Thread

- Thread is a sequential flow of control within a process.
- A process can contain one or more threads.
- Threads have their own program counter and register values, but they share the memory space and other resources of the process.

Thread

- Thread is basically a lightweight process .
- Advantages:
  - It takes less time to create and terminate a new thread than to create, and terminate a process.
  - It takes less time to switch between two threads within the same process .
  - Less communication overheads.

- Study of concurrent and parallel executions is important due to following reasons:
  - i) Some problems are most naturally solved by using a set of co-operating processes.
  - ii) To reduce the execution time.

- **Concurrent execution** is the temporal behavior of the N-client 1-server model .

- **Parallel execution** is associated with the N-client N-server model. It allows the servicing of more than one client at the same time as the number of servers is more than one.

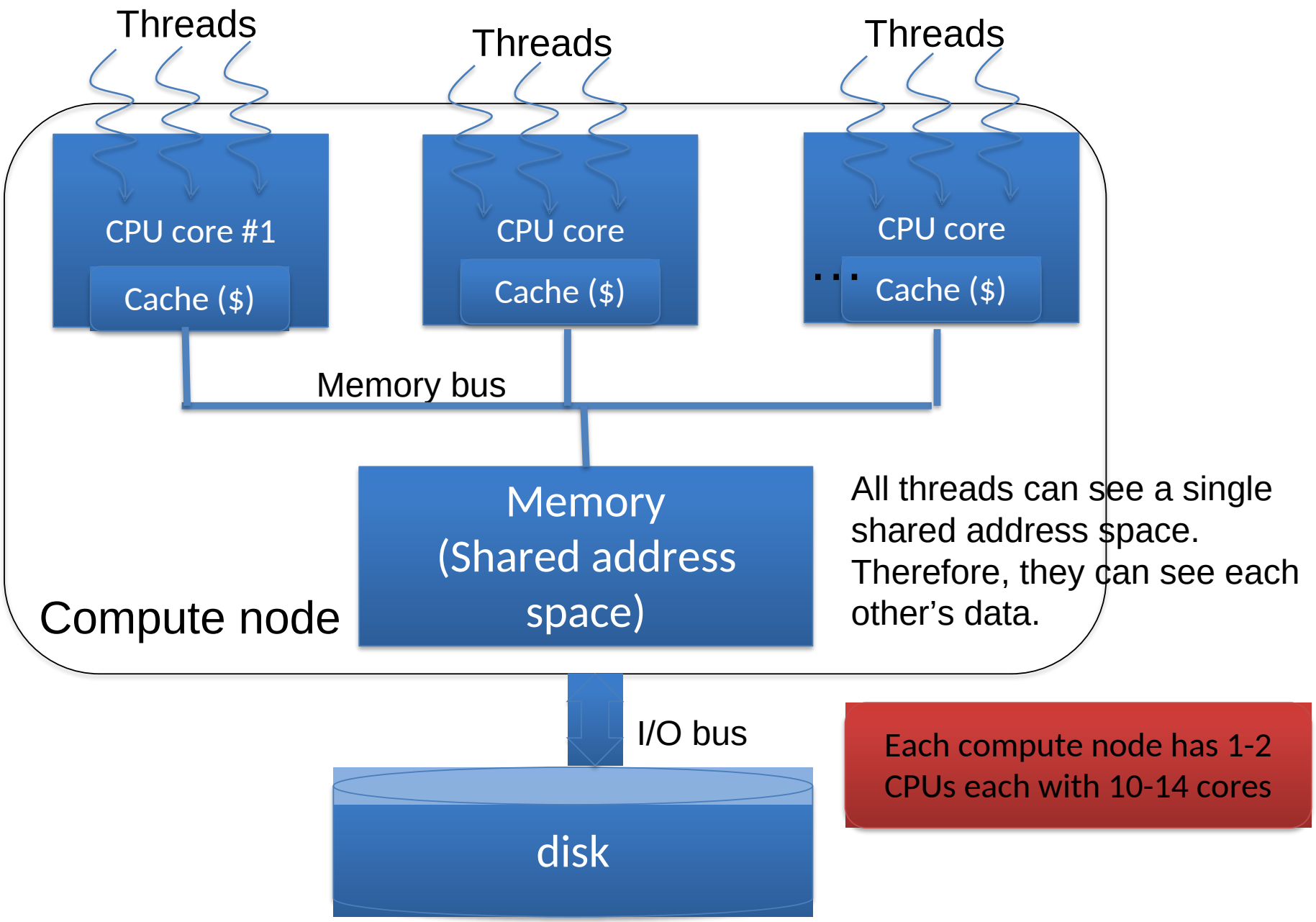
- **Granularity** refers to the **amount of computation done** in **parallel relative** to the **size** of the whole program.

- In parallel computing, granularity is a qualitative measure of the ratio of computation to communication.

Memory

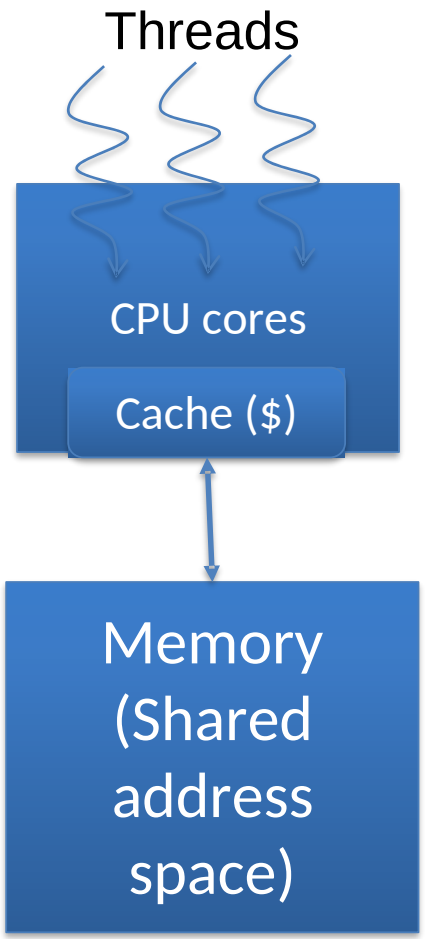
- Two major classes of parallel programming models:
  - Shared Memory
  - Distributed Memory

Shared Memory Architecture

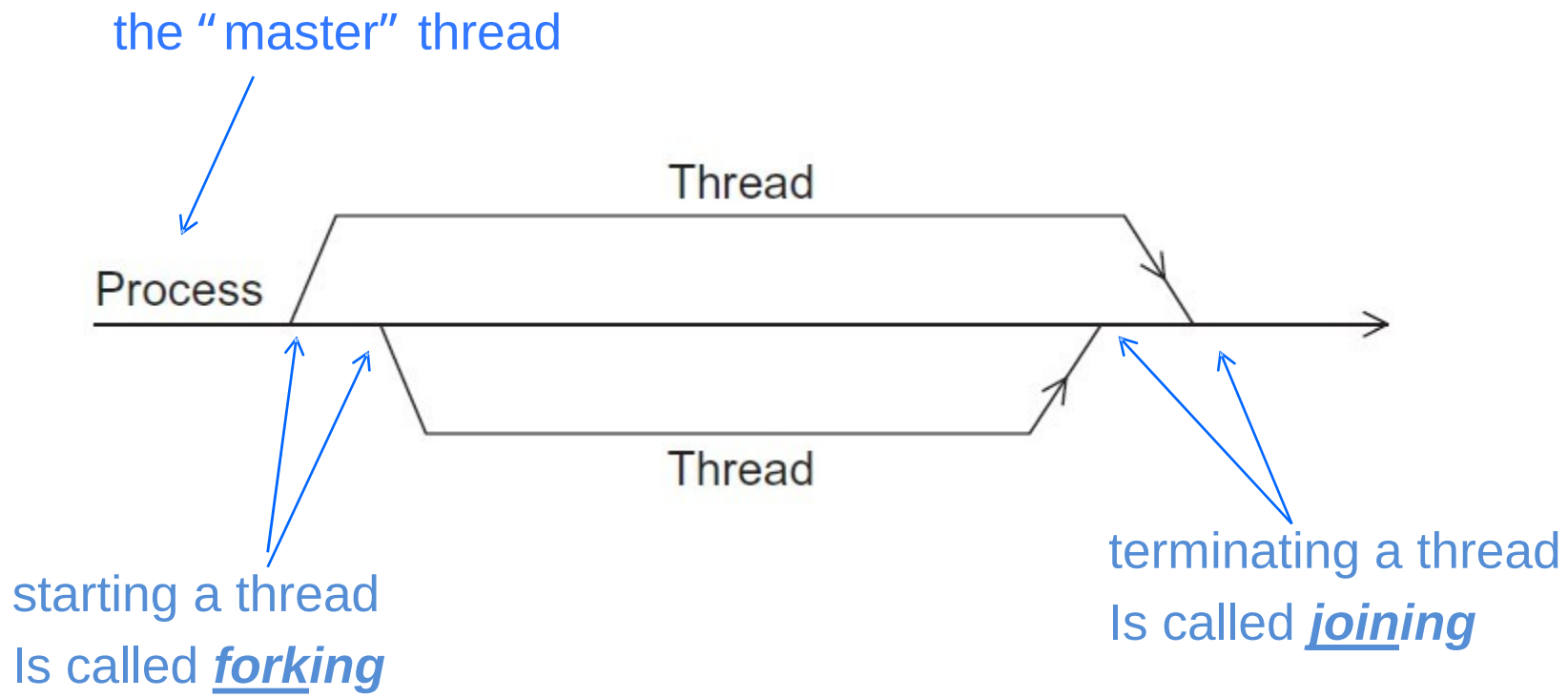


Multi-Threading (for shared memory architectures)

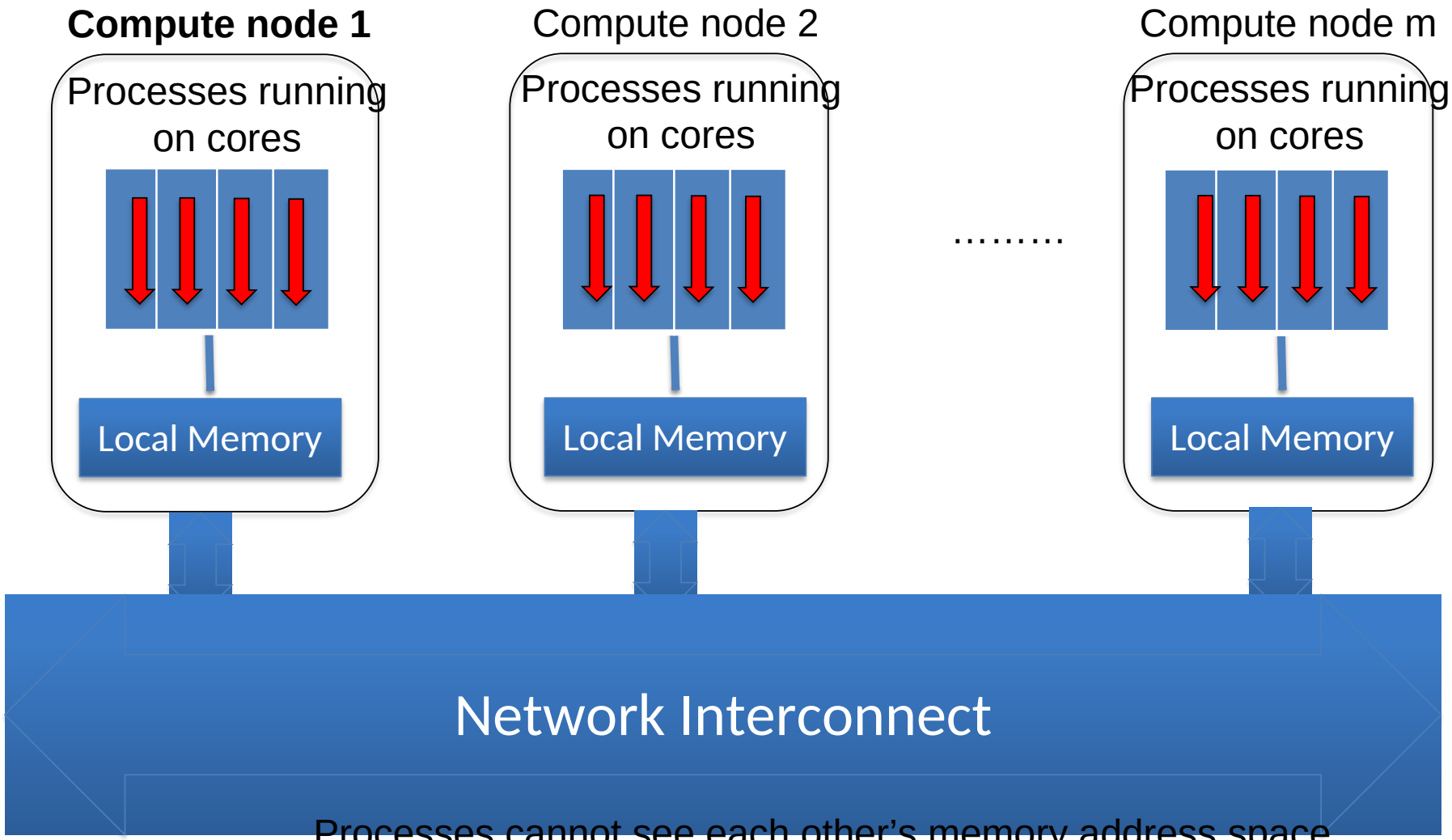
- Threads are contained within processes
  - One process => multiple threads
- All threads of a process share the same address space (in memory).
- Threads have the capability to run concurrently (executing different instructions and accessing different pieces of data at the same time)
- But if the resource is occupied by another thread, they form a queue and wait.
  - For maximum throughput, it is ideal to map each thread to a unique/distinct core



A process and two threads



Distributed Memory Architecture



Processes cannot see each other's memory address space.
They have to send inter-process messages (using MPI).

Distributed Memory System

- Clusters (most popular)
  - A collection of commodity systems.
  - Connected by a commodity interconnection network.
- Nodes of a cluster are individual computers joined by a communication network.

Kamiak provides an Infiniband interconnect between all compute nodes

a.k.a. hybrid systems

Characteristics of Parallel computer

Parallel computers can be characterized based on

- the data and instruction streams forming various types of computer organizations.
- the computer structure, e.g. multiple processors having separate memory or one shared global memory.
- size of instructions in a program called grain size.

TYPES OF CLASSIFICATION

1) Classification based on the instruction and data streams

2) Classification based on the structure of computers

3) Classification based on how the memory is accessed

4) Classification based on grain size

classification of parallel computers

- Flynn's classification based on instruction and data streams
- The Structural classification based on different computer organizations;
- The Handler's classification based on three distinct levels of computer:
  - Processor control unit (PCU), Arithmetic logic unit (ALU), Bit-level circuit (BLC)
- describe the sub-tasks or instructions of a program that can be executed in parallel based on the grain size.

FLYNN'S CLASSIFICATION

- Proposed by Michael Flynn in 1972.
- Introduced the concept of *instruction* and *data* streams for categorizing of computers.
- This classification is based on instruction and data streams
- Working of the instruction cycle.

Instruction Cycle

- The instruction cycle consists of a sequence of steps needed for the execution of an instruction in a program

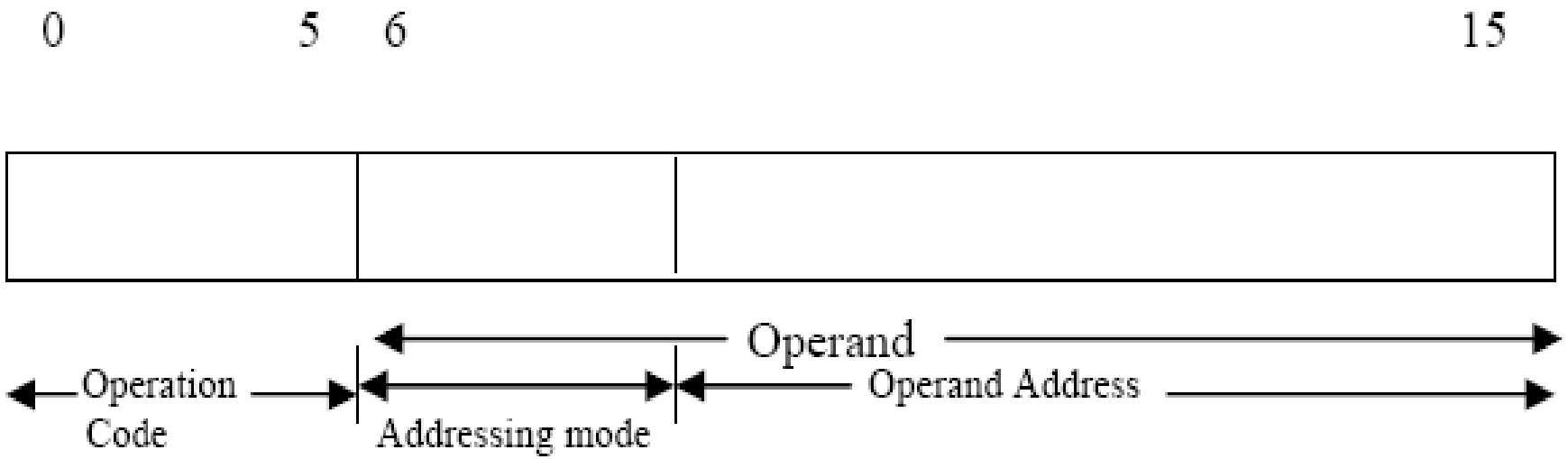


Figure 1: Opcode and Operand

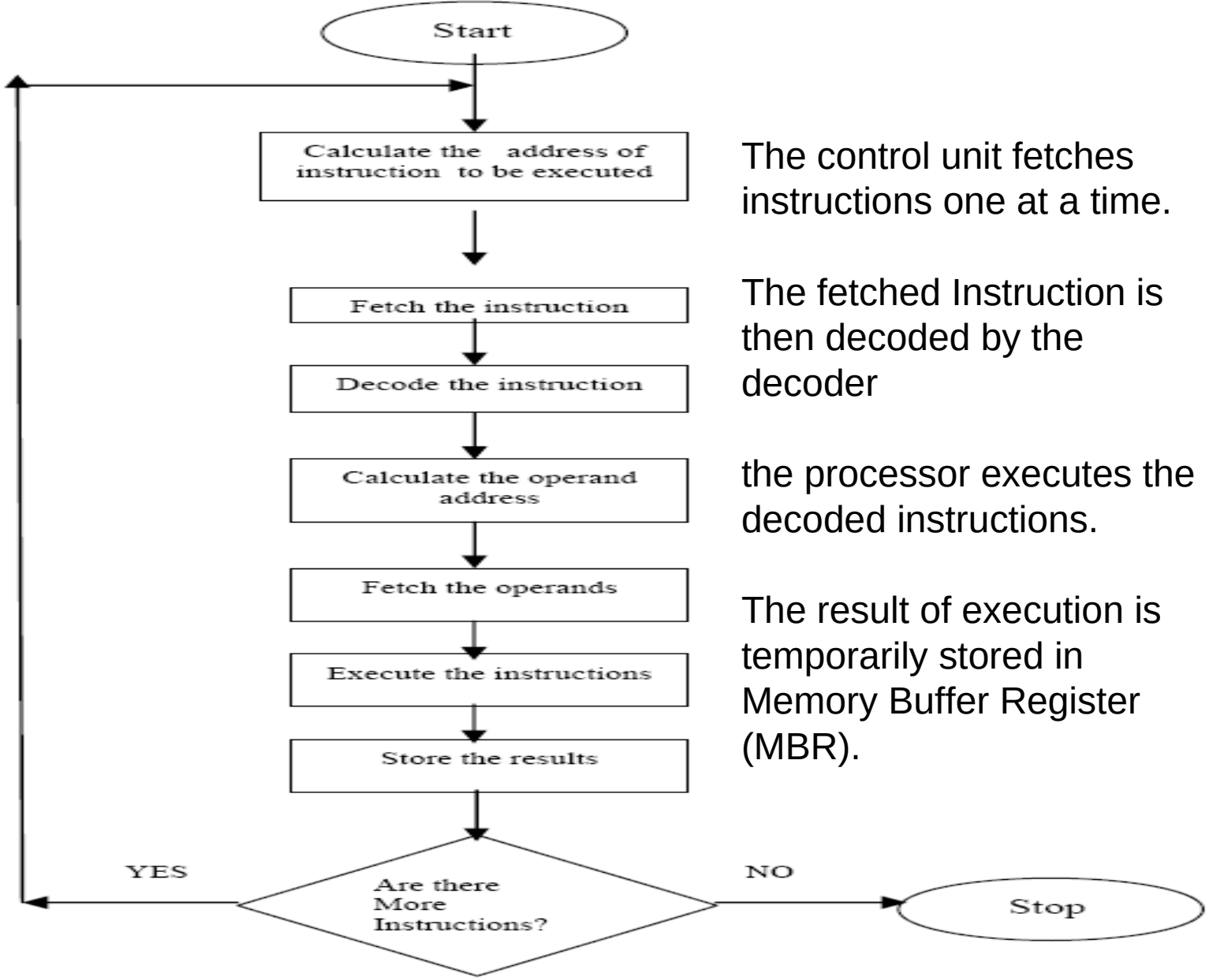


Figure 2: Instruction execution steps

The control unit fetches instructions one at a time.

The fetched Instruction is then decoded by the decoder

the processor executes the decoded instructions.

The result of execution is temporarily stored in Memory Buffer Register (MBR).

Instruction Stream and Data Stream

- flow of instructions is called ***instruction stream.***
- flow of operands between processor and memory is bi-directional. This flow of operands is called ***data stream***.

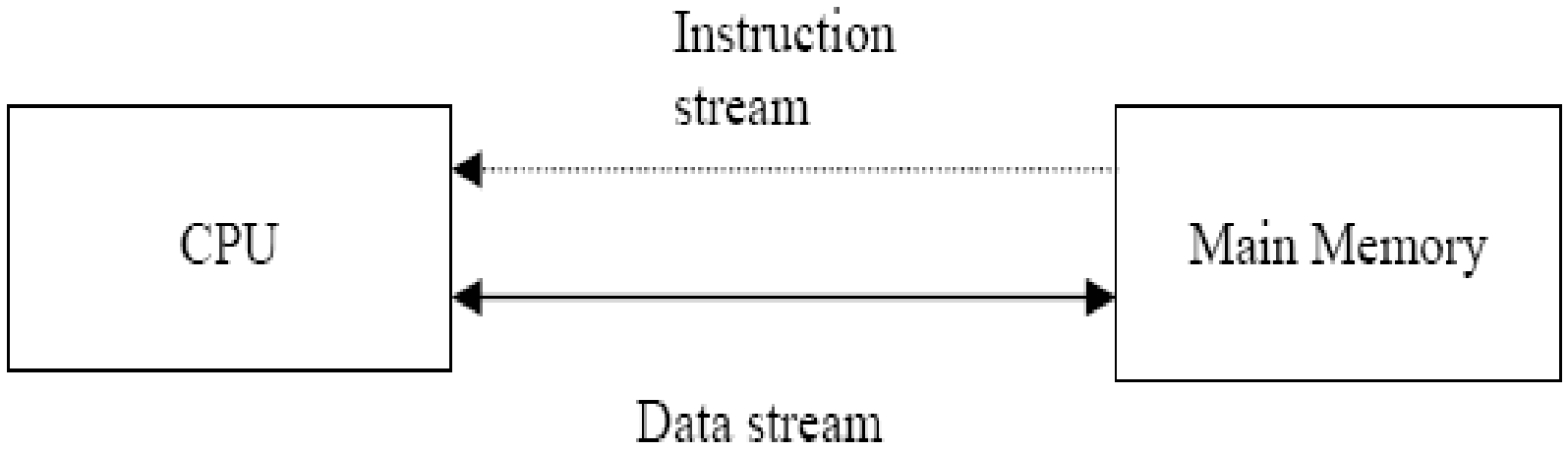


Figure 3: Instruction and data stream

Flynn's Classification

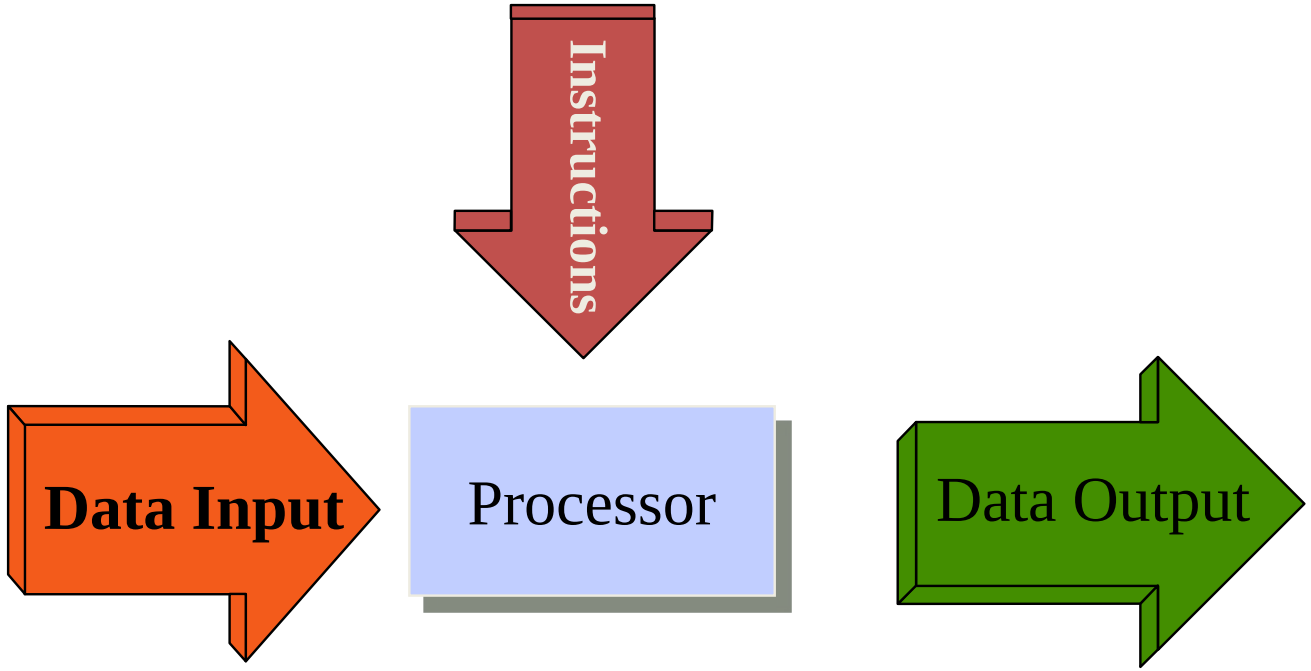
- Based on multiplicity of instruction streams and data streams observed by the CPU during program execution.

1) **Single Instruction and Single Data stream (SISD)**
2) **Single Instruction and multiple Data stream (SIMD)**
3) **Multiple Instruction and Single Data stream (MISD)**
4) **Multiple Instruction and Multiple Data stream (MIMD)**

Single Program Models: SIMD vs. MIMD

- **SP**: Single Program
  - Your parallel program is a single program that you execute on all threads (or processes)
- **SI**: Single Instruction
  - Each thread should be executing the same line of code at any given clock cycle.
- **MI**: Multiple Instruction
  - Each thread (or process) could be independently running a different line of your code (instruction) concurrently
- **MD**: Multiple Data
  - Each thread (or process) could be operating/accessing a different piece of the data from the memory concurrently

SISD : A Conventional Computer



- Speed is limited by the rate at which computer can transfer information internally.

Ex: PCs, Workstations

Single Instruction and Single Data stream (SISD)

- sequential execution of instructions is performed by one CPU containing a single processing element (PE)
- Therefore, SISD machines are conventional serial computers that process only one stream of instructions and one stream of data. Ex: Cray-1, CDC 6600, CDC 7600
-

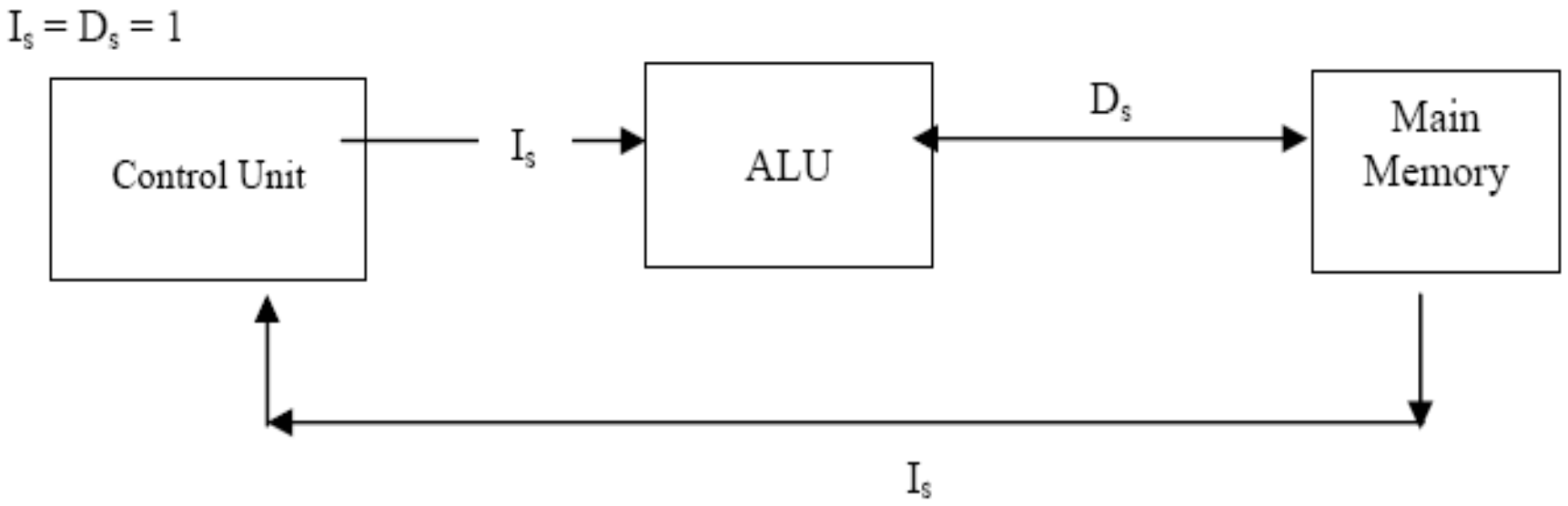
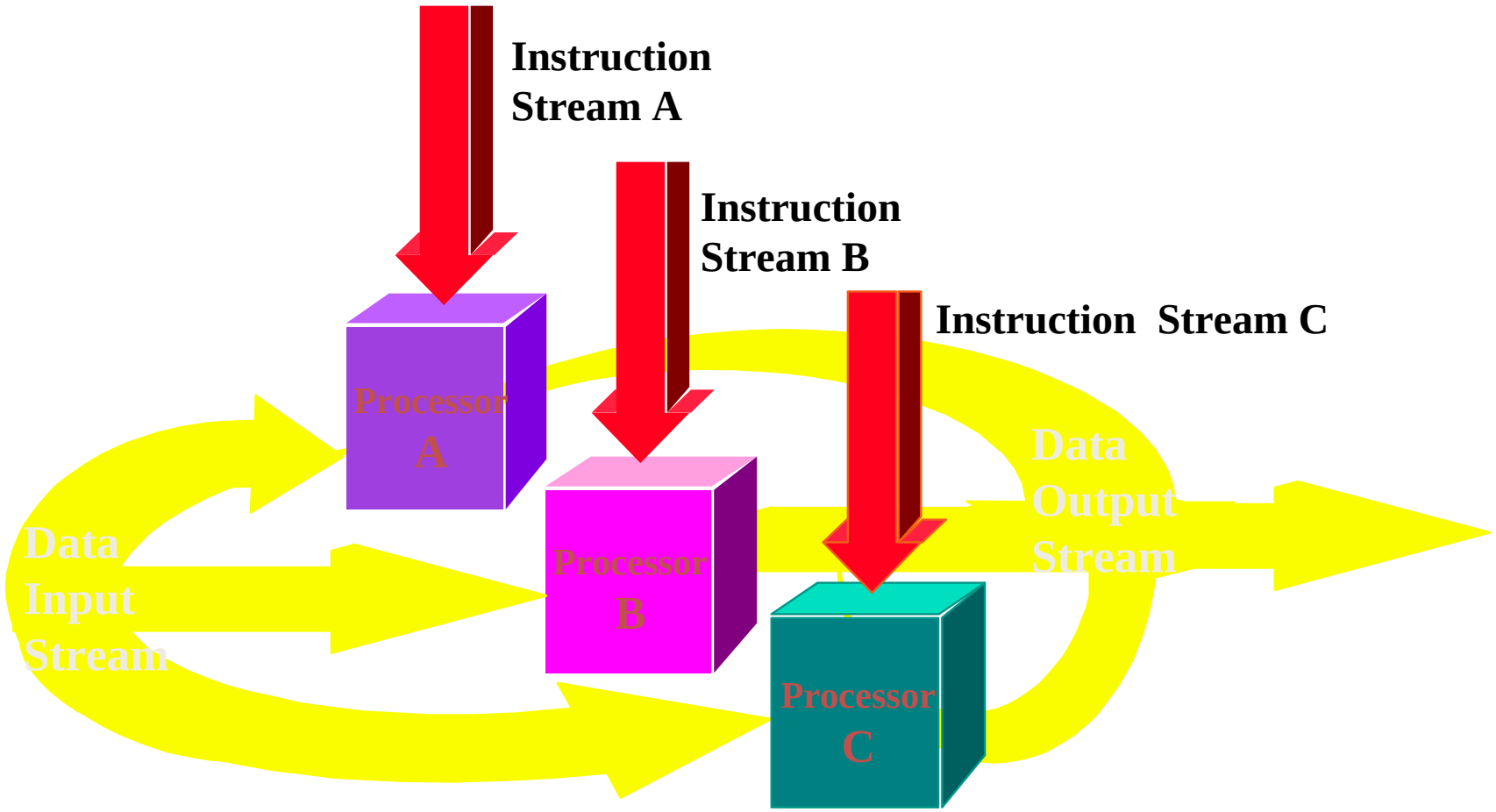


Figure 4: SISD Organisation

The MISD Architecture



- More of an intellectual exercise than a practical configuration. Few built, but commercially not available

Multiple Instruction and Single Data stream (MISD)

- multiple processing elements are organized under the control of multiple control units.
- Each control unit is handling one instruction stream and processed through its corresponding processing element.
- each processing element is processing only a single data stream at a time.
- Ex:C.mmp built by Carnegie-Mellon University.

$I_s > 1$
$D_s = 1$

All processing elements are interacting with the common shared memory for the organization of single data stream

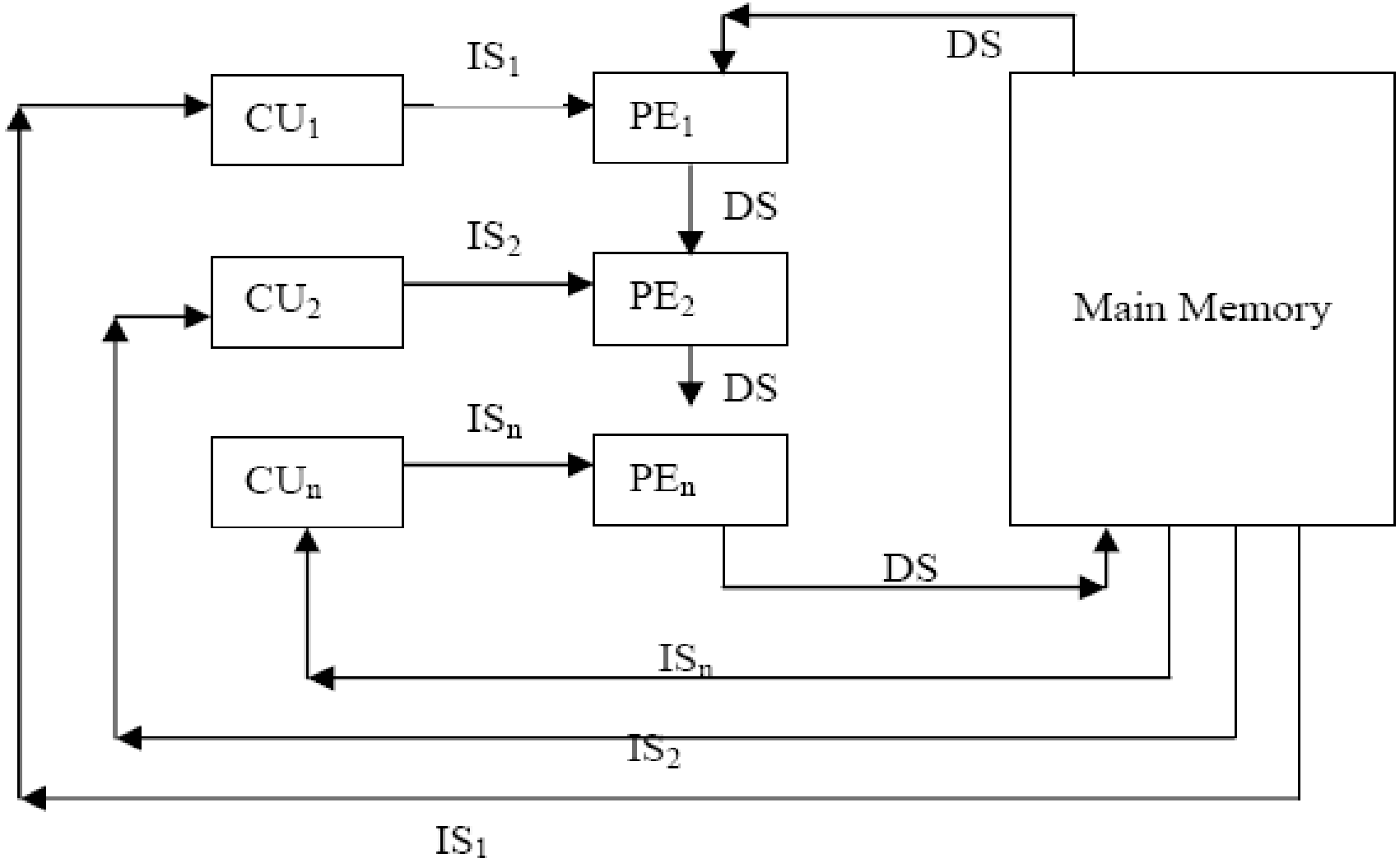


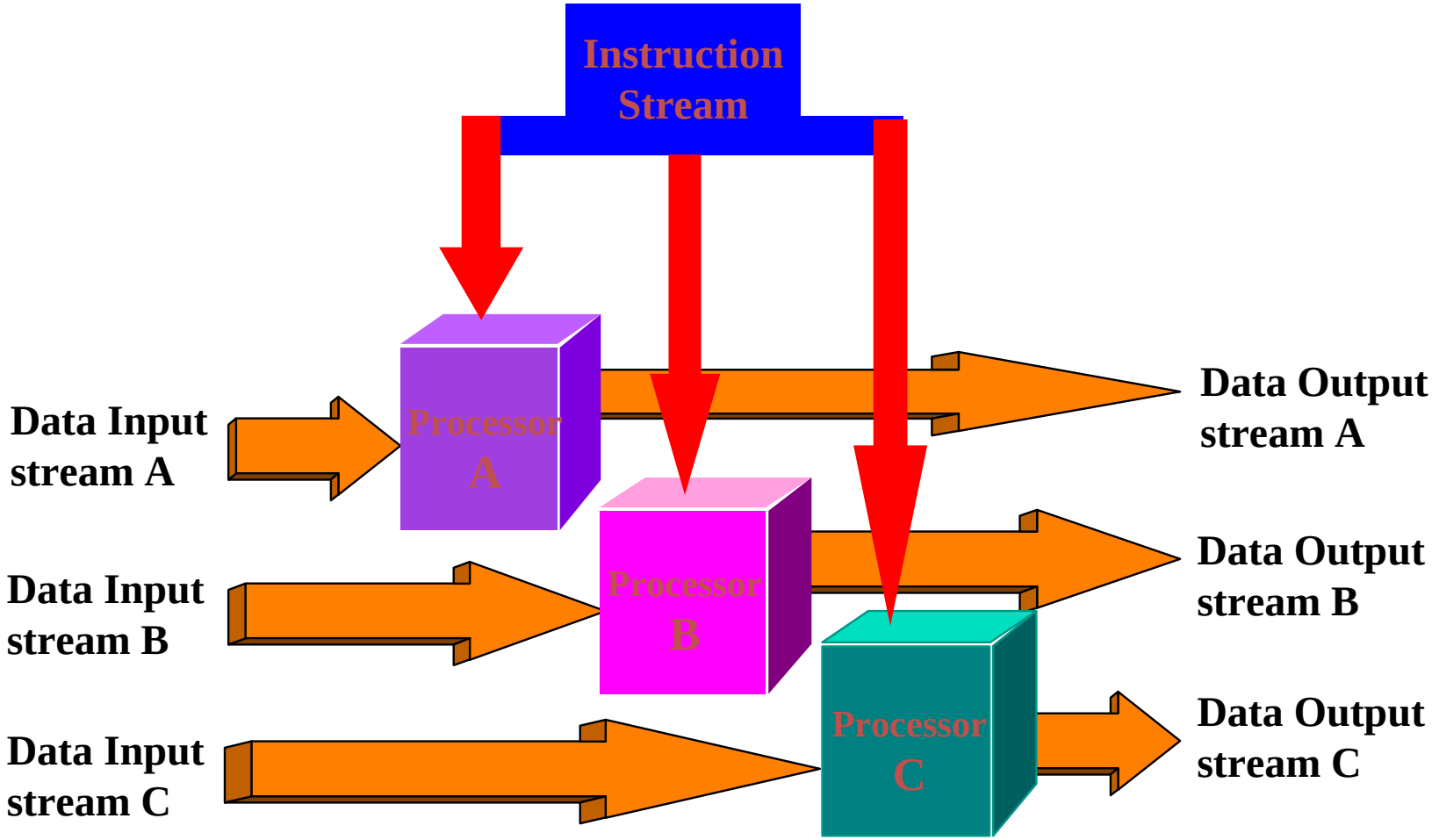
Figure 6: MISD Organisation

Advantages of MISD

– for the specialized applications like

- Real time computers need to be fault tolerant where several processors execute the same data for producing the redundant data.
- All these redundant data are compared as results which should be same otherwise faulty unit is replaced.
- Thus MISD machines can be applied to fault tolerant real time computers.

SIMD Architecture



$C_i <= A_i * B_i$

Ex: CRAY machine vector processing, Intel MMX (multimedia support)

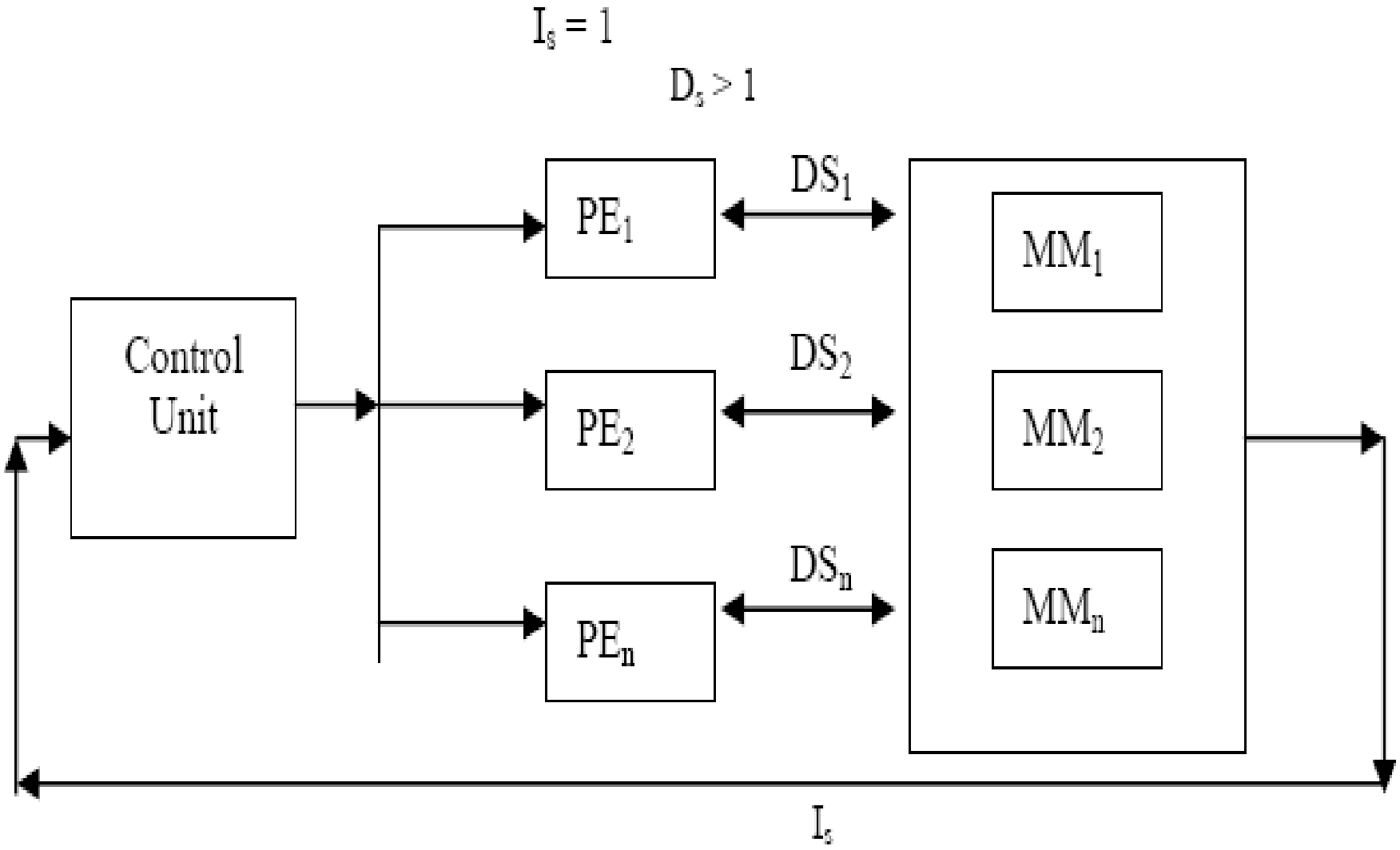


Figure 5: SIMD Organisation

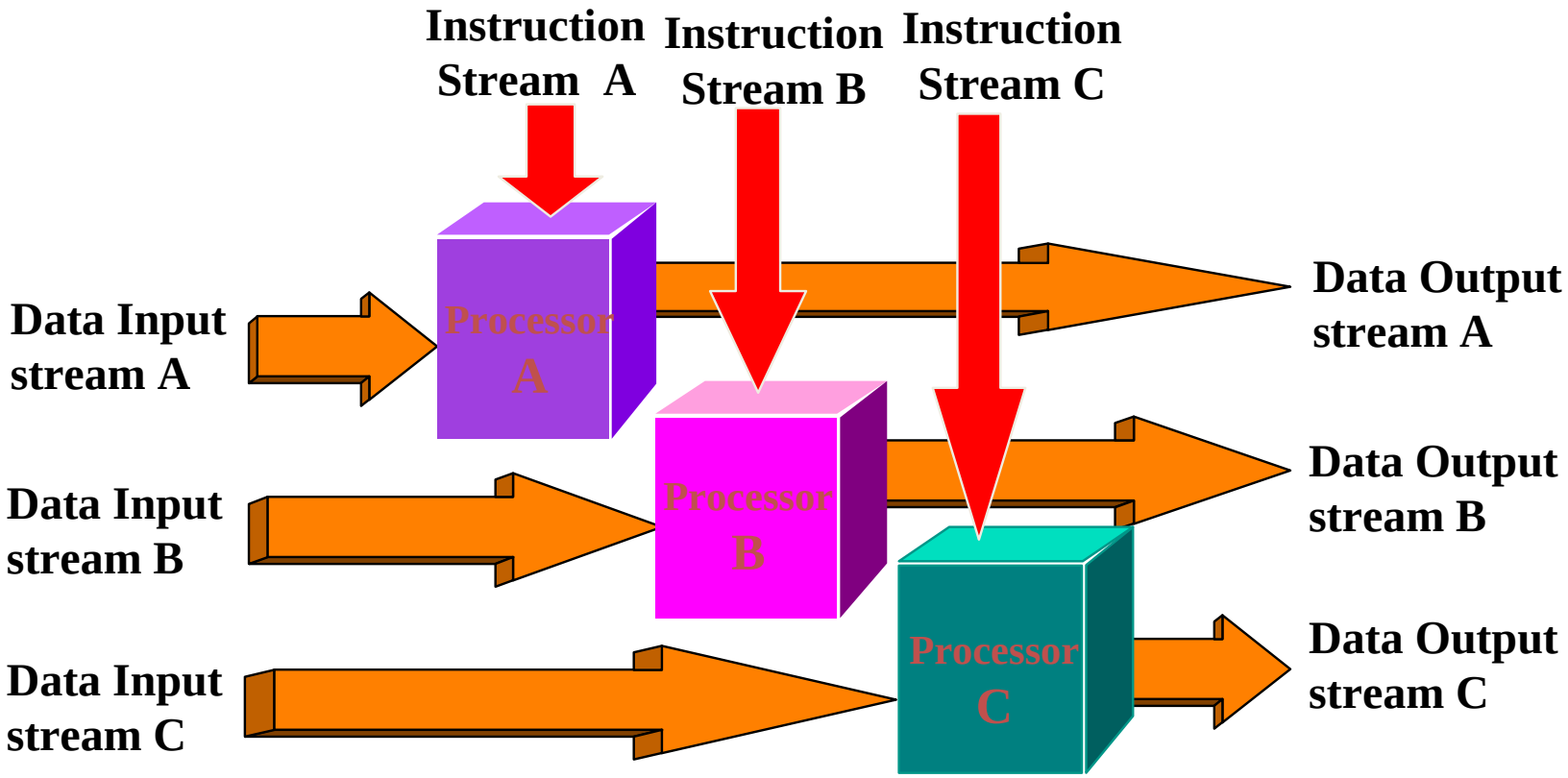
Single Instruction and multiple Data stream (SIMD)

- multiple processing elements work under the control of a single control unit.
- one instruction and multiple data stream.
- All the processing elements of this organization receive the same instruction broadcast from the CU.
- Main memory can also be divided into modules for generating multiple data streams.
- Every processor must be allowed to complete its instruction before the next instruction is taken for execution.
- The execution of instructions is synchronous

SIMD Processors

- Some of the earliest parallel computers such as the Illiac IV, MPP, DAP, CM-2 are belonged to this class of machines.
- Variants of this concept have found use in co-processing units such as the MMX units in Intel processors and IBM Cell processor.
- SIMD relies on the regular structure of computations (such as those in image processing).
- It is often necessary to selectively turn off operations on certain data items. For this reason, most SIMD programming architectures allow for an ``activity mask'', which determines if a processor should participate in a computation or not.

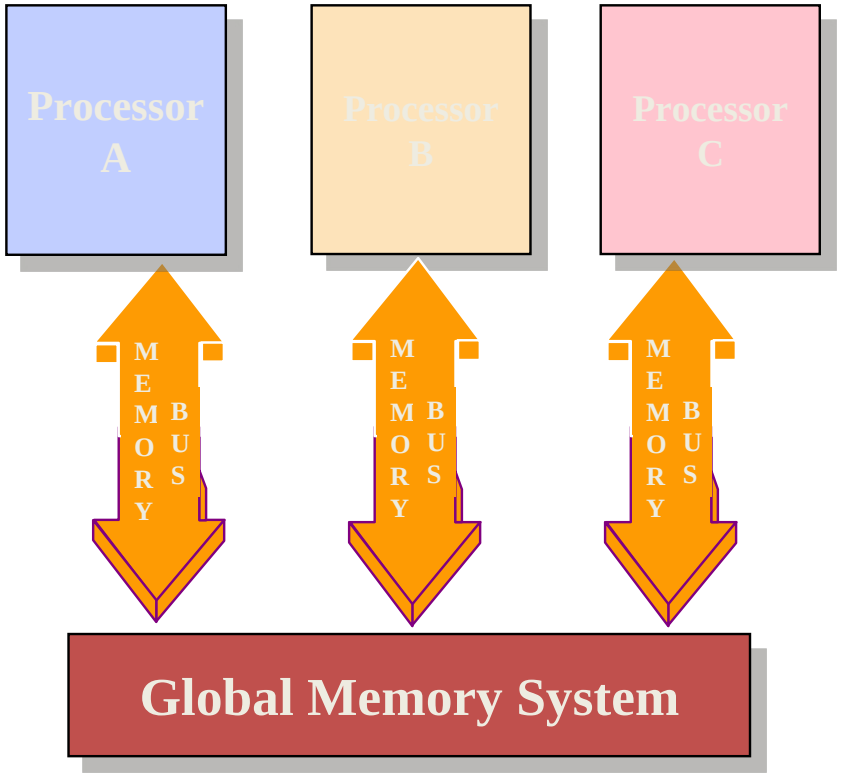
MIMD Architecture



Unlike SISD, MISD, MIMD computer works asynchronously.

- Shared memory (tightly coupled) MIMD
- Distributed memory (loosely coupled) MIMD

Shared Memory MIMD machine

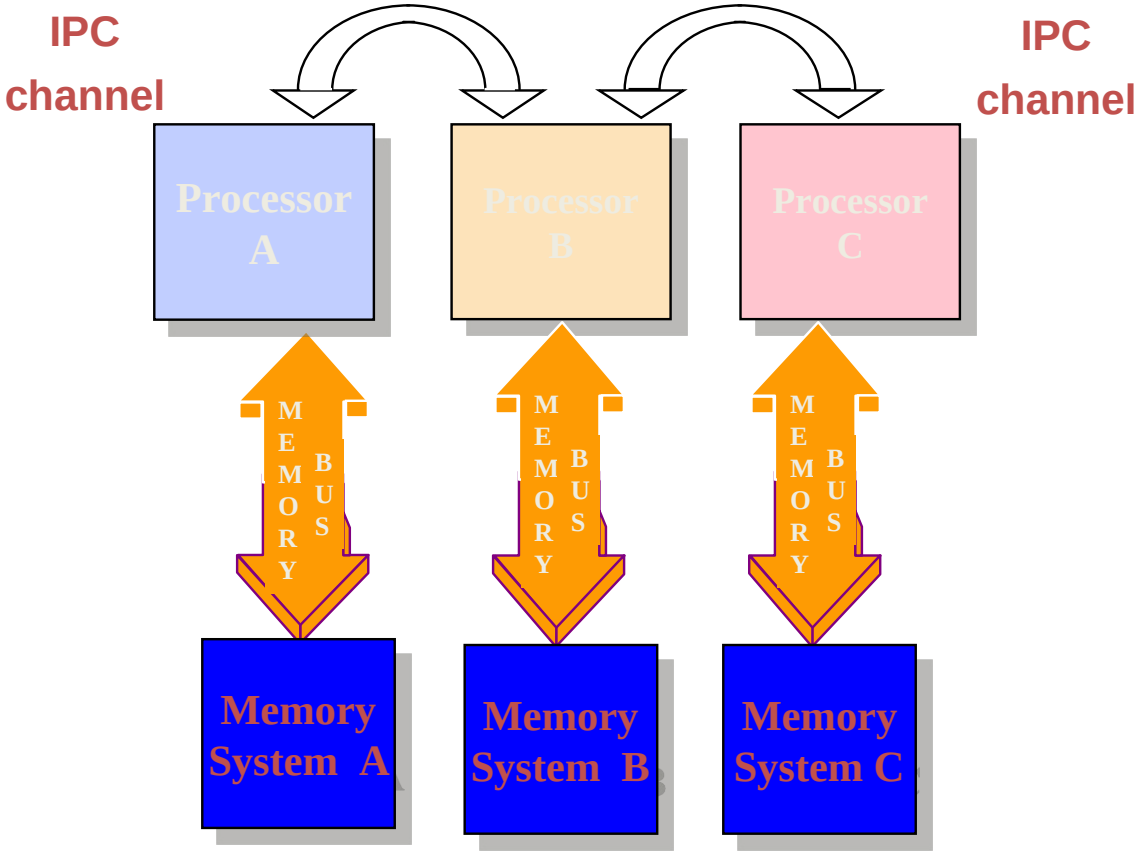


Communication : Source PE writes data to GM & destination retrieves it

- Easy to build, conventional OSes of SISD can be easily ported
- Limitation : reliability & expandability. A memory component or any processor failure affects the whole system.
- Increase of processors leads to memory contention.

  Ex. : Silicon graphics supercomputers....

Distributed Memory MIMD



- Communication : IPC (Inter-Process Communication) via High Speed Network.
- Network can be configured to … Tree, Mesh, Cube, etc.
- Unlike Shared MIMD
  - **easily/ readily expandable**
  - **Highly reliable (any CPU failure does not affect the whole system)**

MIMD Processors

- In contrast to SIMD processors, MIMD processors can execute different programs on different processors.
- A variant of this, called single program multiple data streams (SPMD) executes the same program on different processors.
- It is easy to see that SPMD and MIMD are closely related in terms of programming flexibility and underlying architectural support.
- Examples of such platforms include current generation Sun Ultra Servers, SGI Origin Servers, multiprocessor PCs, workstation clusters.

Multiple Instruction and Multiple Data stream (MIMD)

- multiple processing elements and multiple control units are organized as in MISD.
- for handling multiple instruction streams, multiple control units are there and For handling multiple data streams, multiple processing elements are organized.
- The processors work on their own data with their own instructions.
- Tasks executed by different processors can start or finish at different times.

- in the real sense MIMD organization is said to be a **Parallel computer.**

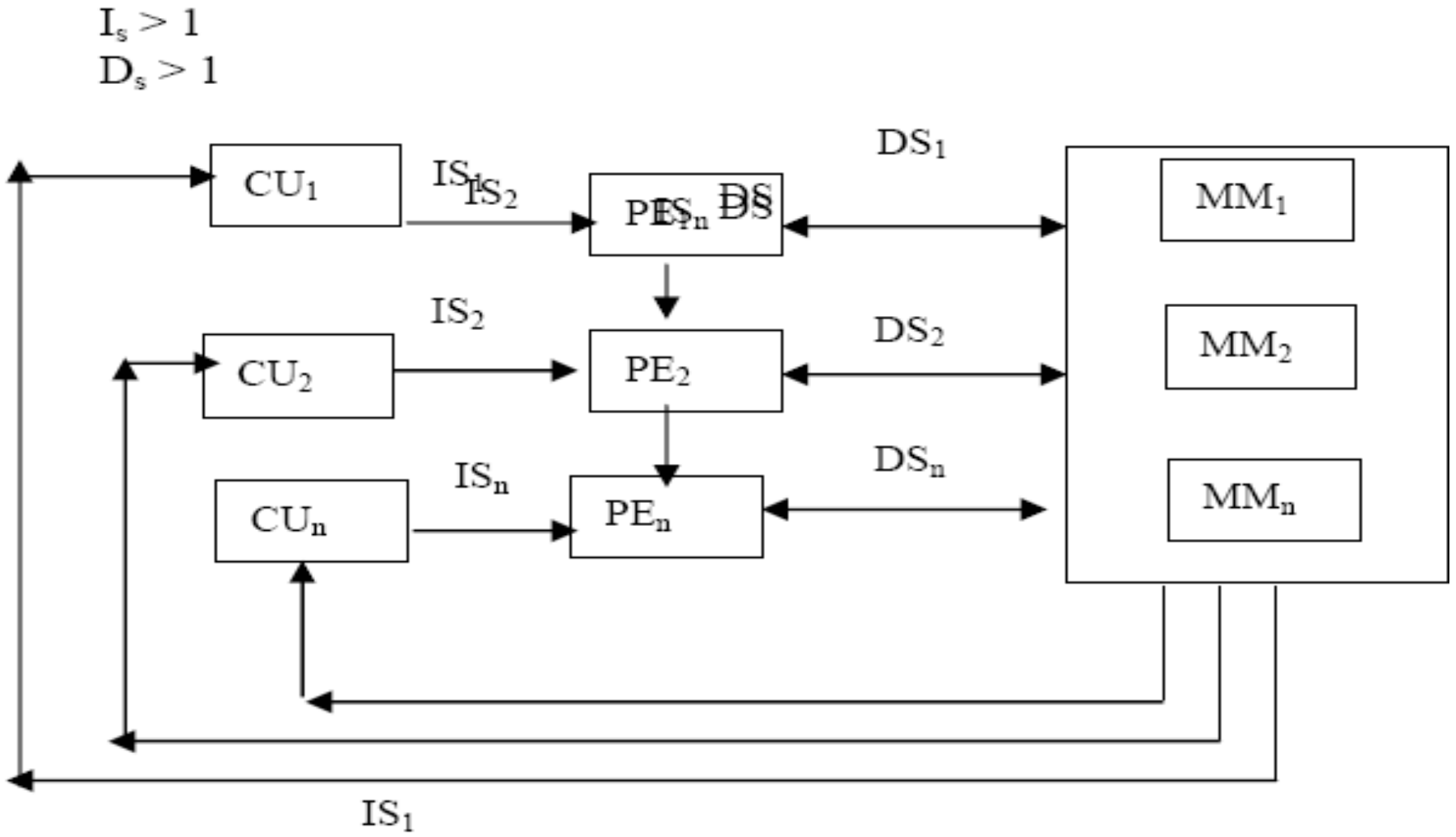


Figure 7: MIMD Organisation

- All multiprocessor systems fall under this classification.
- Examples :C.mmp, Cray-2, Cray X-MP, IBM 370/168 MP, Univac 1100/80, IBM 3081/3084.
- MIMD organization is the most popular for a parallel computer.
- In the real sense, parallel computers execute the instructions in MIMD mode

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 paradigm can be built from inexpensive off-the-shelf components with relatively little effort in a short amount of time.

HANDLER'S CLASSIFICATION

- In 1977, Handler proposed an elaborate notation for expressing the pipelining and parallelism of computers.

- Handler's classification addresses the computer at three distinct levels:

  - Processor control unit (PCU)---- CPU
  - Arithmetic logic unit (ALU)--- processing element
  - Bit-level circuit (BLC)--- logic circuit .

Way to describe a computer

- **Computer = (p * p', a * a', b * b')**

- Where p = number of PCUs
  - p'= number of PCUs that can be pipelined
  - a = number of ALUs controlled by each PCU
  - a'= number of ALUs that can be pipelined
  - b = number of bits in ALU or processing element (PE) word
  - b'= number of pipeline segments on all ALUs or in a single PE

Relationship between various elements of the computer

- The '*' operator is used to indicate that the units are pipelined or macro-pipelined with a stream of data running through all the units.

- The '+' operator is used to indicate that the units are not pipelined but work on independent streams of data.

- The 'v' operator is used to indicate that the computer hardware can work in one of several modes.

- The '~' symbol is used to indicate a range of values for any one of the parameters.

Ex:

- The CDC 6600 has a single main processor supported by 10 I/O processors. One control unit coordinates one ALU with a 60-bit word length. The ALU has 10 functional units which can be formed into a pipeline. The 10 peripheral I/O processors may work in parallel with each other and with the CPU. Each I/O processor contains one 12-bit ALU.

CDC 6600I/O = (10, 1, 12)

- The description for the main processor is:

CDC 6600main = (1, 1 * 10, 60)

- The main processor and the I/O processors can be regarded as forming a macro-pipeline so the '*' operator is used to combine the two structures:

CDC 6600 = (I/O processors) * (central processor) =

(10, 1, 12) * (1, 1 * 10, 60)

STRUCTURAL CLASSIFICATION

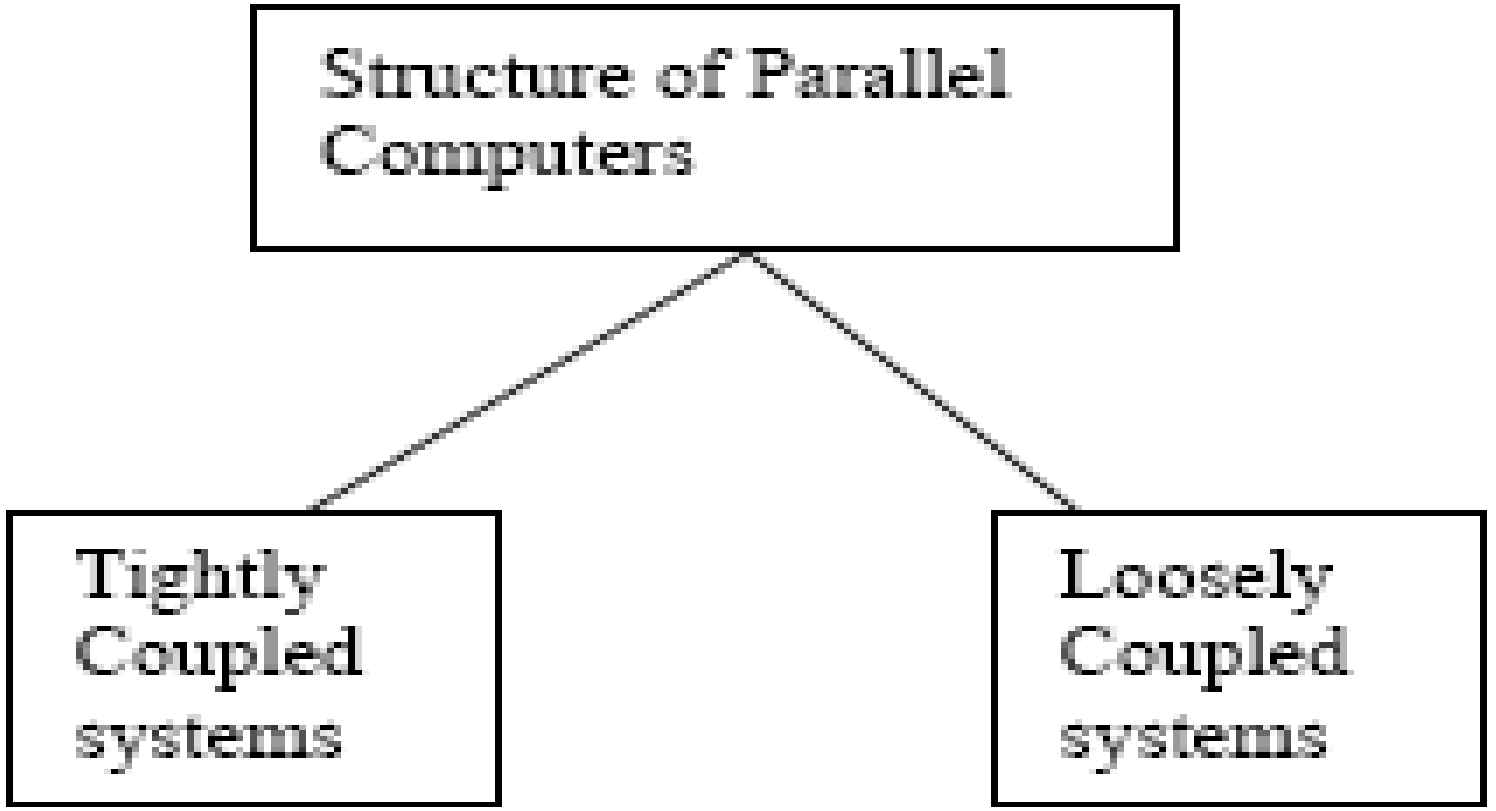


Figure 8: Structural classification

STRUCTURAL CLASSIFICATION

- a parallel computer (MIMD) can be characterized as a set of multiple processors and shared memory or memory modules communicating via an interconnection network.

- When multiprocessors communicate through the global shared memory modules then this organization is called **Shared memory computer** or **Tightly coupled systems**

- Shared memory multiprocessors have the following characteristics:
  - Every processor communicates through a shared global memory
  - For high speed real time processing, these systems are preferable as their throughput is high as compared to loosely coupled systems.

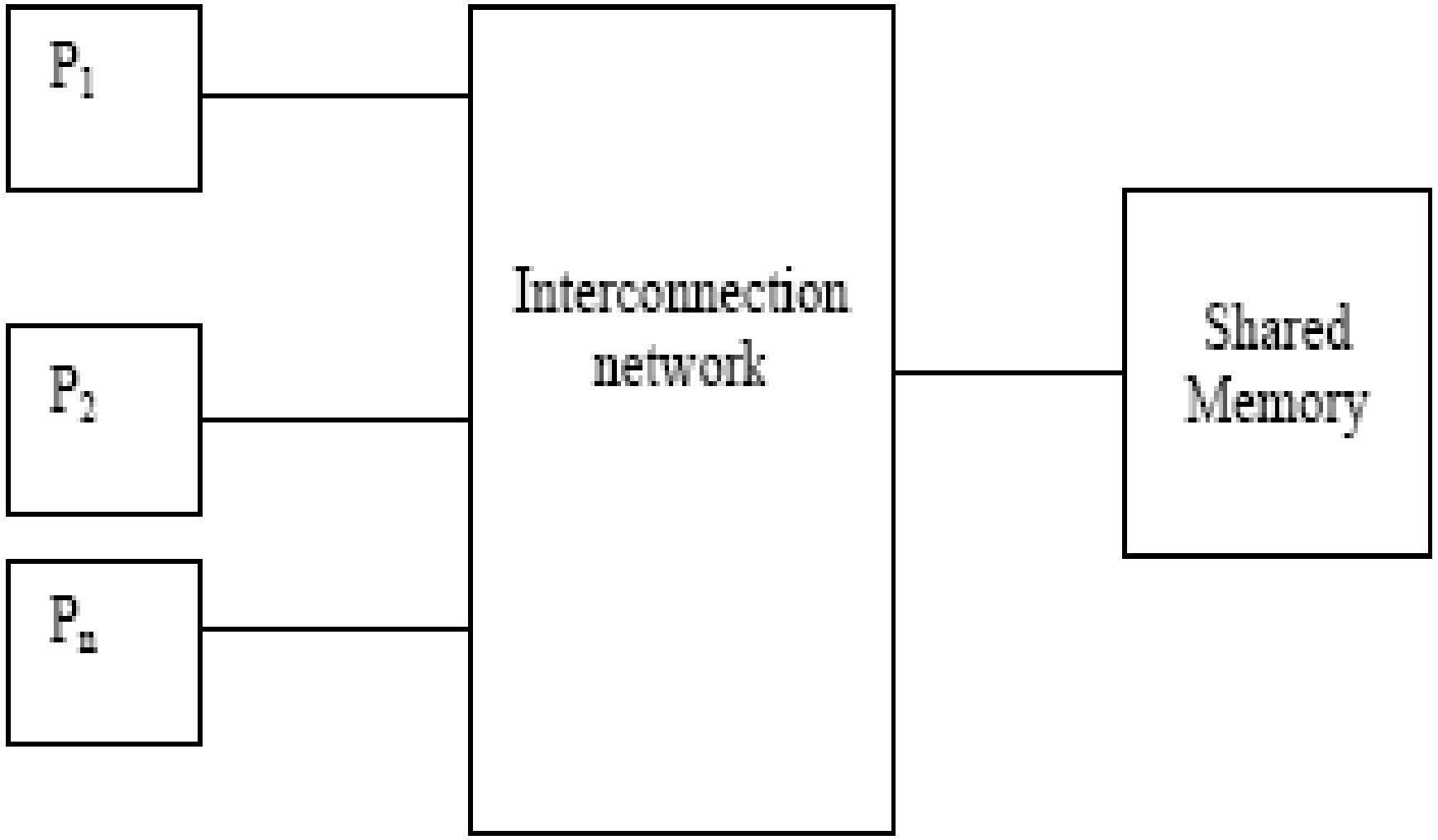


Figure 9: Tightly coupled system

- In tightly coupled system organization, multiple processors share a global main memory, which may have many modules.

- The processors have also access to I/O devices. The inter- communication between processors, memory, and other devices are implemented through various interconnection networks,

Types of Interconnection n/w

- **Processor-Memory Interconnection Network (PMIN)**
  - This is a switch that connects various processors to different memory modules.
- **Input-Output-Processor Interconnection Network (IOPIN)**
  - This interconnection network is used for communication between processors and I/O channels
- **Interrupt Signal Interconnection Network (ISIN)**
  - When a processor wants to send an interruption to another processor, then this interrupt first goes to ISIN, through which it is passed to the destination processor. In this way, synchronization between processor is implemented by ISIN.

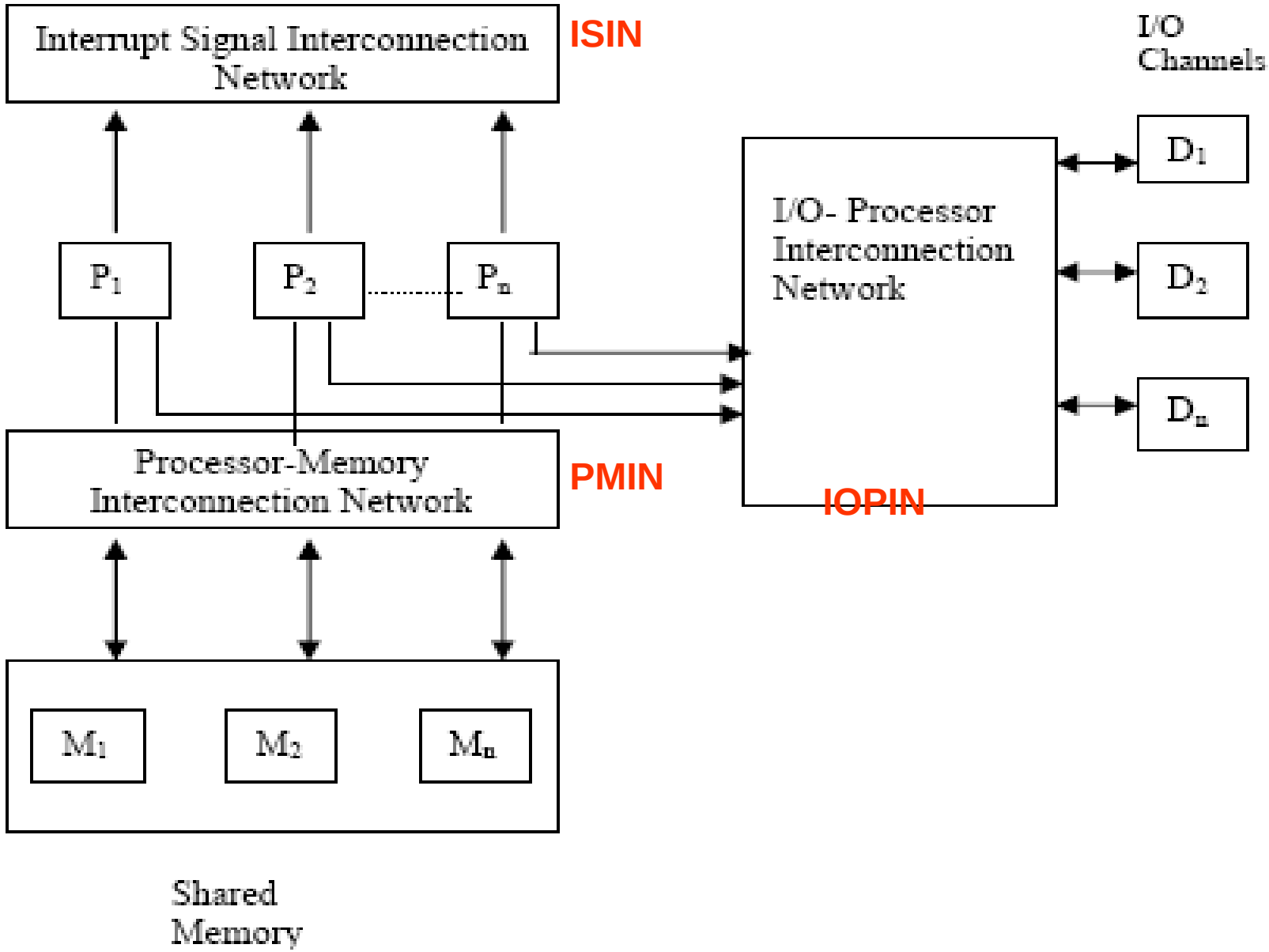
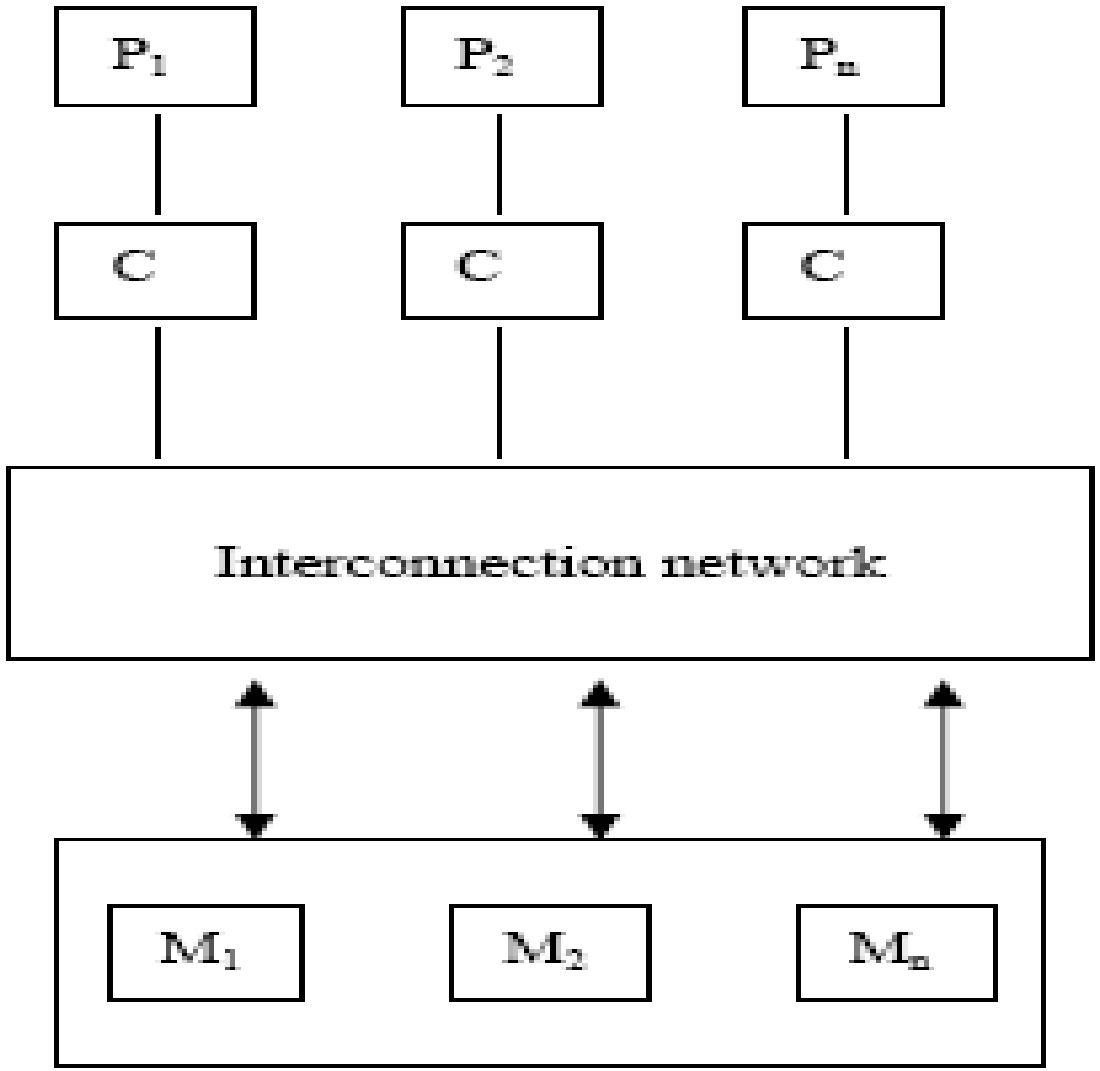


Figure 11: Tightly coupled system organization

– To reduce this delay, every processor may use cache memory for the frequent references made by the processor as



re 12: Tightly coupled systems with cache memory

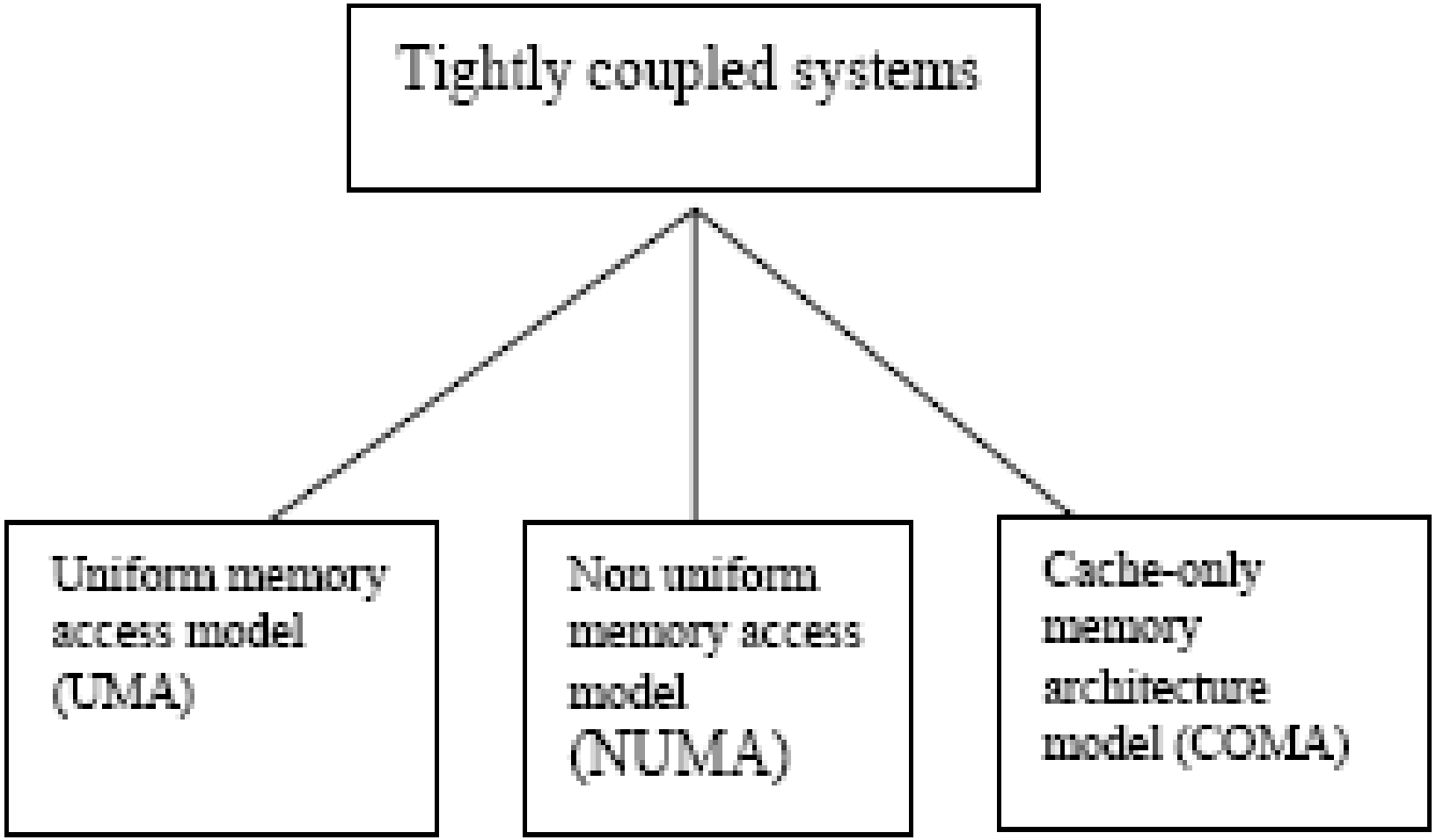


Figure 13: Modes of Tightly coupled systems

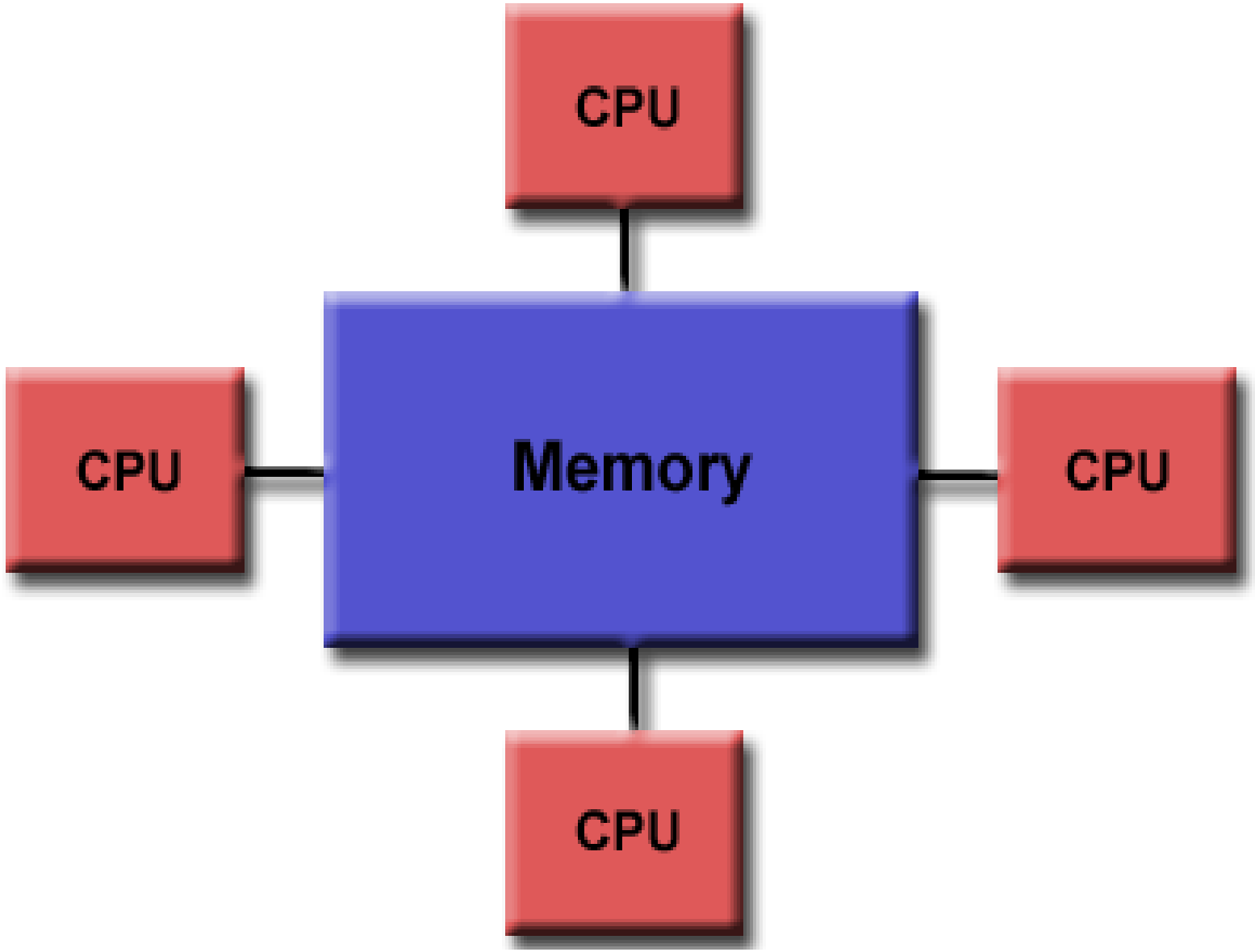
Uniform Memory Access Model (UMA)

- In this model, main memory is uniformly shared by all processors in multiprocessor systems and each processor has equal access time to shared memory.

- This model is used for time-sharing applications in a multi user environment

Uniform Memory Access (UMA):

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines
- Identical processors
- Equal access and access times to memory
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.


CPU
CPU
Memory
CPU
CPU

Non-Uniform Memory Access Model (NUMA)

- In shared memory multiprocessor systems, local memories can be connected with every processor. The collection of all local memories form the global memory being shared.
- global memory is distributed to all the processors .
- In this case, the access to a local memory is uniform for its corresponding processor ,but if one reference is to the local memory of some other remote processor, then the access is not uniform.
- It depends on the location of the memory. Thus, all memory words are not accessed uniformly.

Non-Uniform Memory Access (NUMA):

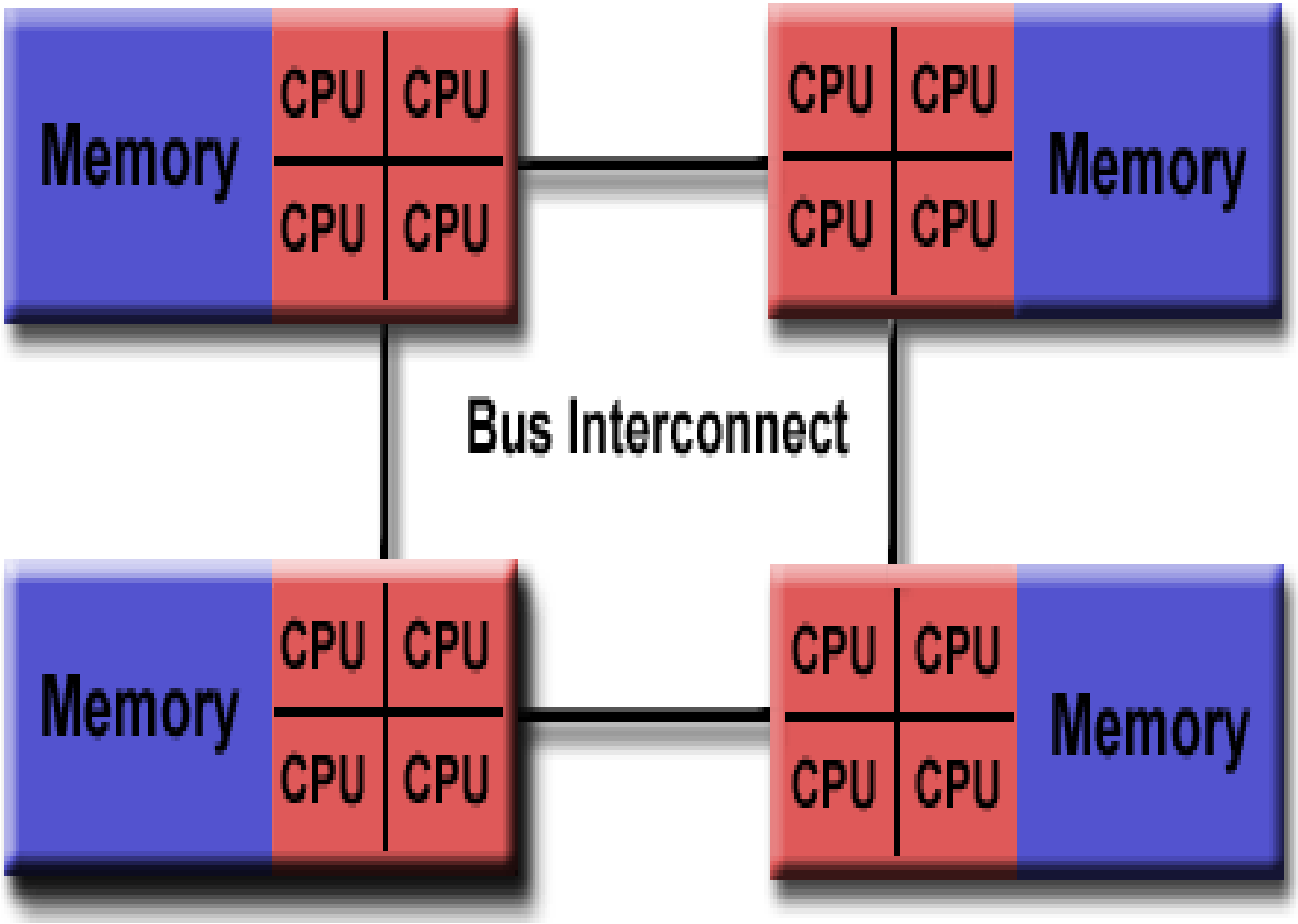
- Often made by physically linking two or more SMPs
- One SMP can directly access memory of another SMP
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

Advantages: Global address space provides a user-friendly programming perspective to memory

- Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs

Disadvantages:

- Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increases traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.
- Programmer responsibility for synchronization constructs that ensure "correct" access of global memory.
- Expense: it becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing numbers of processors


Memory
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
Memory
Bus Interconnect
Memory
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
Memory

Cache-Only Memory Access Model (COMA)

- shared memory multiprocessor systems may use cache memories with every processor for reducing the execution time of an instruction .

Loosely coupled system

- when every processor in a multiprocessor system, has its own local memory and the processors communicate via messages transmitted between their local memories, then this organization is called **Distributed memory computer** or **Loosely coupled system**

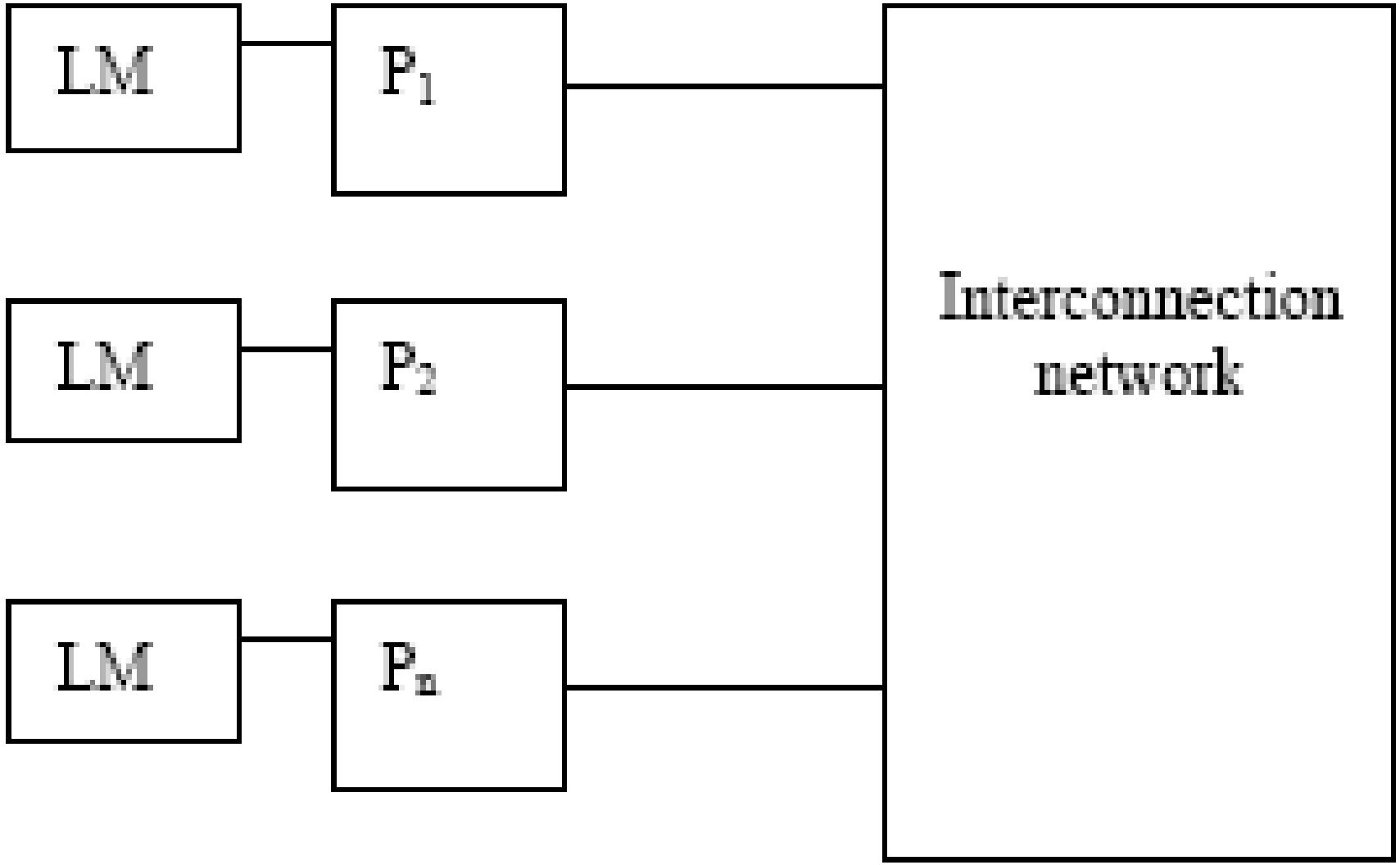


Figure 10 Loosely coupled system

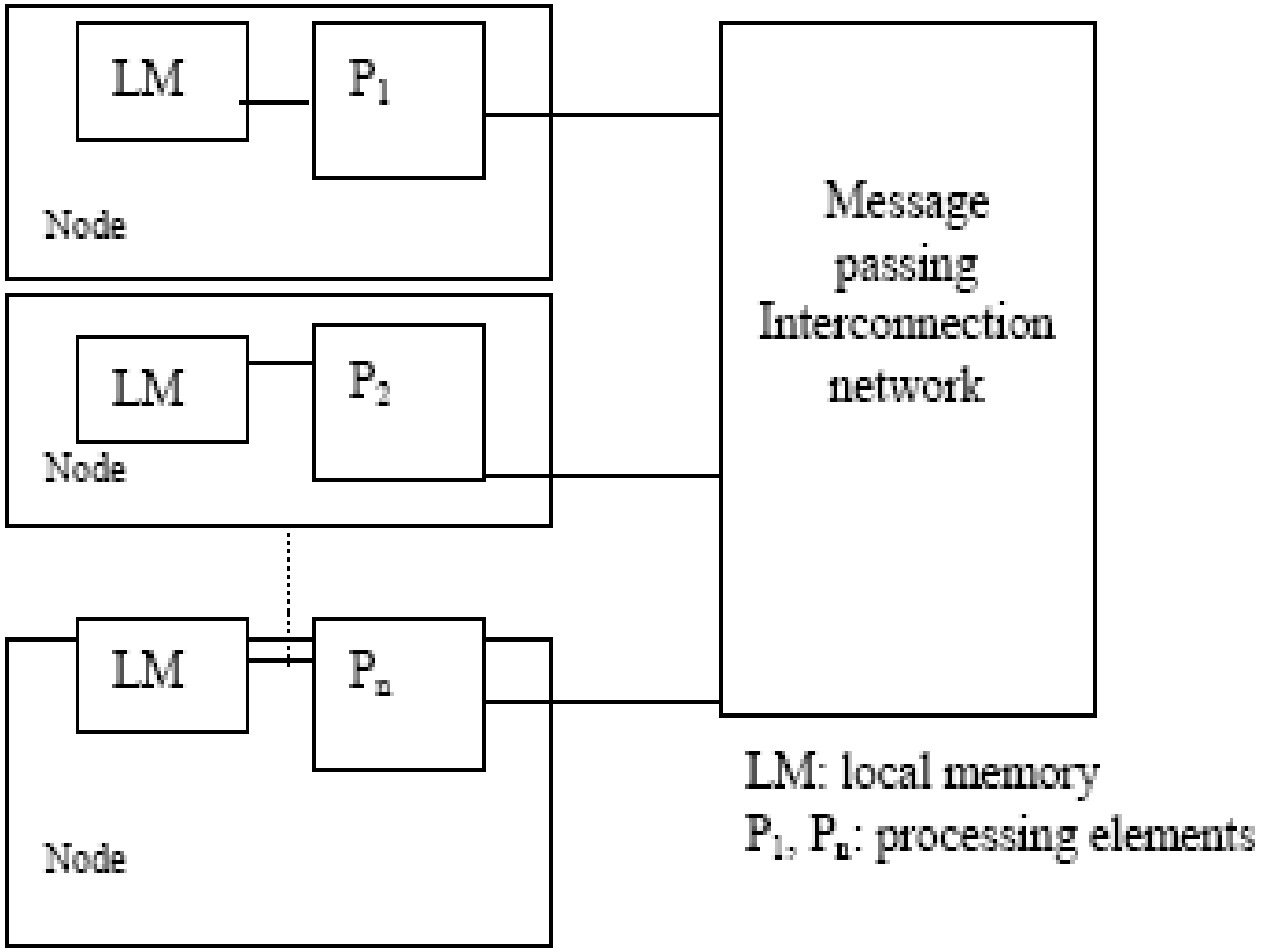


Figure 14: Loosely coupled system organisation

- each processor in loosely coupled systems is having a large local memory (LM), which is not shared by any other processor.
- such systems have multiple processors with their own local memory and a set of I/O devices.
- This set of processor, memory and I/O devices makes a computer system.
- these systems are also called **multi-computer systems.**

- These computer systems are connected together via message passing interconnection network through which processes communicate by passing messages to one another.

- Also called as **distributed multi computer system .**

CLASSIFICATION BASED ON GRAIN SIZE

- This classification is based on recognizing the parallelism in a program to be executed on a multiprocessor system.

- The idea is to identify the sub-tasks or instructions in a program that can be executed in parallel .

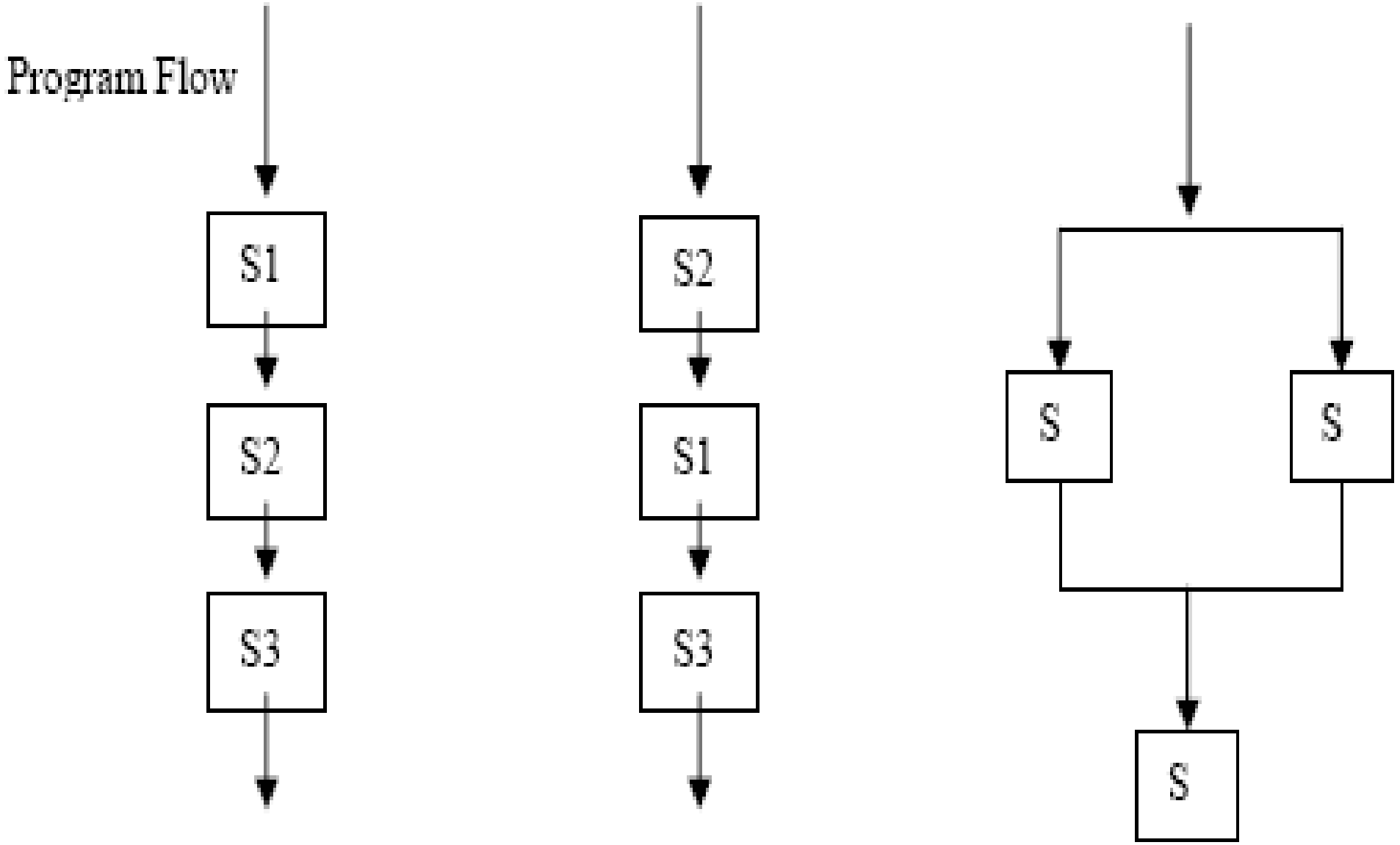


Figure 15: Parallel execution for S1 and S2

Factors affecting decision of parallelism

- Number and types of processors available, i.e. architectural features of host computer
  - Memory organization
  - Dependency of data, control and resources

Parallelism conditions

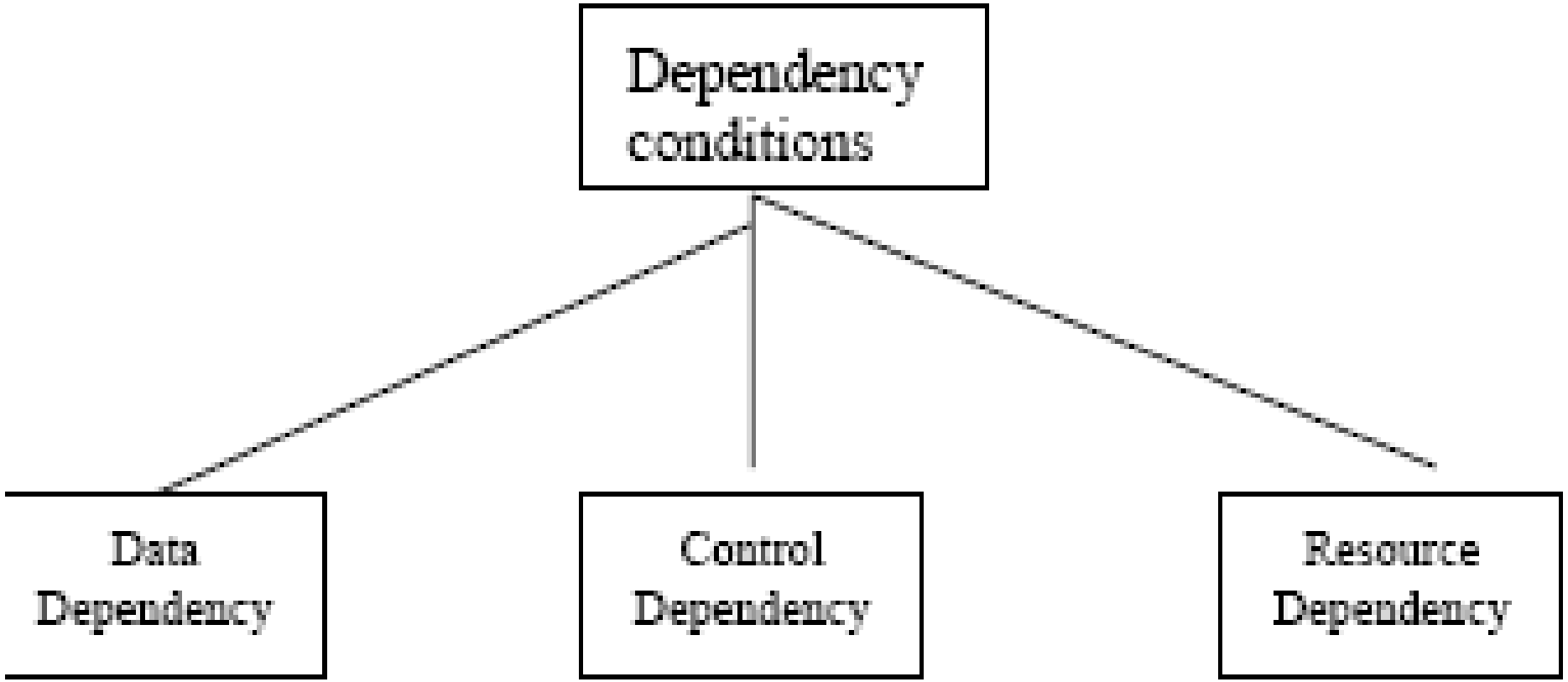


Figure 16: Dependency relations among the segments for parallelism

Data Dependency

- It refers to the situation in which two or more instructions share same data.
- The instructions in a program can be arranged based on the relationship of data dependency
- how two instructions or segments are data dependent on each other

Types of data dependencies

i) **Flow Dependence** : If instruction I2 follows I1 and output of I1 becomes input of I2, then I2 is said to be flow dependent on I1.

ii) **Antidependence** : When instruction I2 follows I1 such that output of I2 overlaps with the input of I1 on the same data.

iii) **Output dependence** : When output of the two instructions I1 and I2 overlap on the same data, the instructions are said to be output dependent.

iv) **I/O dependence** : When read and write operations by two instructions are invoked on the same file, it is a situation of I/O dependence.

Control Dependence

- Instructions or segments in a program may contain control structures.
- dependency among the statements can be in control structures also. But the order of execution in control structures is not known before the run time.
- control structures dependency among the instructions must be analyzed carefully

Resource Dependence

- The parallelism between the instructions may also be affected due to the shared resources.

- If two instructions are using the same shared resource then it is a resource dependency condition

Bernstein Conditions for Detection of Parallelism

- For execution of instructions or block of instructions in parallel, The instructions should be independent of each other.

- These instructions can be data dependent / control dependent / resource dependent on each other .

Bernstein conditions are based on the following two sets of variables

i) The Read set or input set RI that consists of memory locations read by the statement of instruction I1.

ii) The Write set or output set WI that consists of memory locations written into by instruction I1.

Parallelism based on Grain size

- **Grain size:** Grain size or Granularity is a measure which determines how much computation is involved in a process.

- Grain size is determined by counting the number of instructions in a program segment.

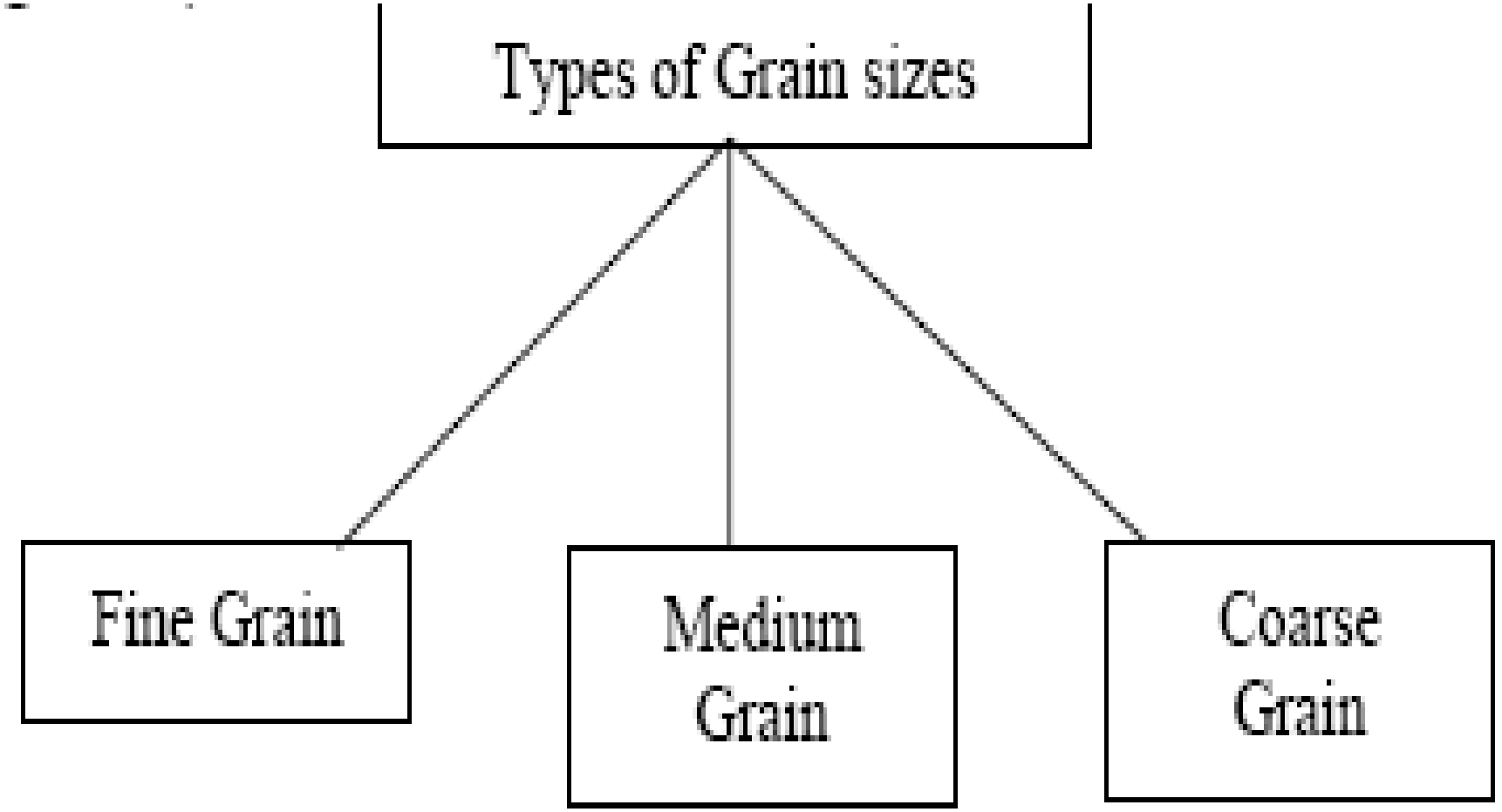


Figure 17: Types of Grain sizes

1) **Fine Grain**: This type contains approximately less than 20 instructions.

2) **Medium Grain**: This type contains approximately less than 500 instructions.

3) **Coarse Grain**: This type contains approximately greater than or equal to one thousand instructions.

LEVELS OF PARALLEL PROCESSING

- Instruction Level
- Loop level
- Procedure level
- Program level

Instruction level

- This is the lowest level and the degree of parallelism is highest at this level.
- The fine grain size is used at instruction level
- The fine grain size may vary according to the type of the program. For example, for scientific applications, the instruction level grain size may be higher.
- As the higher degree of parallelism can be achieved at this level, the overhead for a programmer will be more.

Loop Level

- This is another level of parallelism where iterative loop instructions can be parallelized.
- Fine grain size is used at this level also.
- Simple loops in a program are easy to parallelize whereas the recursive loops are difficult.
- This type of parallelism can be achieved through the compilers .

Procedure or Sub Program Level

- This level consists of procedures, subroutines or subprograms.
- Medium grain size is used at this level containing some thousands of instructions in a procedure.
- Multiprogramming is implemented at this level.

Program Level

- It is the **last level consisting** of independent programs for parallelism.
- Coarse grain size is used at this level containing tens of thousands of instructions.
- Time sharing is achieved at this level of parallelism

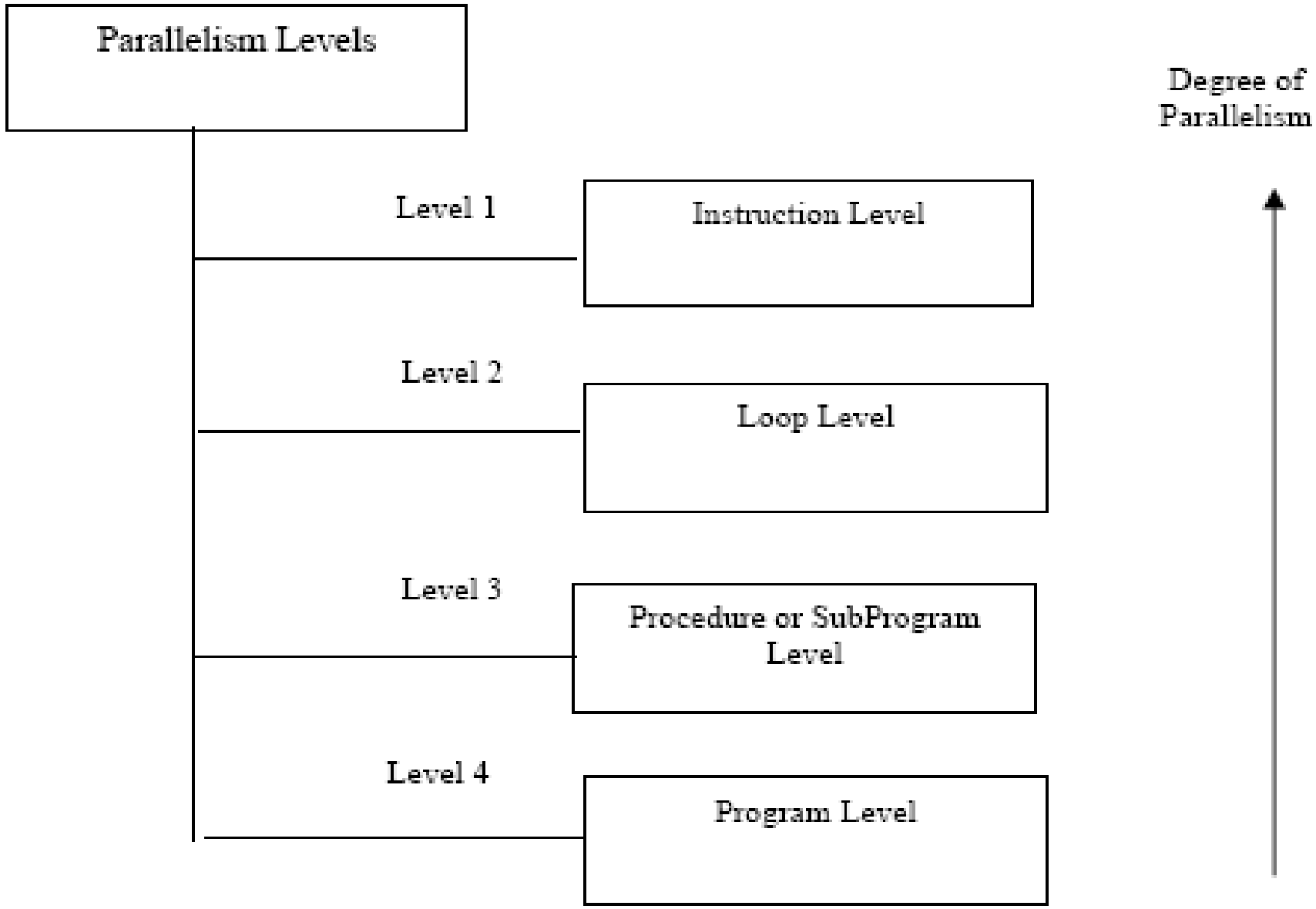


Figure 18: Parallelism Levels

Table 1: Relation between grain sizes and parallelism

Grain Size	**Parallelism Level**
Fine Grain	Instruction or Loop Level
Medium Grain	Procedure or SubProgram Level
Coarse Grain	Program Level