SISD (Single Instruction stream, Single Data stream) class corresponds to the traditional sequential computers or uni-processor computers (a single CPU, and a single memory unit connected by a system bus),/which are based on the von-Neumann architecture. In this type, a single processing unit receives a single stream of instructions that operate on a single stream of data. The single processor executes the sequence of instructions that operate on a single stream of data, as shown in Figure 2.5.

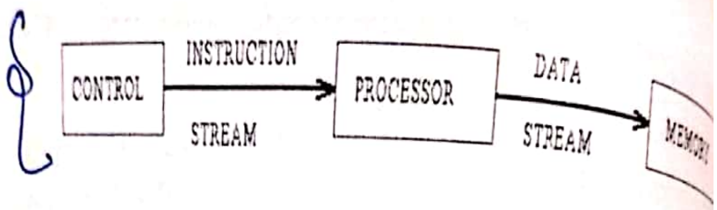


Figure 2.5: SISD Computers

Examples of this architecture are: personal computers
some exceptions such as Intel Dual Pentium machines)
workstations and low-end servers.

EXAMPLE: To compute the sum of N numbers a1, a2, ... aN
processor needs to gain access to memory N consecutive times
receive one number at each time). Also, N-1 additions
executed in sequence. Therefore, the computation takes O
operations. That is, algorithms for SISD computers do not cont
any parallelism, there is only one processor.

2.5.2 SIMD Computers

$\overset{+}{AB}$ $A+B$
CD $C+D$

In this architecture, there are N identical processors operat
under the control of a single instruction stream issued by a
central control unit. Those processors are usually special
purpose since full generality is not required. Indeed, there
are N data streams, one per processor so different data can
be used in each processor. To ease understanding, assume
that each processor holds the same identical program.

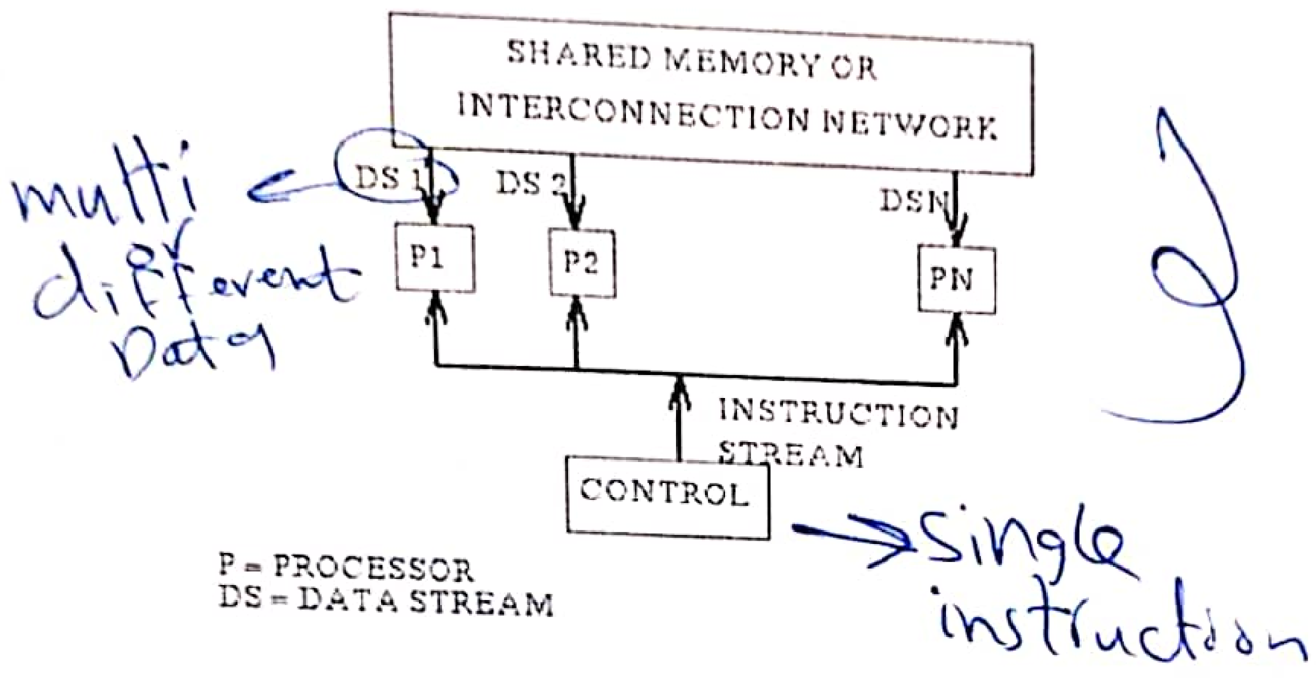


Figure 2.6: SIMD Computers

The SIMD corresponds to the processing by multiple homogenous processors which execute in lock-step on different data items. The processors operate synchronously and a global clock is used to ensure lockstep operation, i.e. at each step (global clock tick) all processors execute the same instruction, each on a different data.

Several of the earliest parallel computers, such as Illiac-IV, MPP, CM2, and MasPar MP-1 were SIMD machines. Vector processors, array processors' and systolic arrays also belong to the SIMD class of processing. The two most important representatives of the SIMD class are the array of processors computer and all vector processors. Array of processors such as the ICL DAP (Distributed Array Processor) and pipelined

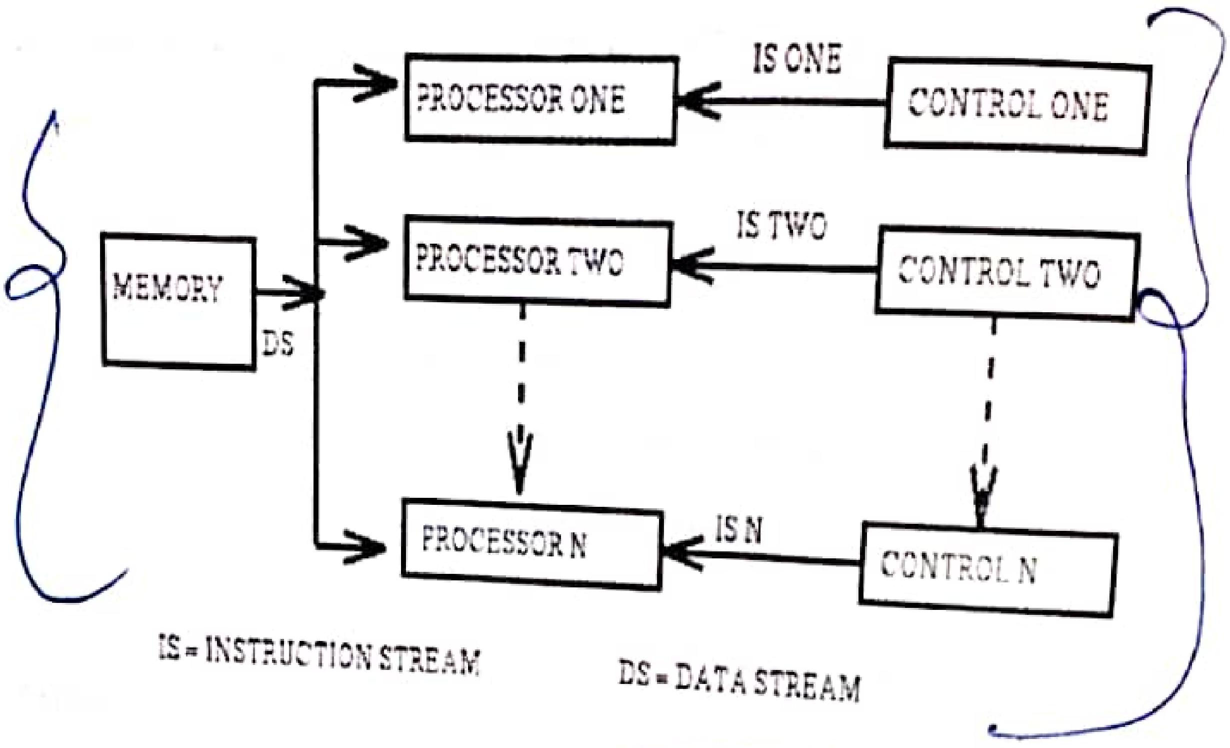


Figure 2.7: MISD Computers

There are N streams of instructions (algorithms/programs) and one stream of data. Parallelism is achieved by letting the processors do different things at the same time on the same data. This mode corresponds to the execution of different operations in parallel on the same data. This is a specialized mode of operation with limited but niche applications, e.g., visualization. For most applications, MISD are very awkward to use and no commercial machines exist with this design.

Traditionally, no existing commercial machines correspo the MISD model, but special purpose machines existin example, a "cracker computer" with a set of pro where all are fed the same stream of ciphered dat each tries to crack using a different algorithm.

Generally, MISD machines are useful in computations where the same input is to be subjected to several different operations.

EXAMPLE: Checking whether a number Z is prime. A simple solution is to try all possible divisions of Z. Assume the number of processors, N, is given by N = Z-2. All processors take Z as input and try to divide it by its associated divisor. So it is possible in one step to check if Z is prime. More realistically if N < Z-2 then a subset of divisors would be assigned to each processor.

AB +
CD

2.5.4 MIMD Computers

The MIMD (Multiple Instruction stream, Multiple Data stream) architecture is the most general and most powerful of the above classification. We have N processors, N streams of instructions and N streams of data, as shown in Figure 2.8. Machines (multiprocessors/multicomputers) of this architecture are composed of a set of processors that execute different programs which access their corresponding data sets. Each processor operates under the control of an instruction stream issued by its own control unit, i.e., each processor is capable of executing its own program on a

different data. This means that, the processors
asynchronously, typically, can be doing different thi
different data at the same time. As with SIMD com
communication of data or results between processors
via a shared memory or interconnection network.

The MIMD architectures have been a success because
are typically cheaper than special purpose SIMD mac
since they can be built with off-the-shelf components.

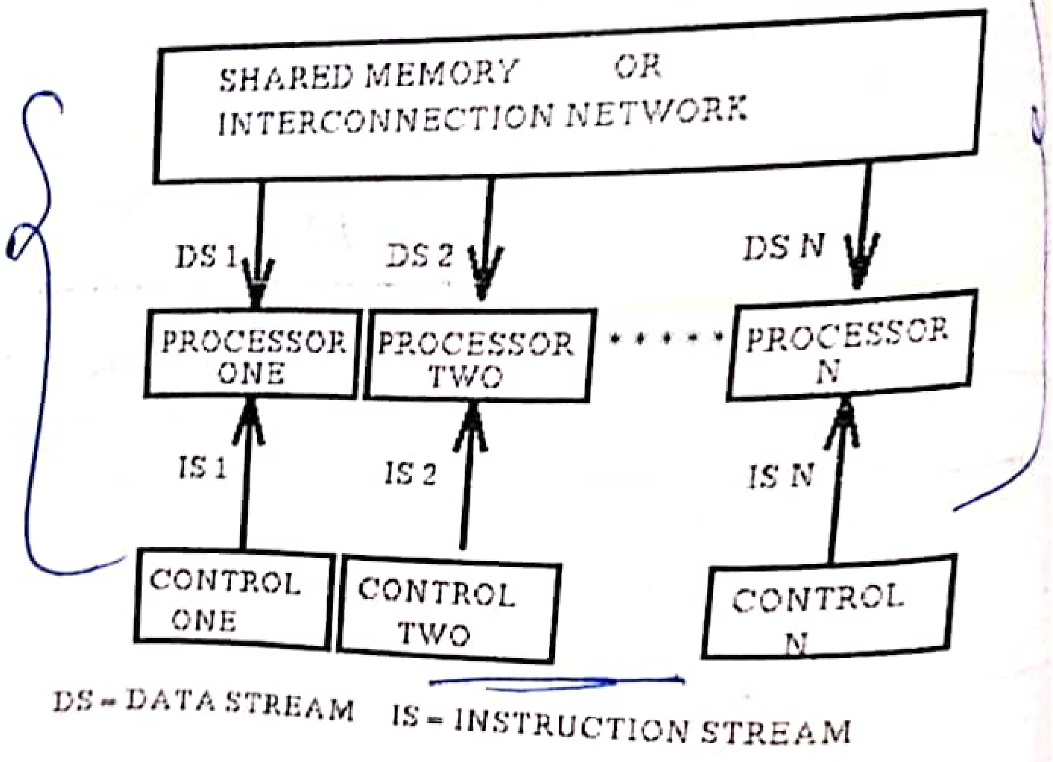


Figure 2.8: MIMD Computers

- **MIMD can be split into two classifications**

Multiprocessors - CPUs share a common memory

Multicomputers - CPUs have separate memories

- **It can be further subclassified as**

  Bus - All machines connected by single medium (e.g., bus, backplane, cable)

  Switched - Single wire from machine to machine, with possibly different wiring patterns (e.g, Internet)

- **Further classification is**

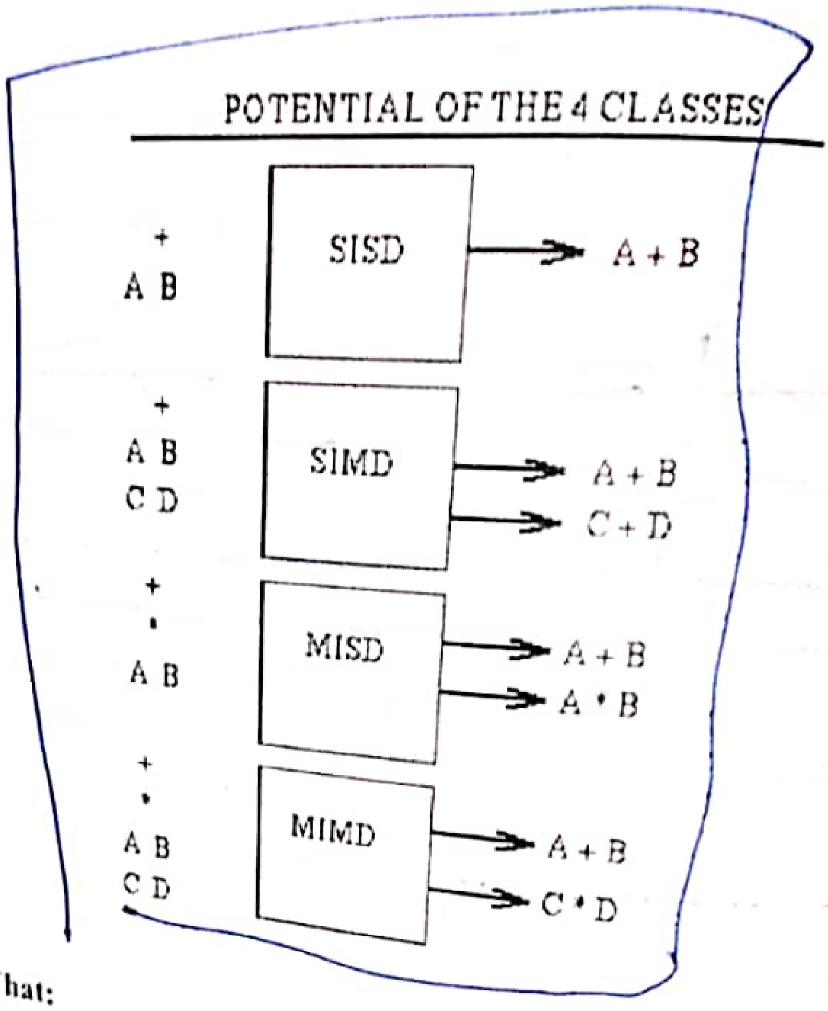
  1 **Tightly-coupled - short delay in communication between computers, high data rate (e.g., Parallel computers working on related computations)**

  2 **Loosely-coupled - Large delay in communications, Low data rate (Distributed Systems working on unrelated computations)**

MIMD computers with shared memory are known as multiprocessors or tightly coupled machines. Examples are ENCORE, MULTIMAX, SEQUENT & BALANCE. MIMD computers with distributed memory and an interconnection network are known as multicomputers or loosely coupled machines. Examples are INTEL iPSC, NCUBE/7 and transputer networks.

The majority of today's parallel computers belong to the MIMD (multiple instruction streams, multiple data stream) class. MIMD machines have multiple processors, which can execute one (or more) program(s) independently. Shared, distributed, and distributed shared memory multiprocessors as described above belong to this class. In this mode, the various processors execute different code on different data. This is the mode of operation in distributed systems as well as in the vast majority of parallel systems. There is no common clock among the system processors. Sun Ultra servers, multicomputer PCs, and IBM SP machines are examples of machines that execute in MIMD mode.

Now, we can summarize the different classe follows:



Note That:

Sometimes it may be necessary to have only a subset of the processors executes an instruction i.e. only some data needs to be operated on for that instruction. This information can be encoded in the instruction itself indicating whether

1. the processor is active (execute the instruction)
2. the processor is inactive (wait for the next instruction)

[illegible]t prob[illegible]s to be solved on SIMD (and MIMD) computers, it [illegible]ocessors to be able to communicate with each [illegible] data or results. This can be done in two ways: