Computer Arithmetic

- However, when A < B, the sign of the result is the complement of the original sign of A. It is then necessary to complement A, to obtain the correct sign.

- The final result is found in register A and its sign in As

- The value in AVF provides an overflow indication. The final value of E is immaterial.

Computer Arithmetic

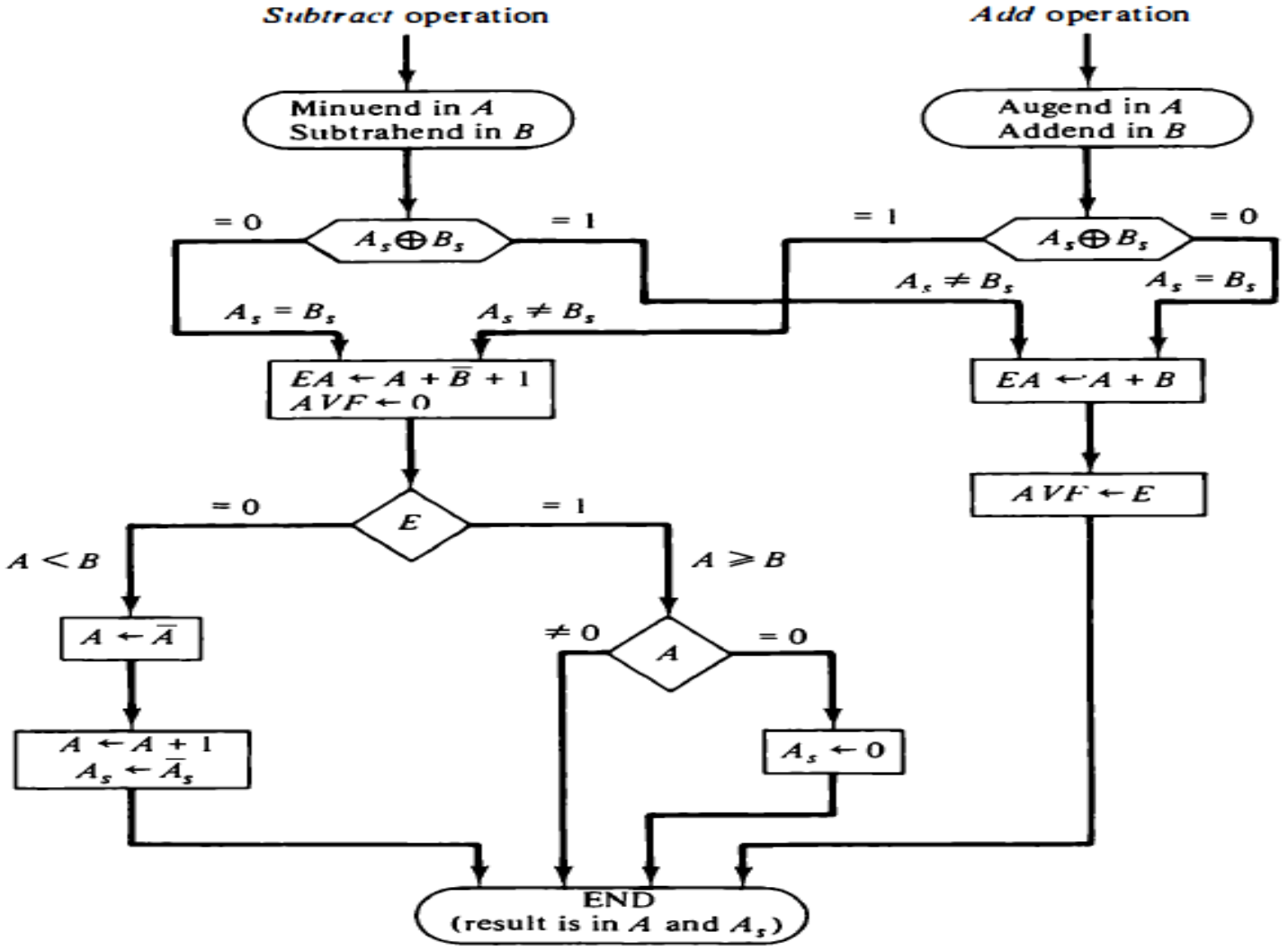


Figure 10-2 Flowchart for add and subtract operations.

Computer Arithmetic

- **Addition and Subtraction with Signed-2’s Complement Data**

- The signed-2's complement representation of numbers together with arithmetic algorithms for addition and subtraction are introduced.

- The leftmost bit of a binary number represents the sign bit: 0 for positive and 1 for negative.

Computer Arithmetic

- If the sign bit is 1, the entire number is represented in 2' s complement form.
- Thus + 33 is represented as 00100001 and - 33 as 1101 1 1 1 1 .
- Note that 11011111 is the 2's complement of 00100001, and vice versa.

Computer Arithmetic

- The addition of two numbers in signed-2's complement form consists of adding the numbers with the sign bits treated the same as the other bits of then number. A carry-out of the sign-bit position is discarded.

- The subtraction consists of first taking the 2' s complement of the subtrahend and then adding it to the minuend.

Computer Arithmetic

- When two numbers of n digits each are added and the sum occupies n + 1 digits, we say that an overflow occurred.

- An overflow can be detected by inspecting the last two carries out of the addition.

- When the two carries are applied to an exclusive-OR gate, the overflow is detected when the output of the gate is equal to 1.

Computer Arithmetic

- The register configuration for the hardware implementation i s shown in Fig. 10-3.
- We name the A register AC (accumulator) and the B register BR.
- The leftmost bit in AC and BR represent the sign bits of the numbers.

Computer Arithmetic

- The two sign bits are added or subtracted together with the other bits in the complementer and parallel adder.

- The overflow flip-flop V is set to 1 if there is an overflow.

- The algorithm for adding and subtracting two binary numbers in signed- 2' s complement representation is shown in the flowchart of Fig. 10-4.

Computer Arithmetic

- The sum is obtained by adding the contents of AC and BR (including their sign bits).

- The overflow bit V is set to 1 if the exclusive-OR of the last two carries is 1, and it is cleared to 0 otherwise.

Computer Arithmetic

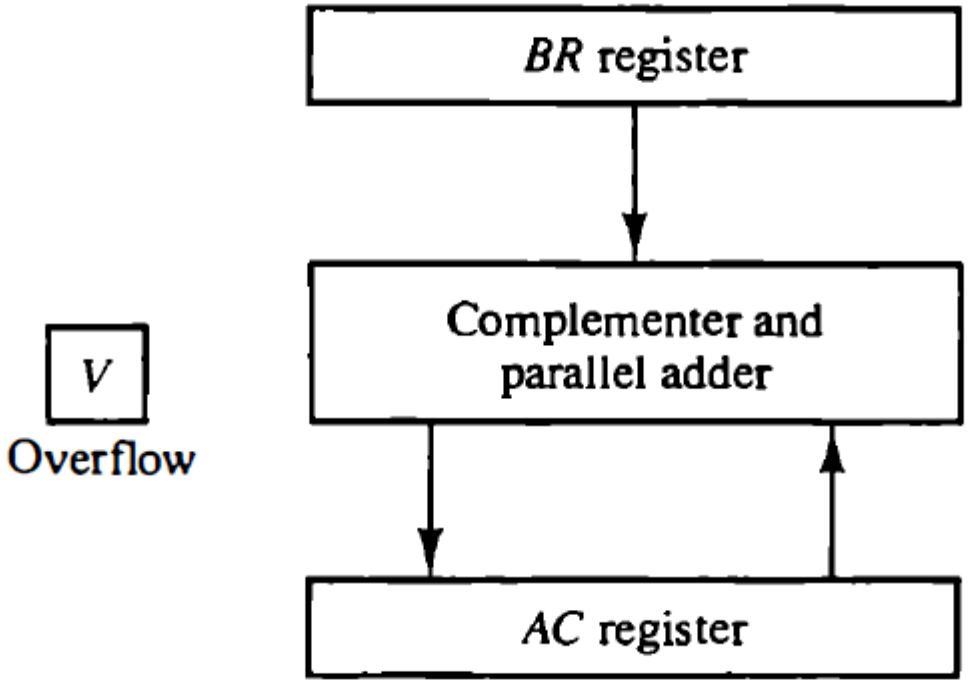
- The subtraction operation is accomplished by adding the content of AC to the 2's complement of BR .

- Taking the 2's complement of BR has the effect of changing a positive number to negative, and vice versa.

Computer Arithmetic

- An overflow must be checked during thls operation because the two numbers added could have the same sign.

Figure 10-3 Hardware for signed-2's complement addition and subtraction.



Computer Arithmetic

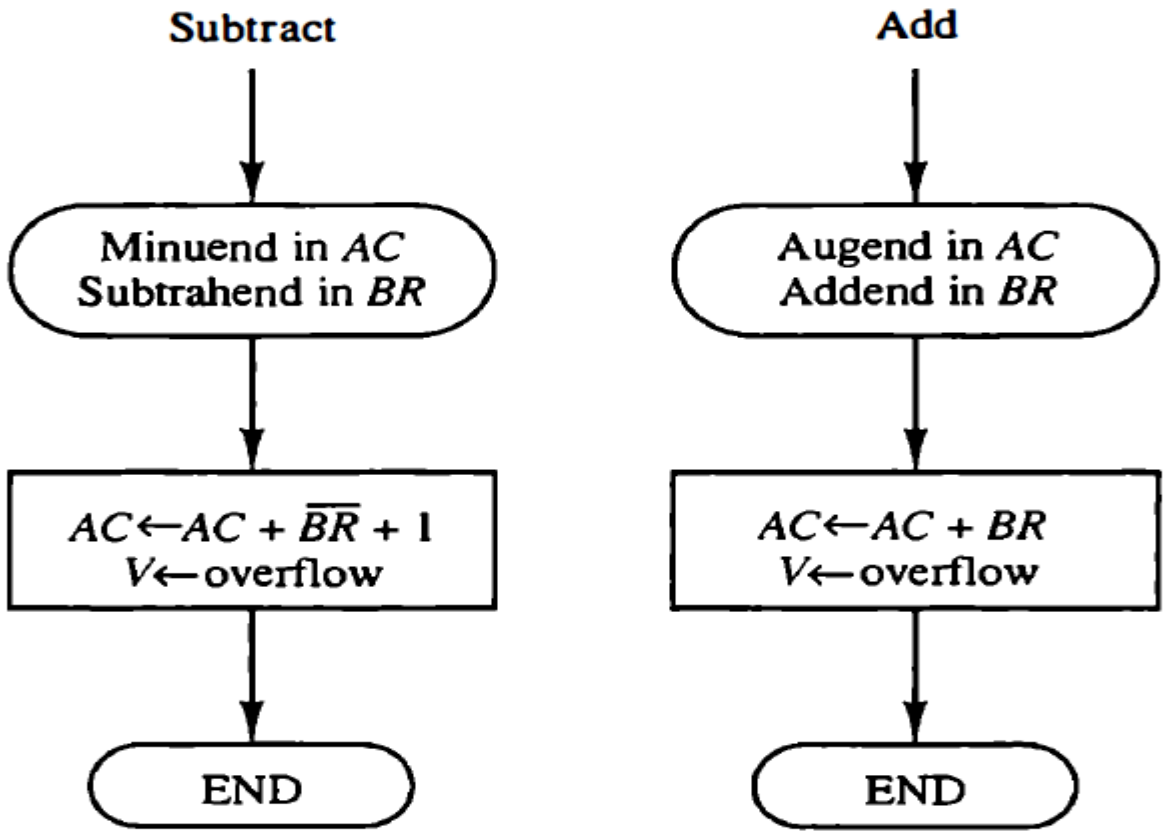
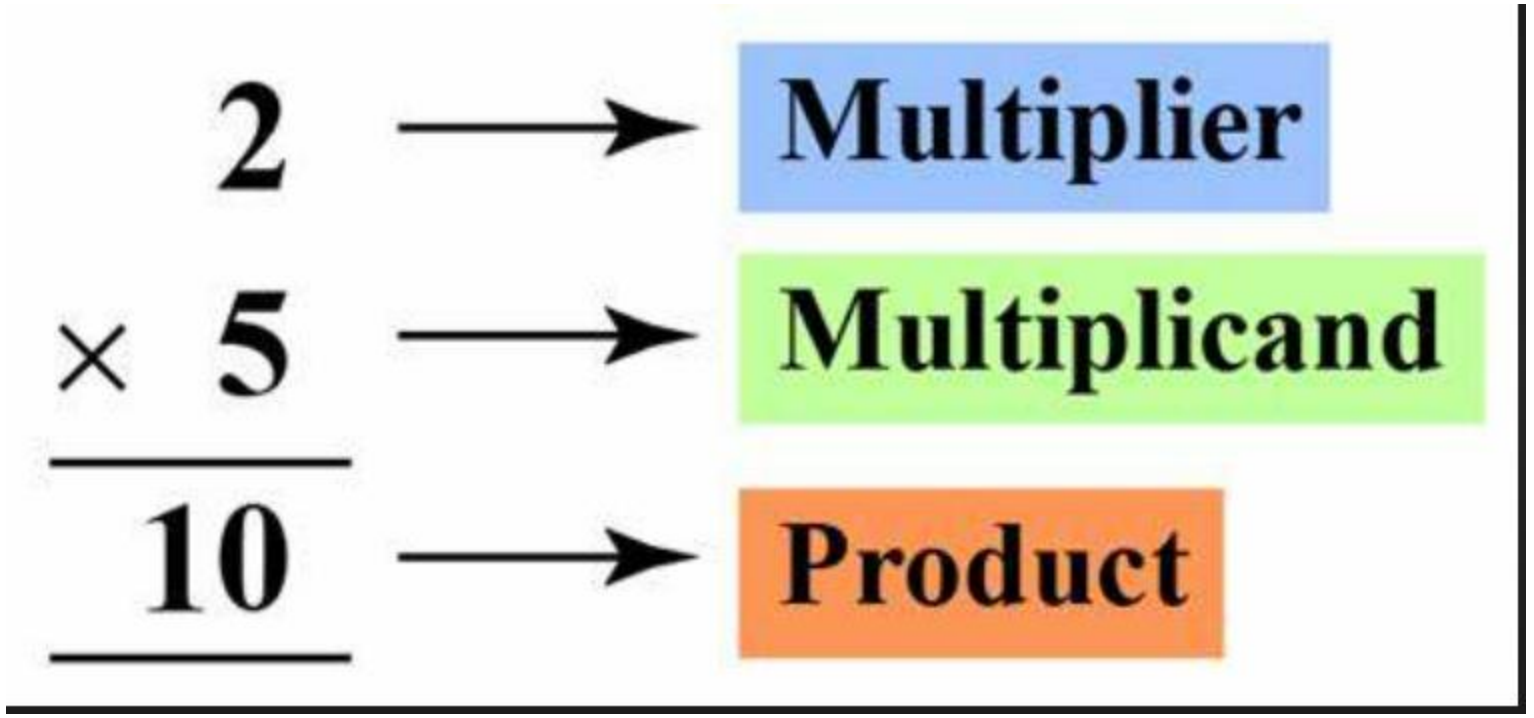


Figure 10-4 Algorithm for adding and subtracting numbers in signed-2's complement representation.

Computer Arithmetic

- 

Computer Arithmetic

Figure 10-6 Flowchart for multiply operation.

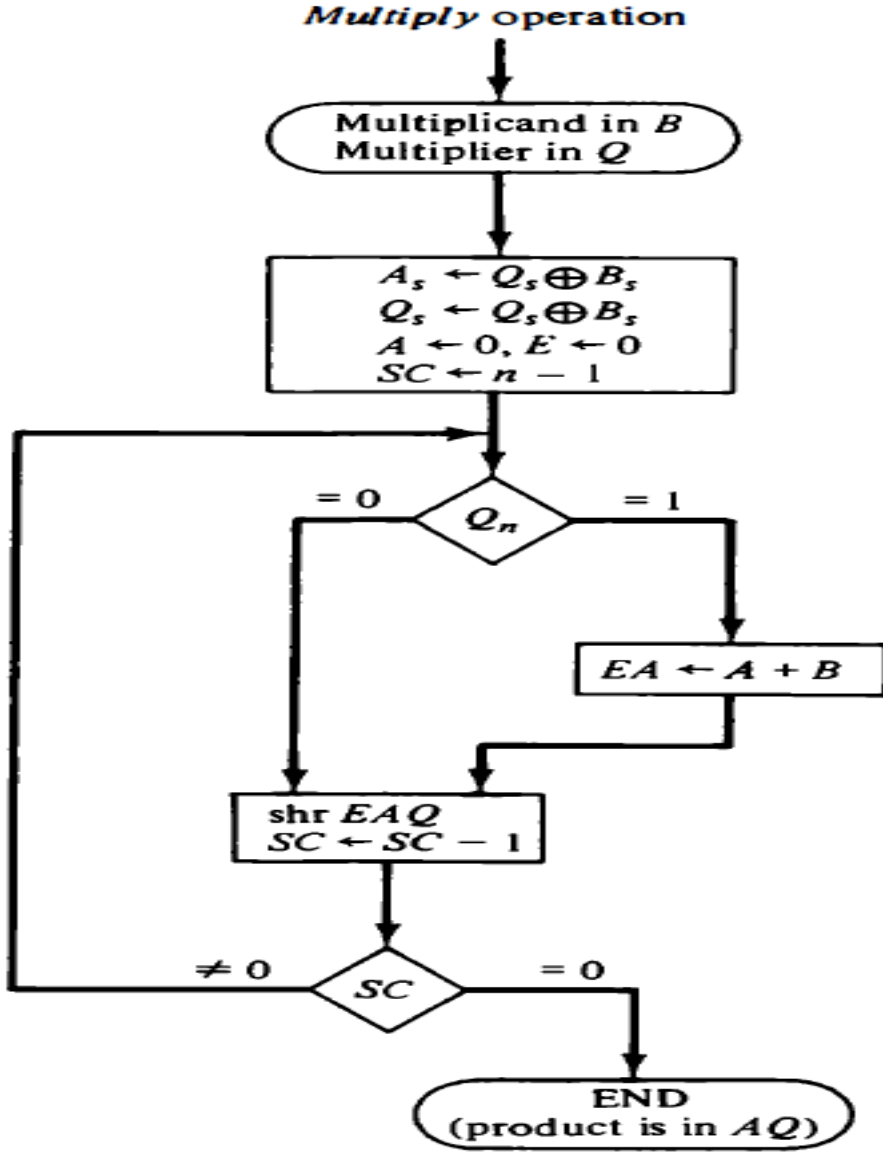


TABLE 10-2 Numerical Example for Binary Multiplier

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Q_n = 1; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
Q_n = 1; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
Q_n = 0; shift right EAQ	0	01000	10110	010
Q_n = 0; shift right EAQ	0	00100	01011	001
Q_n = 1; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

Computer Arithmetic-Booth Algorithm

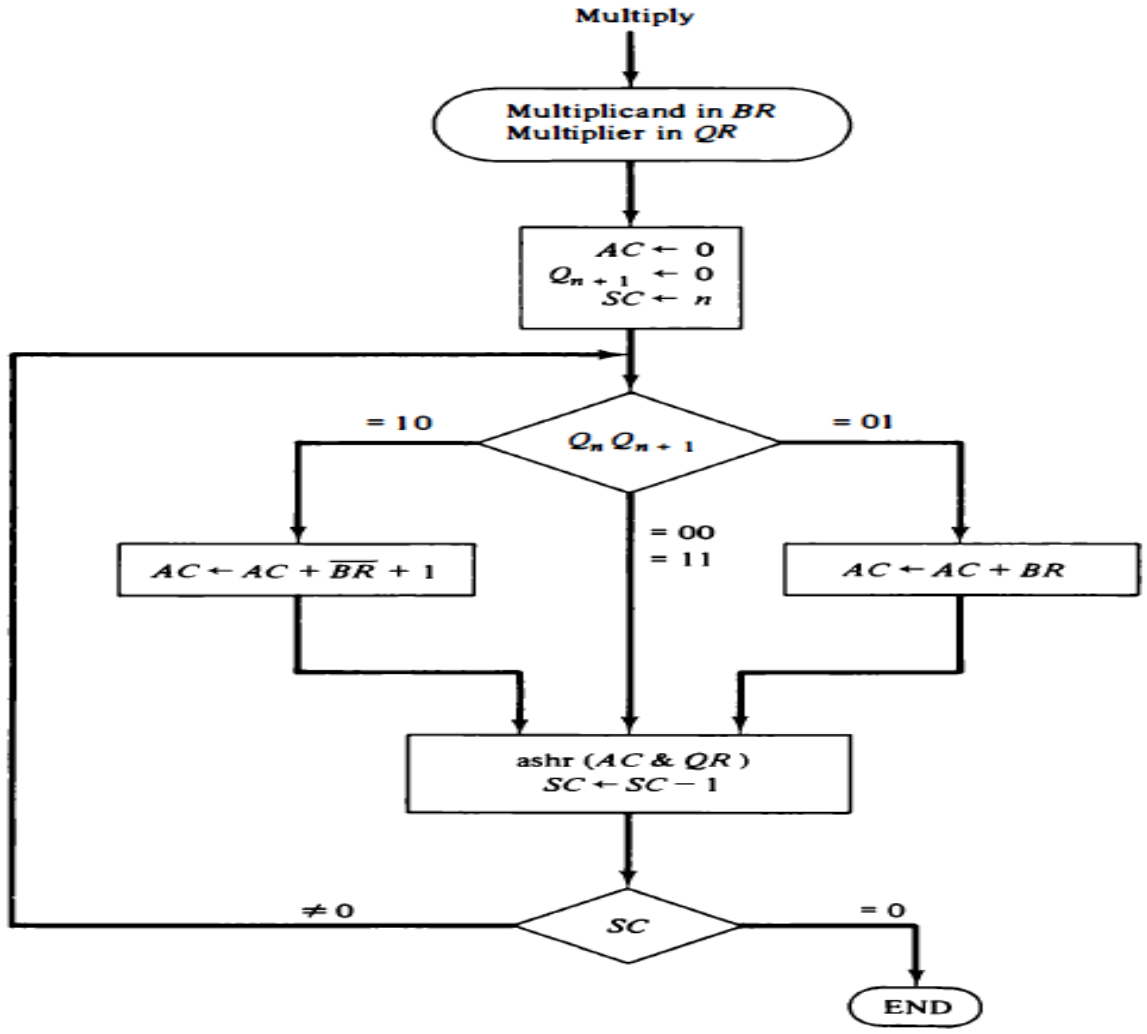


Figure 10-8 Booth algorithm for multiplication of signed-2's complement numbers.

Computer Arithmetic-Booth Algorithm

TABLE 10-3 Example of Multiplication with Booth Algorithm

$Q_n Q_{n+1}$	$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000

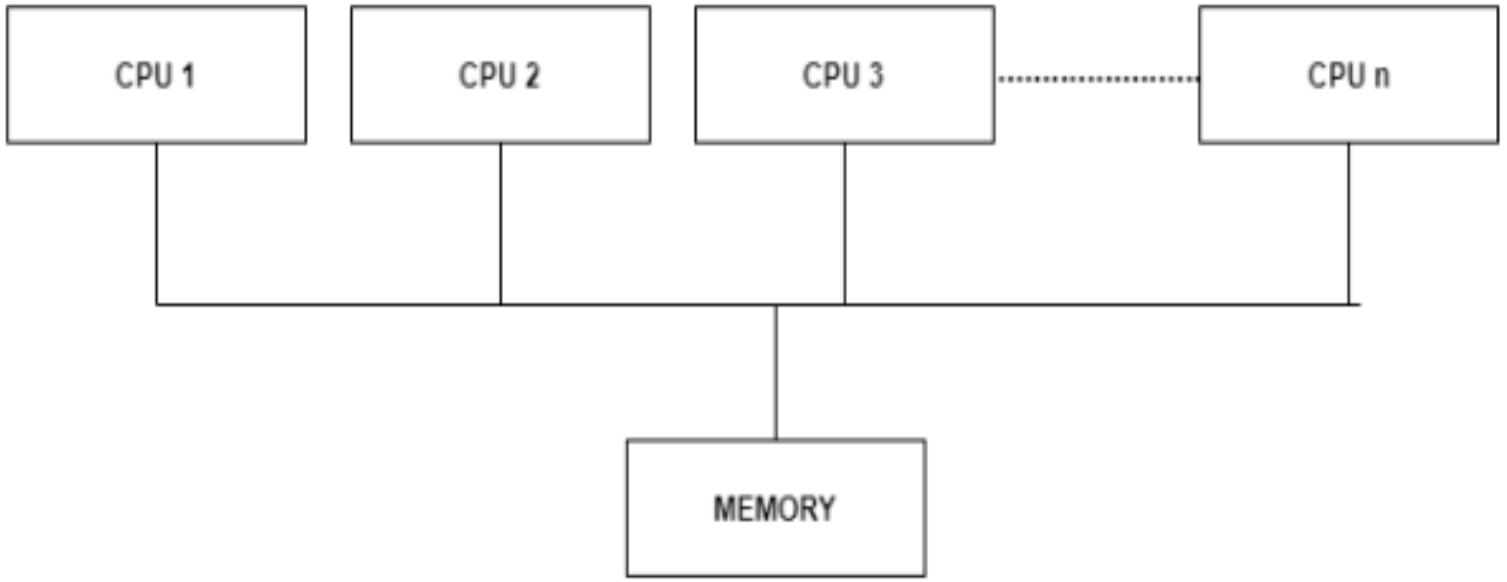
Multiprocessors

- 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 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).

Multiprocessors



Multiprocessing Architecture

Multiprocessors

❑ **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.

Multiprocessors

- **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.

Multiprocessors

❑Advantage of Multiprocessor Systems

❑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.

Multiprocessors

- 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.

Multiprocessors

- **Disadvantages of Multiprocessor Systems**

- 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.

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

Coupling of Processors

❑Loosely Coupled System

- ❑ - Tasks or processors do not communicate in a synchronized fashion
- ❑ - 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 between the processors and memory in a shared memory system among the processing elements.

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

Interconnection Structure

- **Time Shared Common Bus**

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

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

Interconnection Structure

- 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 bus structure is depicted in Fig.

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

Interconnection Structure

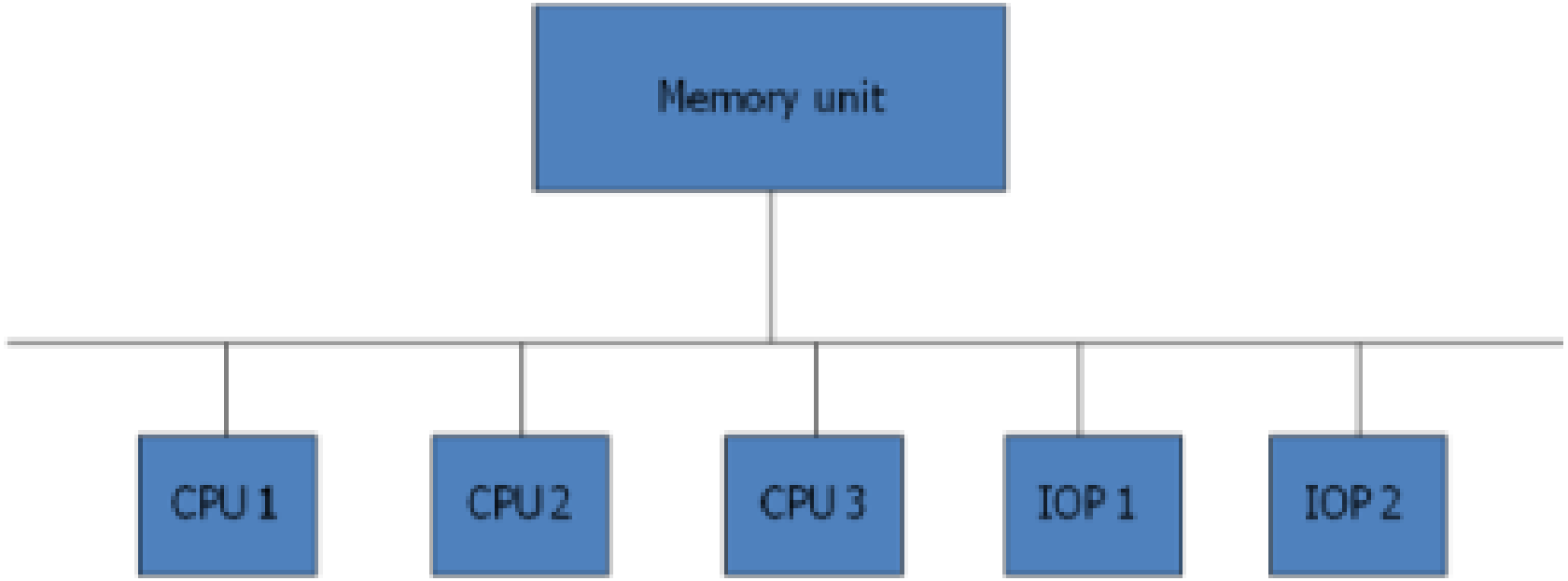


Fig: Time shared common bus organization

Interconnection Structure

- **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.

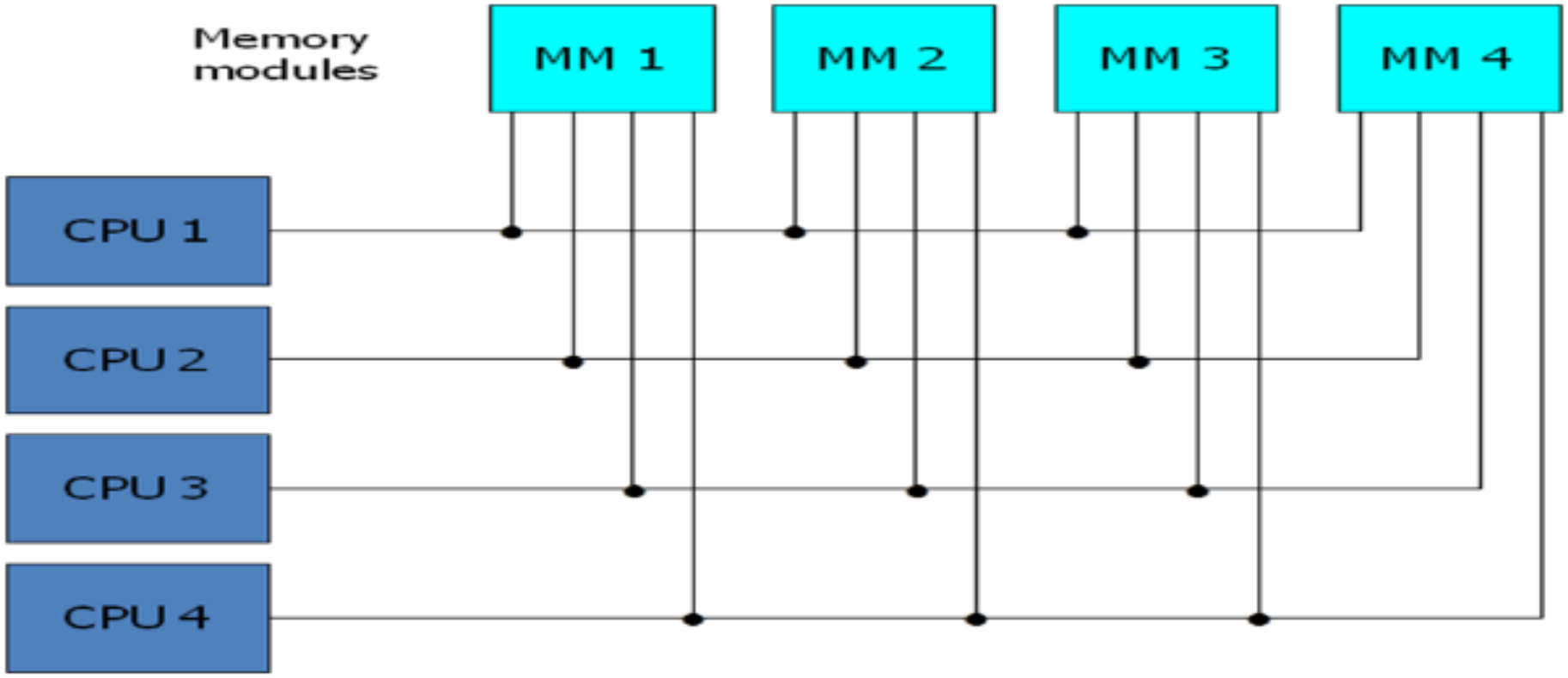
Interconnection Structure

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

- Adv.:
- The high transfer rate can be achieved because of the multiple paths.

- Disadv.:
- It requires expensive memory control logic and a large number of cables and connections

Interconnection Structure



Interconnection Structure

- **Cross Bar Switch**

- 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.:**
- Supports simultaneous transfers from all memory modules.

- **Disadv.:**
- The hardware required to implement the switch can become quite large and complex.

Interconnection Structure

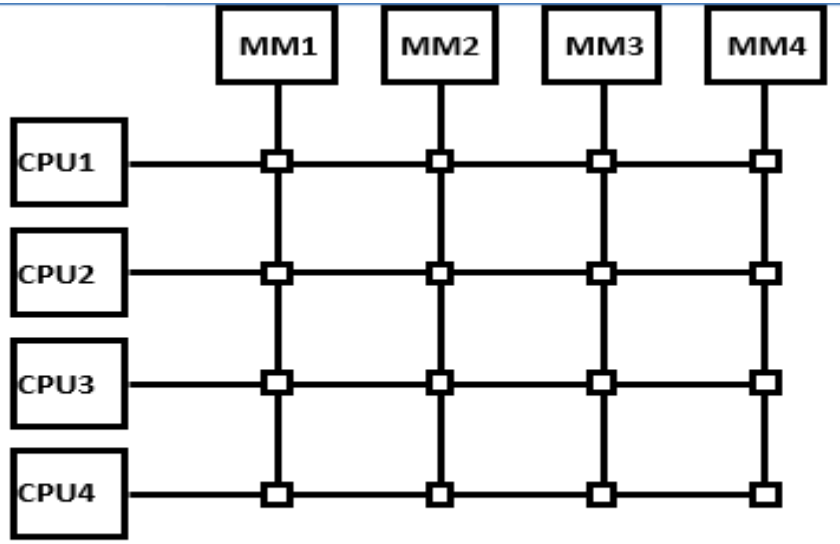


fig. shows the functional design of a crossbar switch connected to one memory module.

Multiprocessor architectures

- 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.

Multiprocessor architectures

- 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.

Multiprocessor architectures

- (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, and other read-only data, stacks, and local variables in the private memories.

- The shared memory is then only used for writable shared variables.

Multiprocessor architectures

-

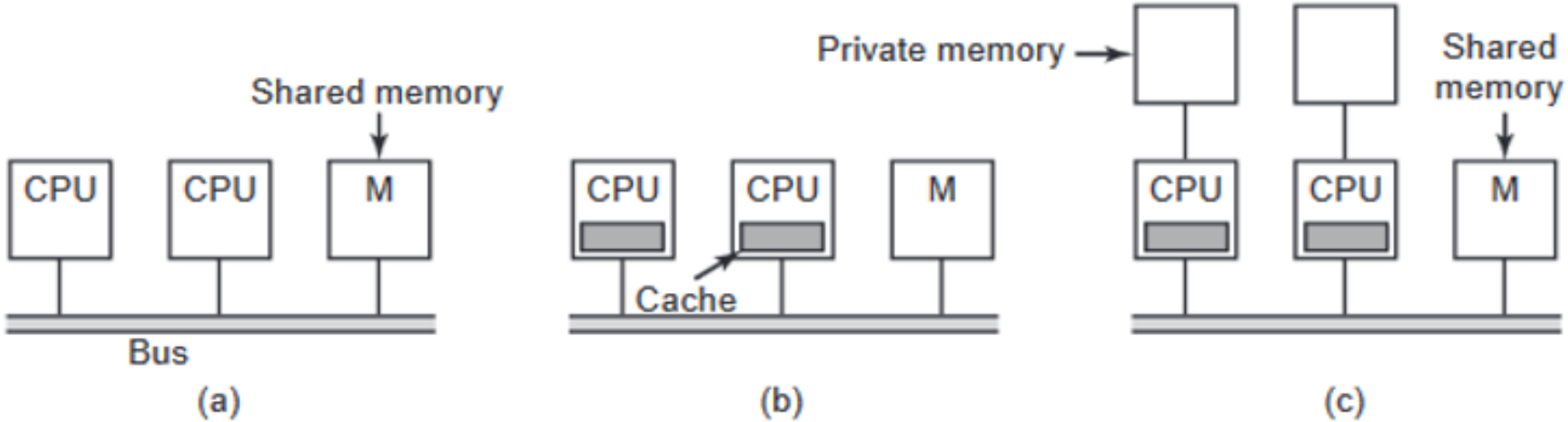


Figure 8-1. Three bus-based multiprocessors. (a) Without caching. (b) With caching. (c) With caching and private memories.

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.

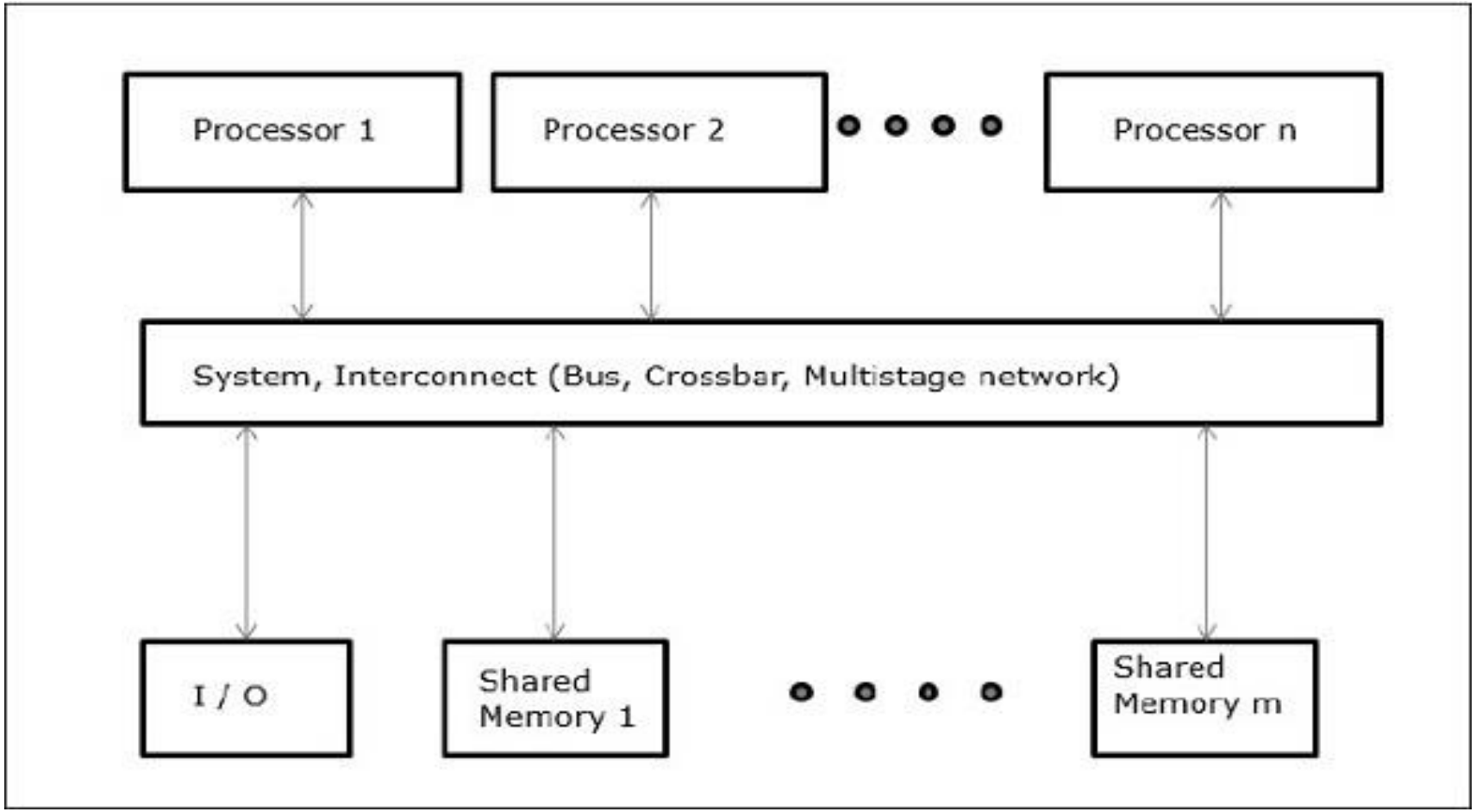
- Each processor may have a private cache memory. Same rule is followed for peripheral devices.

Shared-Memory Multicomputers

- 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.
-

Shared-Memory Multi-computers



Shared-Memory Multi-computers

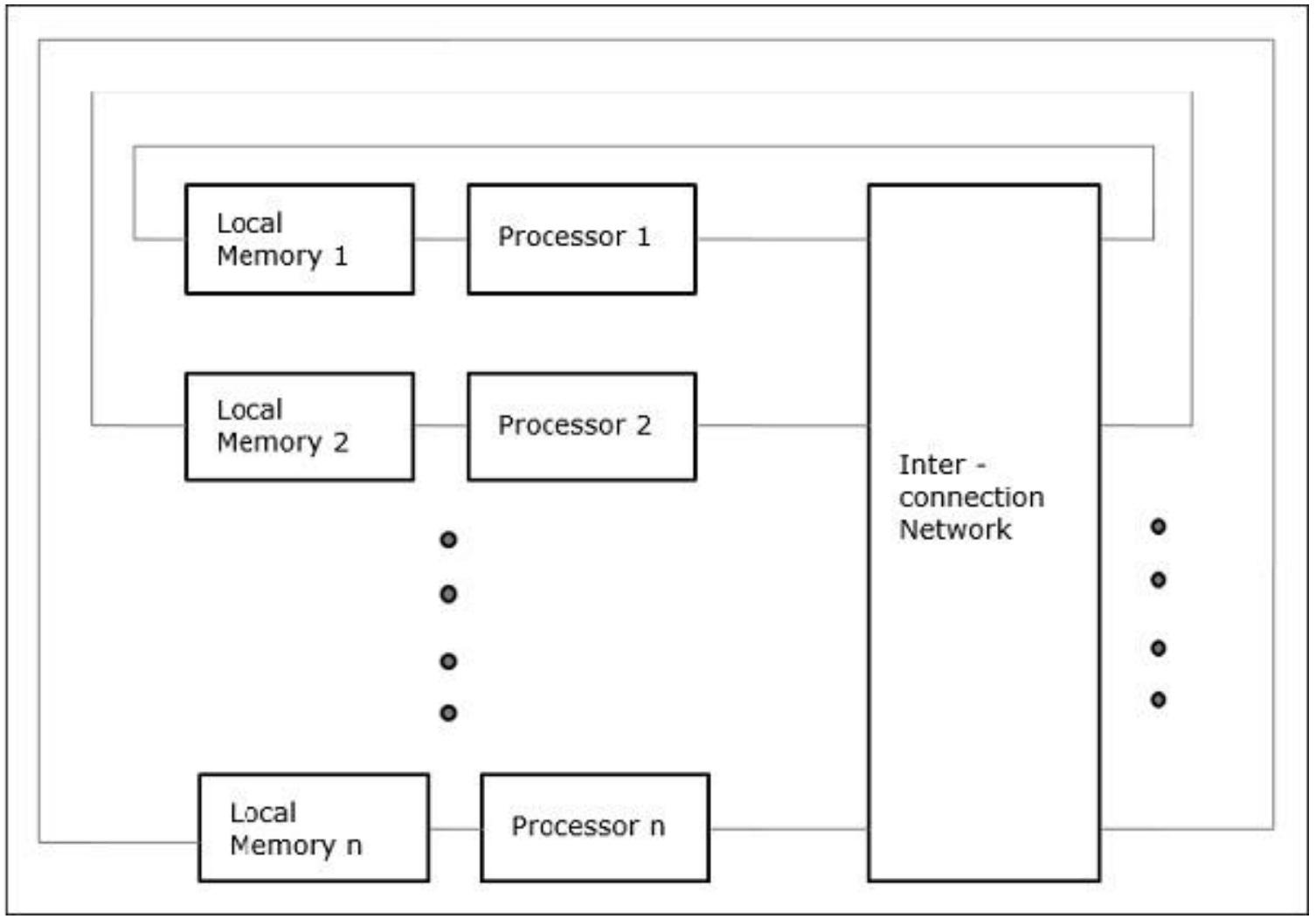
- **Non-uniform Memory Access (NUMA)**

- 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.

Shared-Memory Multi-computers



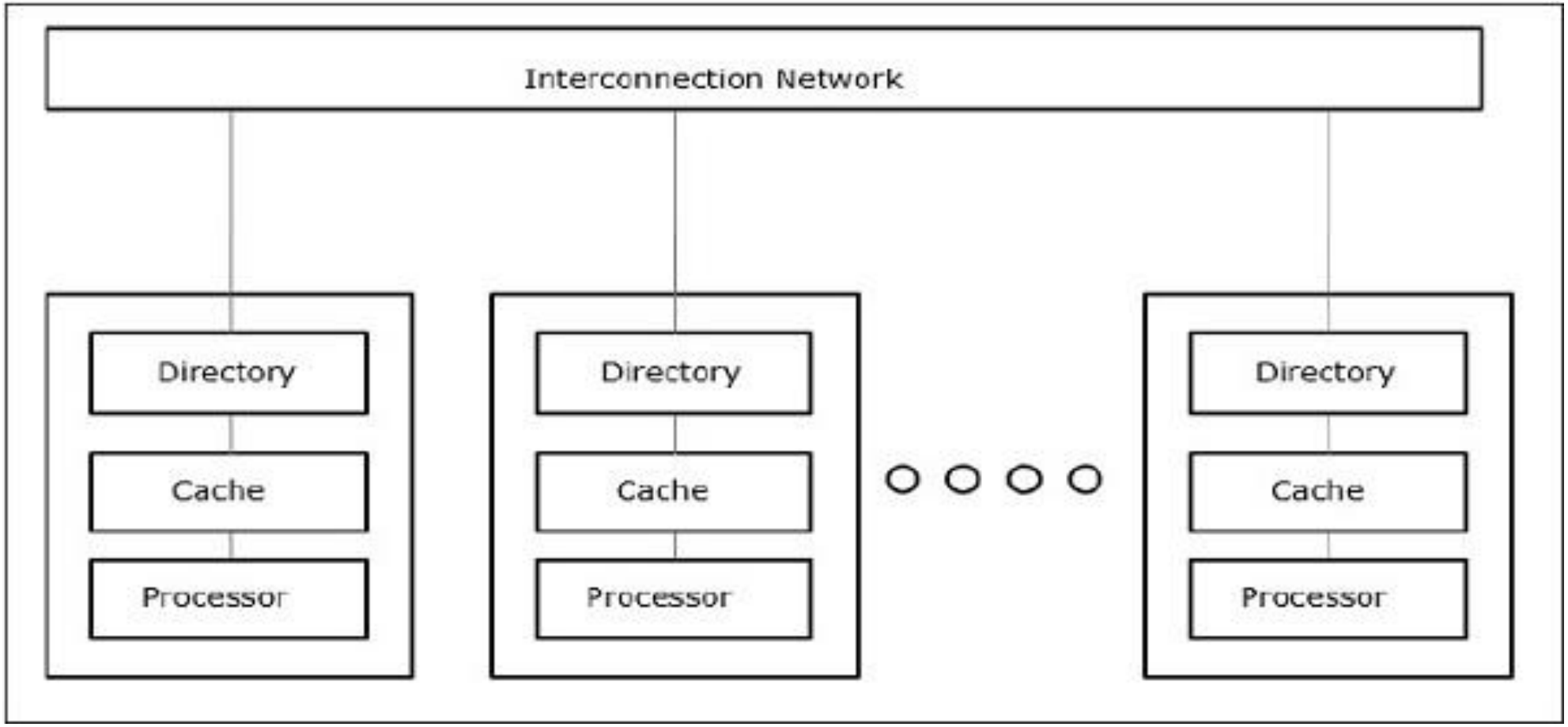
Shared-Memory Multi-computers

- **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.

Shared-Memory Multi-computers



Parallel processing

❑ **Serial processing deals with following:**

❑In this, a problem statement is broken into discrete instructions.

❑Then the instructions are executed one by one.

❑Only one instruction is executed at any moment of time.

Parallel processing

- Why Parallel Architecture?

- Parallel computer architecture adds a new dimension in the development of computer system by using more and more number of processors.

- In principle, performance achieved by utilizing large number of processors is higher than the performance of a single processor at a given point of time.

Parallel processing

- **Advantages of Parallel Computing over Serial Computing are as follows:**

- It saves time and money as many resources working together will reduce the time and cut potential costs.

- It can be impractical to solve larger problems on Serial Computing.

- Serial Computing ‘wastes’ the potential computing power, thus Parallel Computing makes better work of hardware.

Parallel processing

- Execution of Concurrent Events in the computing.
- Process to achieve faster Computational Speed.

Parallel processing

❑ 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.

- 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 processing

- **Types of Parallelism:**

- **Bit-level parallelism:**

- Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers.

- It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation.

- A 16-bit processor can perform the operation with just one instruction.

Parallel processing

- **Instruction-level parallelism:**

- A processor can only address less than one instruction for each clock cycle phase.

- These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program.

- This is called instruction-level parallelism.

Parallel processing

- **Task Parallelism:**

- Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution.

- The processors perform execution of sub tasks concurrently.

Parallel Computers

Architectural Classification

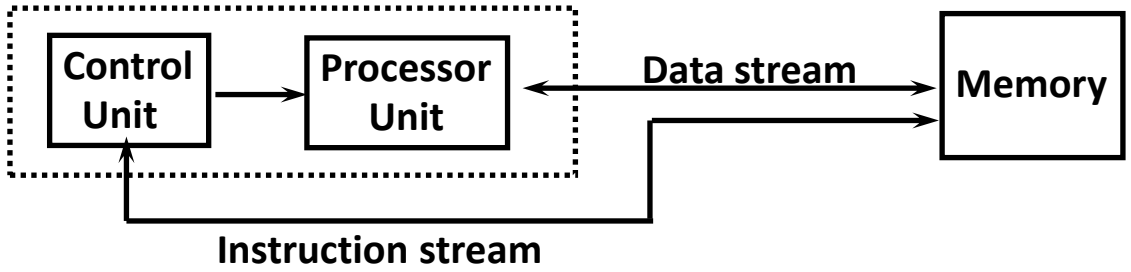
- **Flynn's classification**
  - **Based on the multiplicity of *Instruction Streams* and *Data Streams***
  - **Instruction Stream**
    - **Sequence of Instructions read from memory**
  - **Data Stream**
    - **Operations performed on the data in the processor**

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

Parallel processing

- Single instruction stream, single data stream (SISD)
  Single instruction stream, multiple data stream (SIMD)
  Multiple instruction stream, single data stream (MISD)
  Multiple instruction stream, multiple data stream (MIMD)

SISD



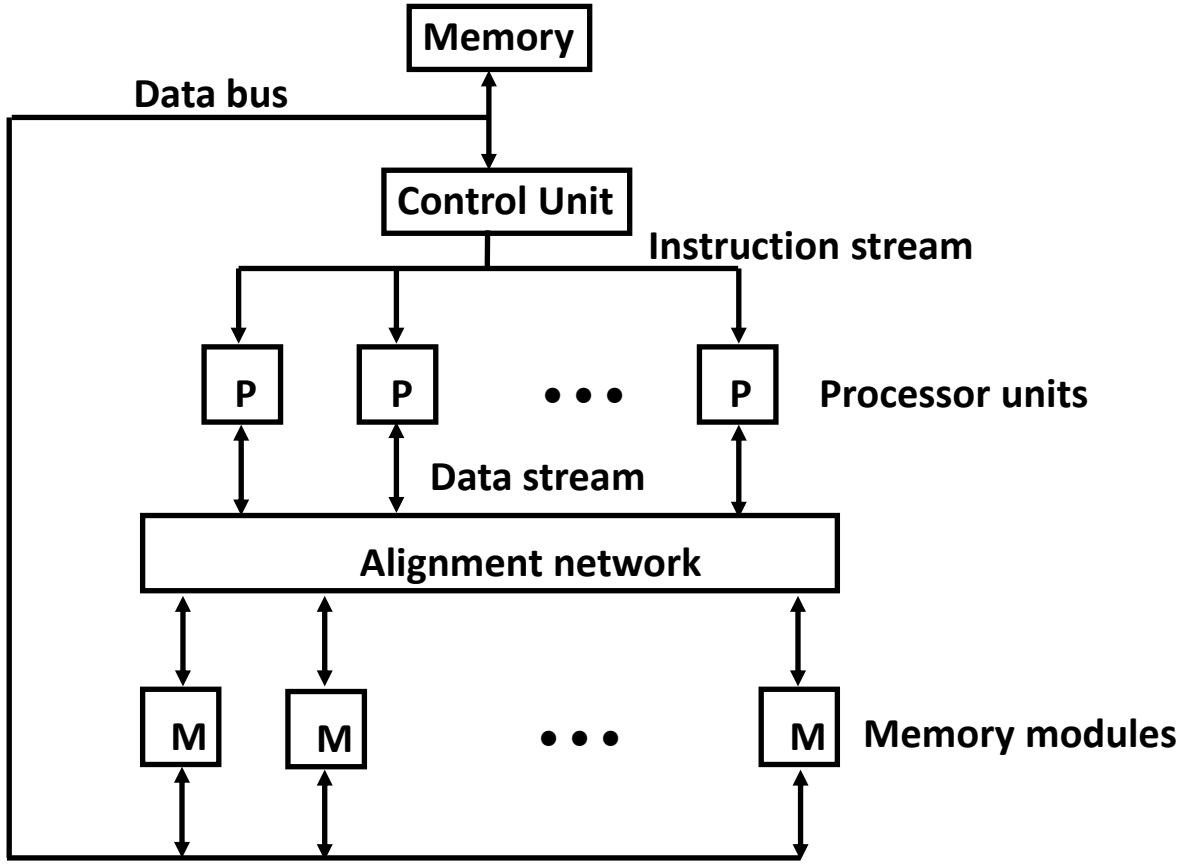
Characteristics

- 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

SIMD



Characteristics

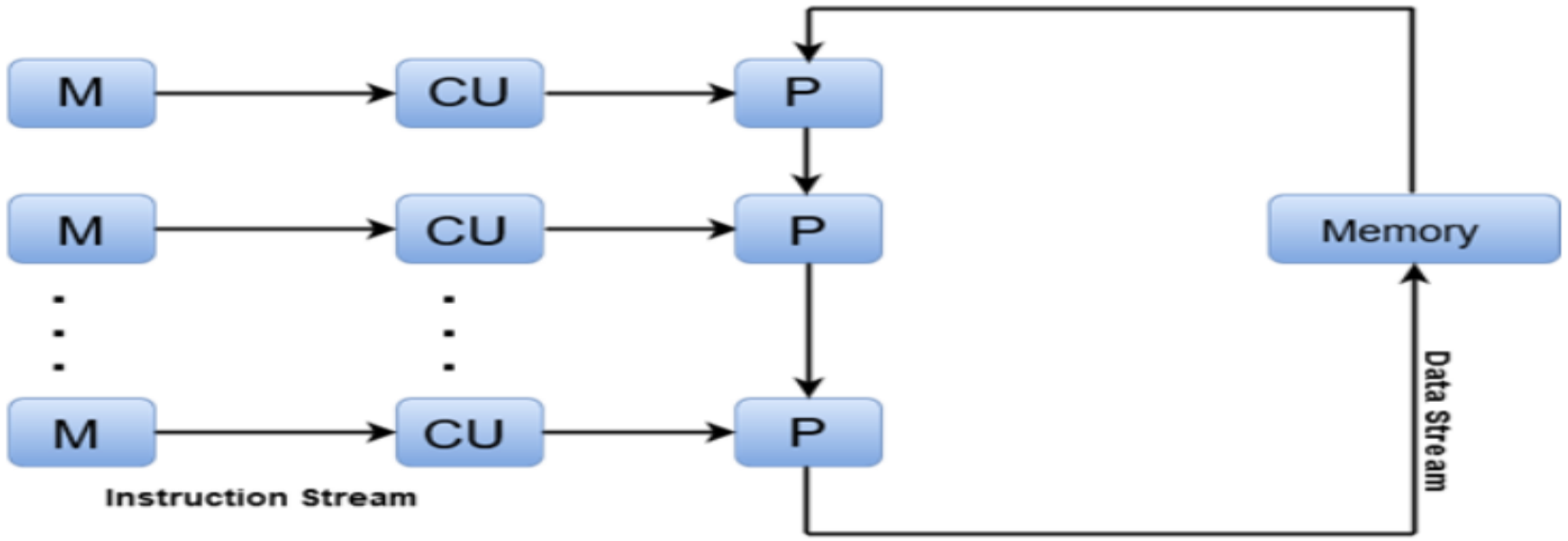
- Only one copy of the program exists
- A single controller executes one instruction at a time

MISD

MISD structure is only of theoretical interest since no practical system has been constructed using this organization.

In MISD, multiple processing units operate on one single-data stream. Each processing unit operates on the data independently via separate instruction stream.

MISD:



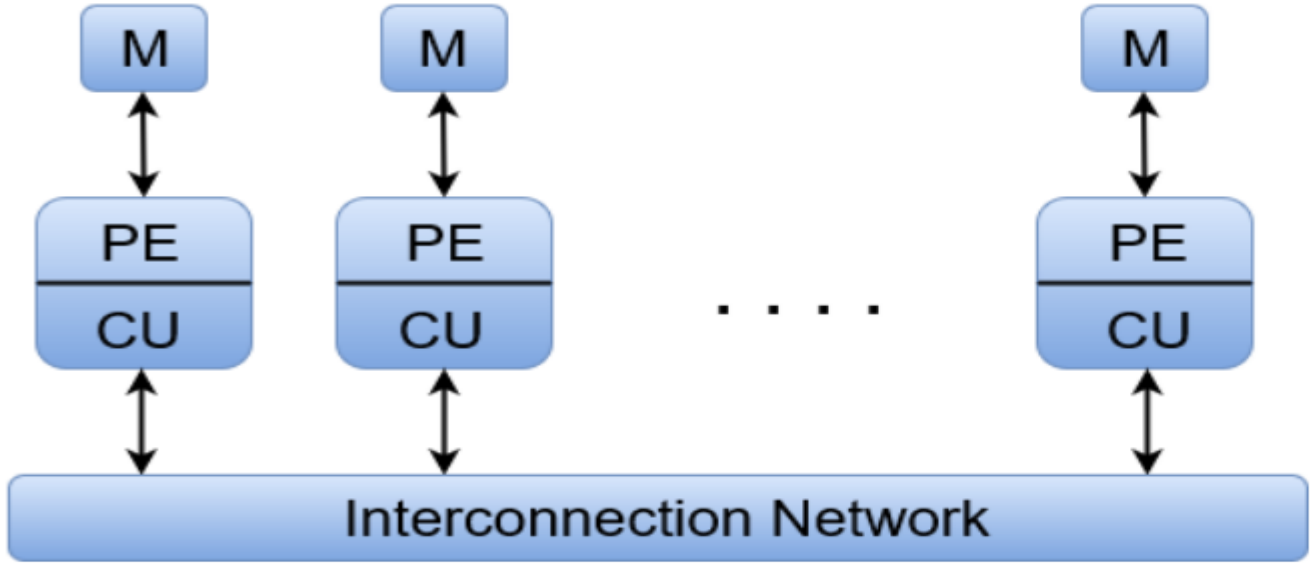
Where, M = Memory Modules, CU = Control Unit, P = Processor Units

MIMD

In this organization, all processors in a parallel computer can execute different instructions and operate on various data at the same time.

In MIMD, each processor has a separate program and an instruction stream is generated from each program.

MIMD:



Where, M = Memory Module, PE = Processing Element, and CU = Control Unit

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

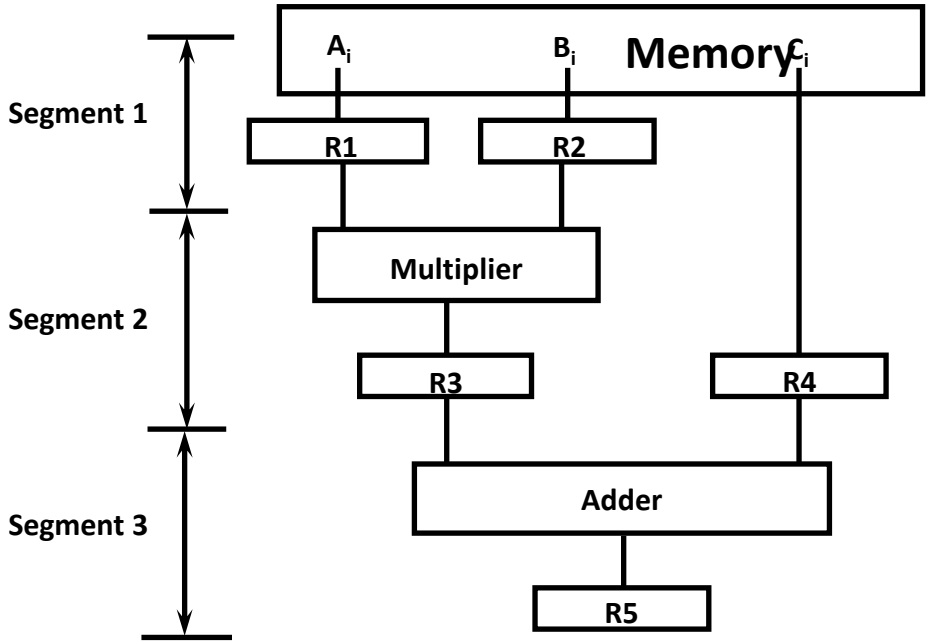
- The result obtained from computation is in each segment is transferred to next segment in the pipeline

- 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