

William Stallings
Computer Organization
and Architecture
8th Edition
Chapter 17
Parallel Processing

Multiprocessor Configurations

1. Parallel processing systems and concepts,
2. Flynn's classification,
3. Introduction to pipeline processing and pipeline hazards,
4. design issues of pipeline architecture,
5. Instruction pipelining: 6 stage

Parallel Processing Concepts

- To fulfil increasing demands for **higher performance**
- Need to process data concurrently to achieve **better throughput**
- **Parallelism**
 - Multiple functional units-several ALU's
 - Multiple processors-several processors operating concurrently

Parallel Processing Concepts

- 2 approaches for multiprocessor systems
 1. Tightly coupled systems(shared memory)
 2. Loosely coupled systems(Distributed memory)

Advantages of multiprocessor systems:

1. Increased reliability due to redundancy in processors.
2. Increased throughput because of execution of:
 - multiple jobs in parallel
 - portions of same job in parallel

Parallel Processing Concepts

1. Tightly coupled systems(shared memory):

coupled or distributed memory multi-processors. Such processors themselves through interconnection structure and using message passing.

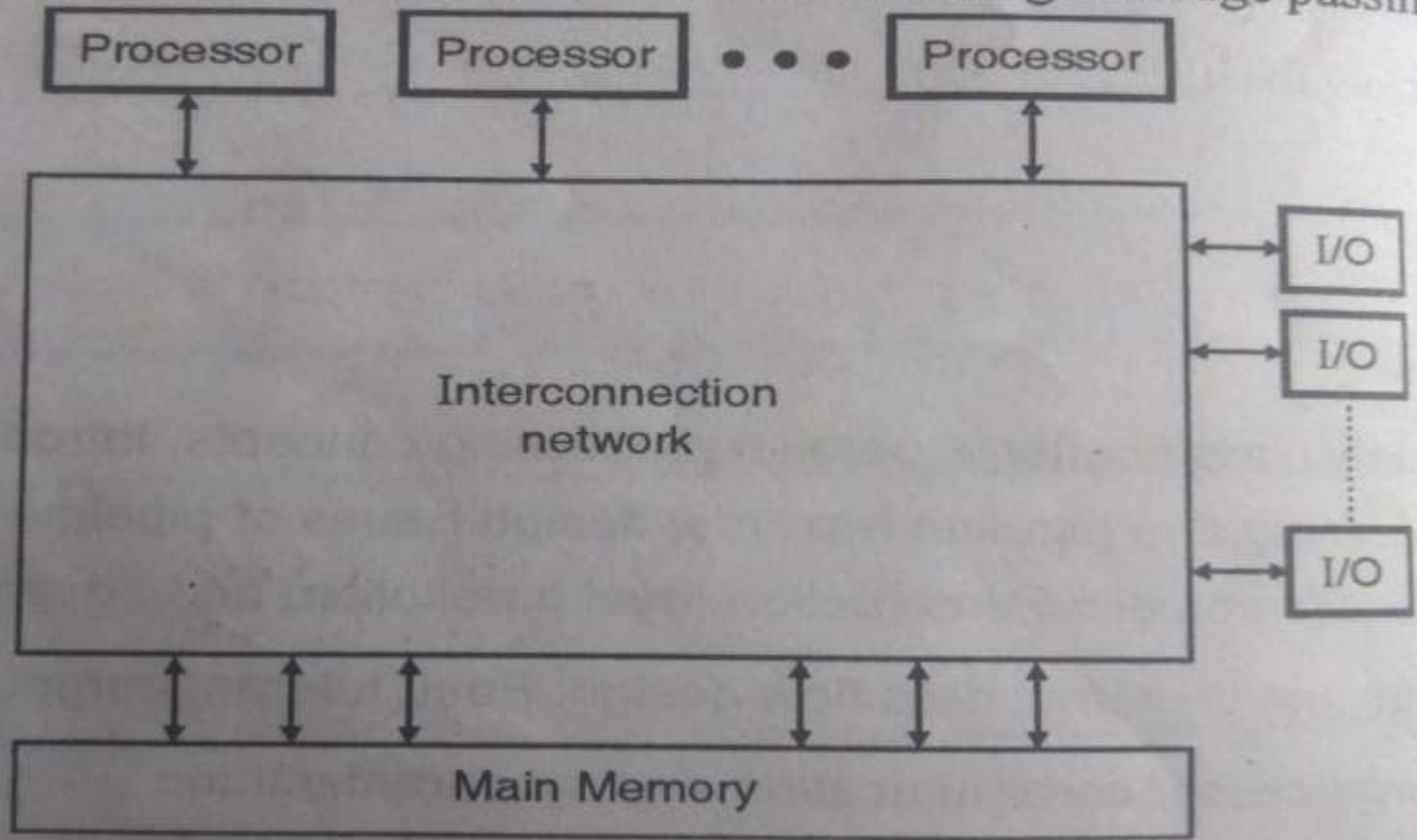


Fig. 6.1 : A tightly coupled multiprocessor system

Parallel Processing Concepts

1. Tightly coupled systems(shared memory):
 - High degree of resource sharing
 - Can be used to speed up the execution of a single large program in time-critical applications.
 - Number of processors share main memory, I/O facilities and interconnected by a bus.
 - All processors can perform the same functions or specialized functions.
 - Entire system is controlled by a single OS.
 - Parallel systems to solve a single problem.
 - Also called as Symmetric Multiprocessor System.

Parallel Processing Concepts

2. Loosely coupled systems(Distributed memory):

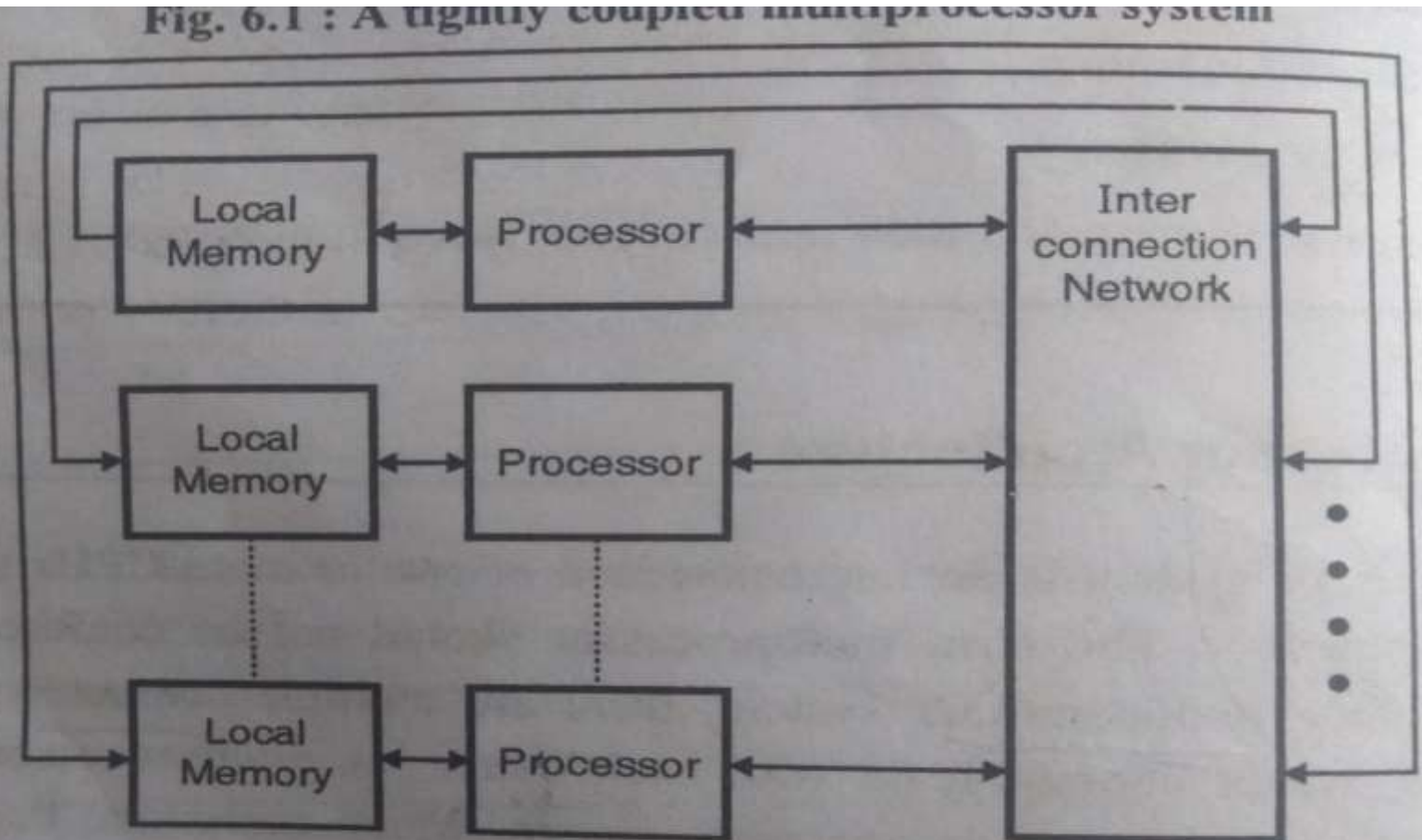


Fig. 6.2 : A loosely coupled multiprocessor configuration

Parallel Processing Concepts

2. Loosely coupled systems(Distributed memory):

- Shared memory is distributed to all processors(local memory).
- Faster memory access.
- send() and receive() functions for interaction among processors.
- More scalable.
- Different OS for different processor.
- NUMA(Non Uniform Memory Access)

Flynn's Classification

- Based on
 - Internal organization of processors
 - Interconnection Structures

Instruction Stream

Instruction from main memory to processor

Data Stream

Operands flowing to and from the processor

FLYNN's CLASSIFICATION

Multiple Processor Organization

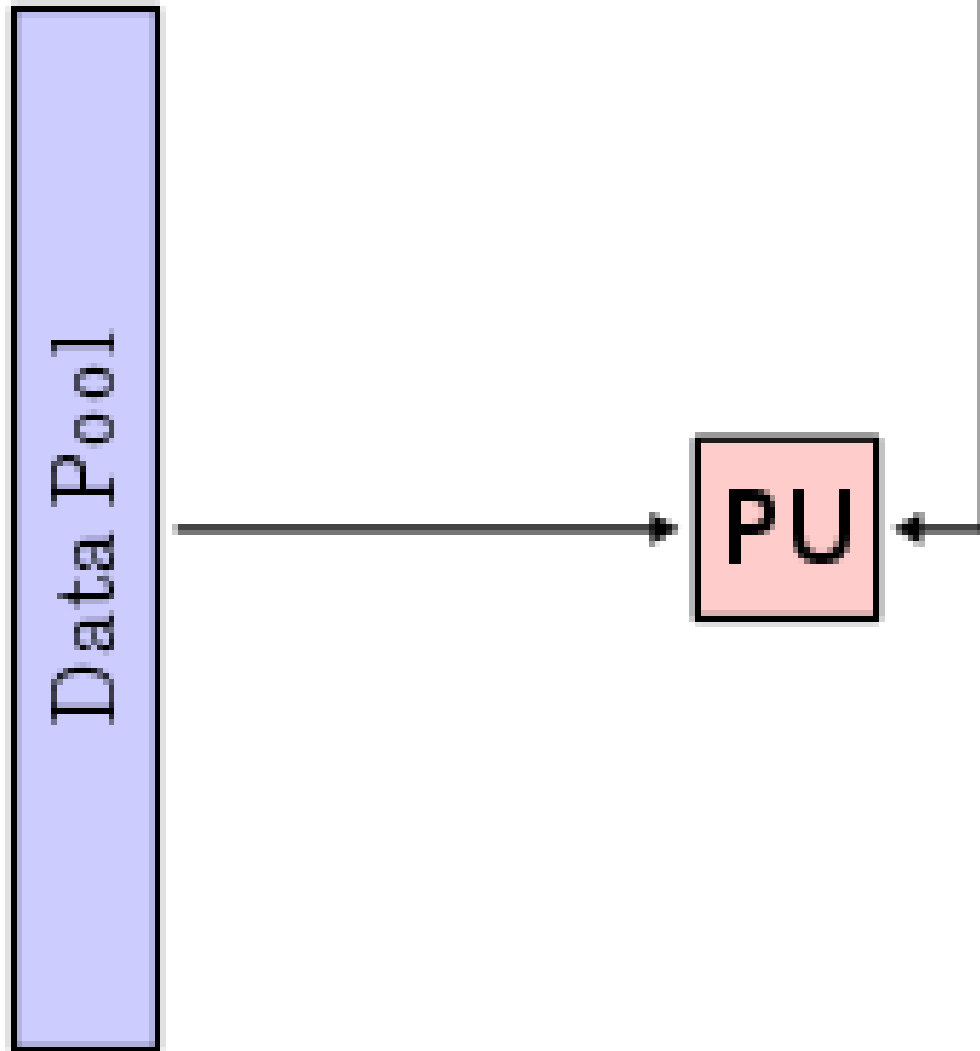
- Single instruction, single data stream - SISD
- Single instruction, multiple data stream - SIMD
- Multiple instruction, single data stream - MISD
- Multiple instruction, multiple data stream- MIMD

SISD

Instruction Pool

Data Pool

PU



SISD

MISD (Multiple instruction single data)

MIMD (Multiple instruction multiple data).

1 SISD :

Conventional computer with single CPU come under this category.

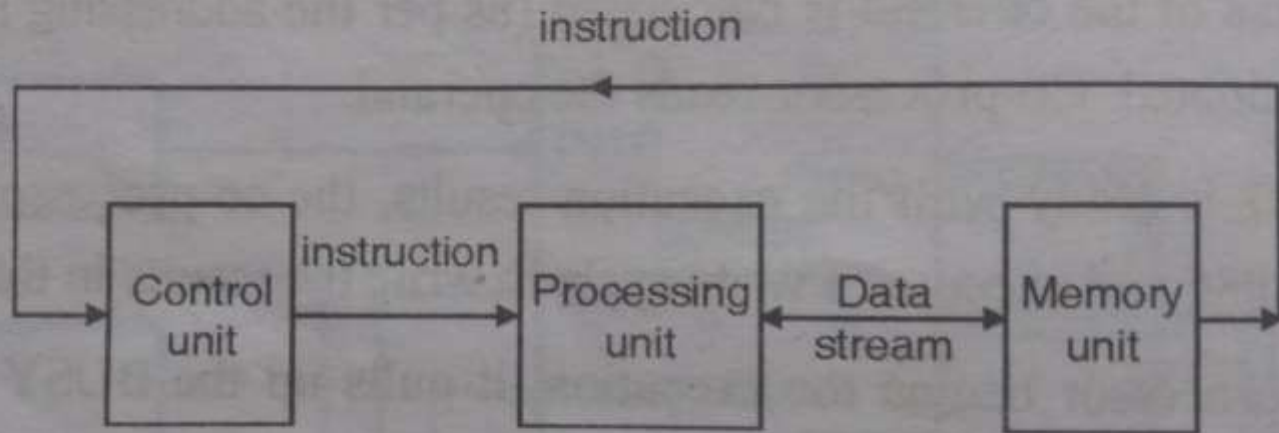


Fig. 6.5 : SISD signal processor architecture

In a conventional computer, CPU fetches an instruction and instruction a single data stream. A SISD architecture has almost no parallelism.

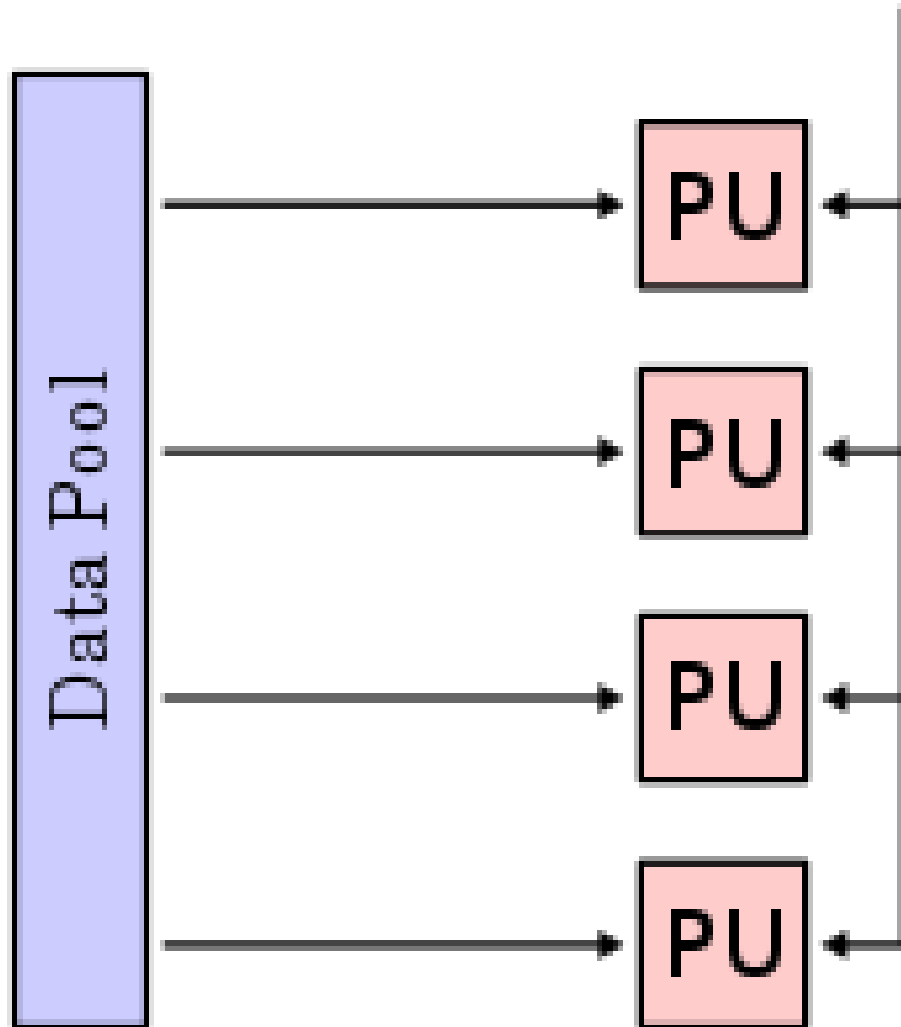
SIMD :

Single Instruction, Single Data Stream - SISD

- Single processor
- Single instruction stream
- Data stored in single memory
- Uni-processor
- E.g. conventional computers, IBM 701, IBM 1620
- CPU fetches an instn and instn is executed using a single data stream.

SIMD

Instruction Pool



Single Instruction, Multiple Data Stream - SIMD

- Single machine instruction
- Controls simultaneous execution
- Number of processing elements
- Each processing element has associated data memory
- Each instruction executed on different set of data by different processors
- Processing elements operate under the control of central controller.
- Vector and array processors

SIMD

In a conventional computer, there is a single data stream. A SISD architecture has almost no parallelism.

SIMD :

An array processor is an example of SIMD architecture. Each processing unit can operate independently on a different data stream. Each processing element can have its local memory. These processing elements operate under the control of central controller.

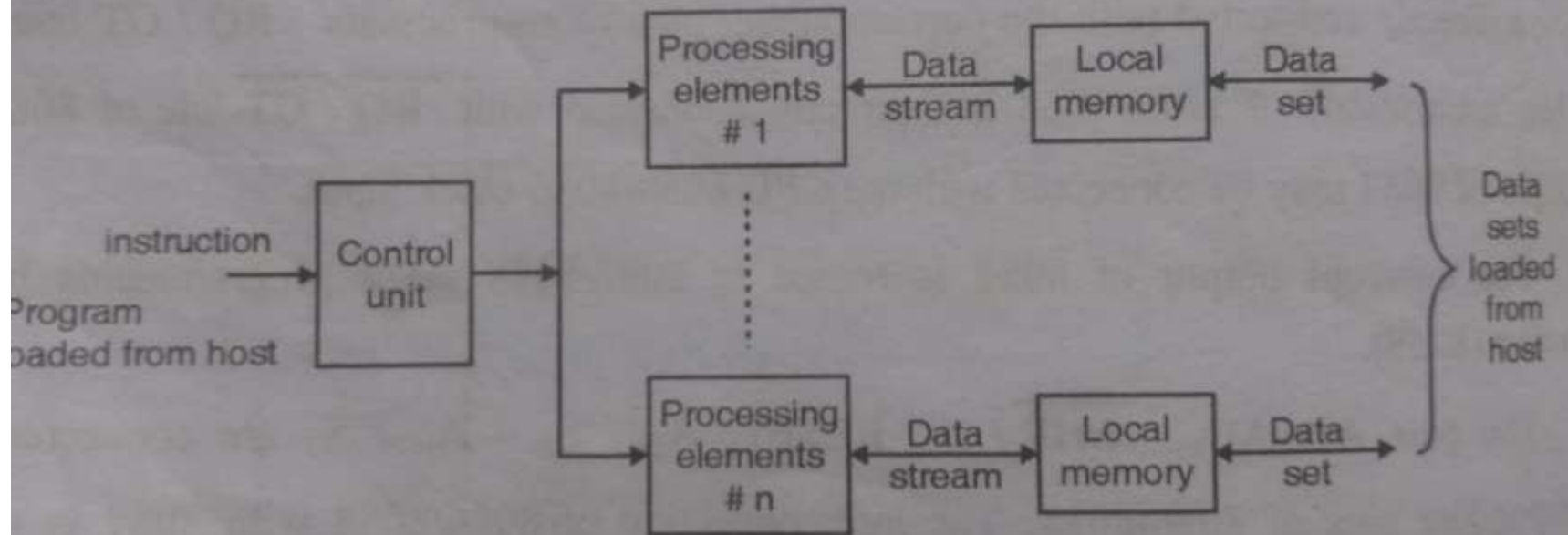


Fig. 6.6 : SIMD architecture with distributed memory

MISD :

Several parallel computers fit into this class. Fault-tolerant computer where several C

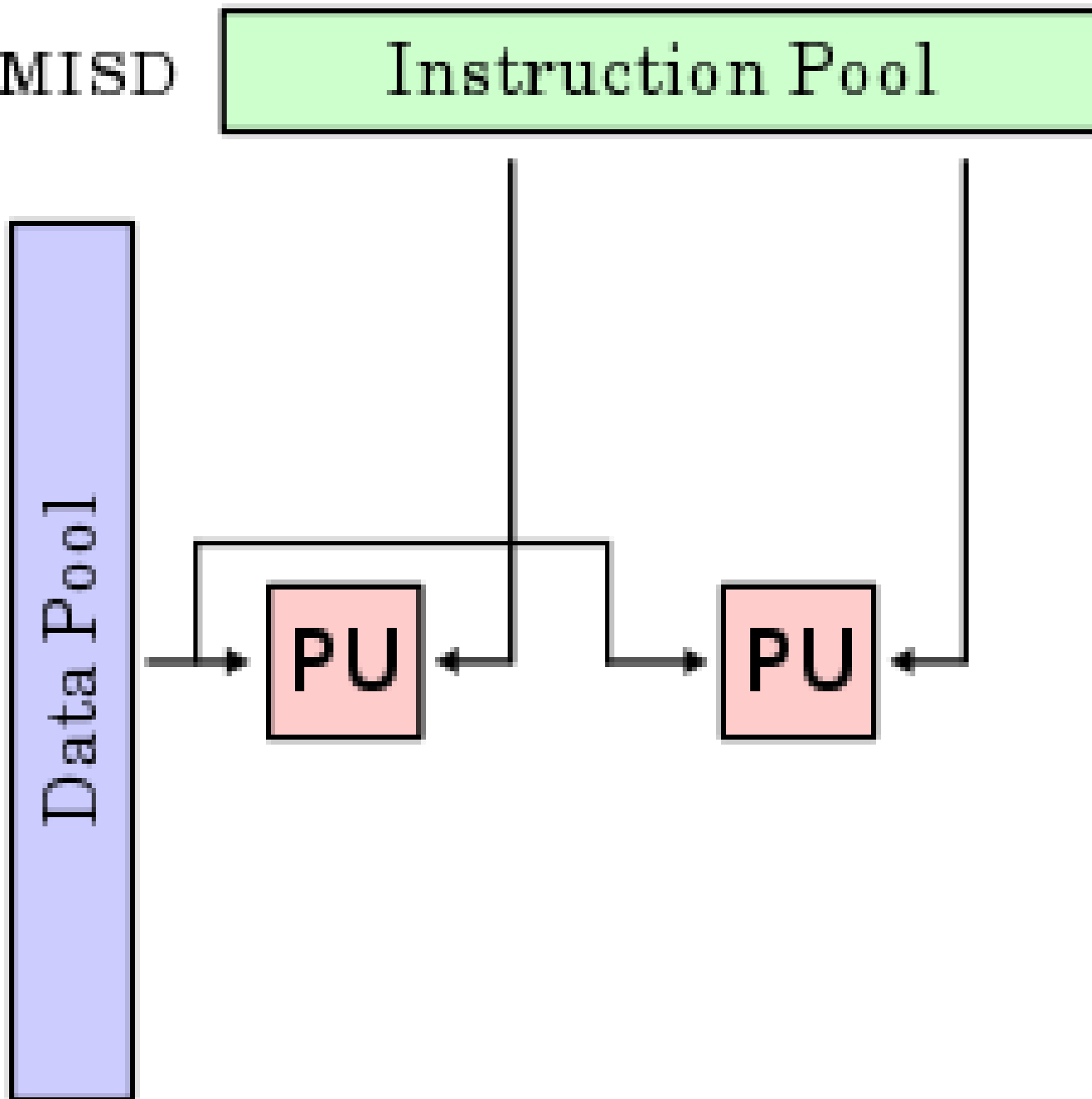
MISD

Instruction Pool

Data Pool

PU

PU



MISD

Computer Architecture (MU) 6-7 SPARC & Systolic Architecture

process the same data using different programs are MISD. A single data stream can pass through a linear array of processors executing different programs.

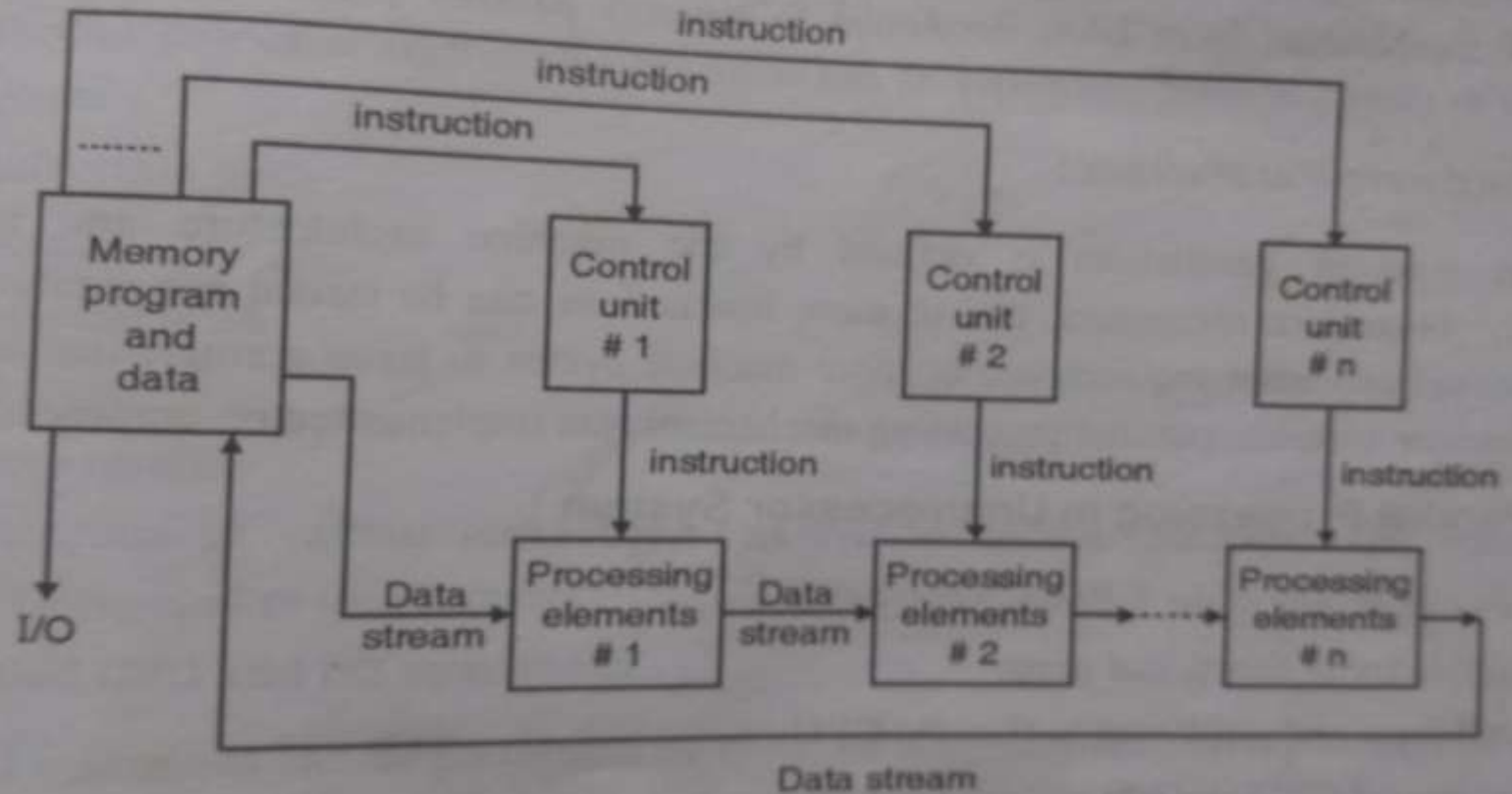


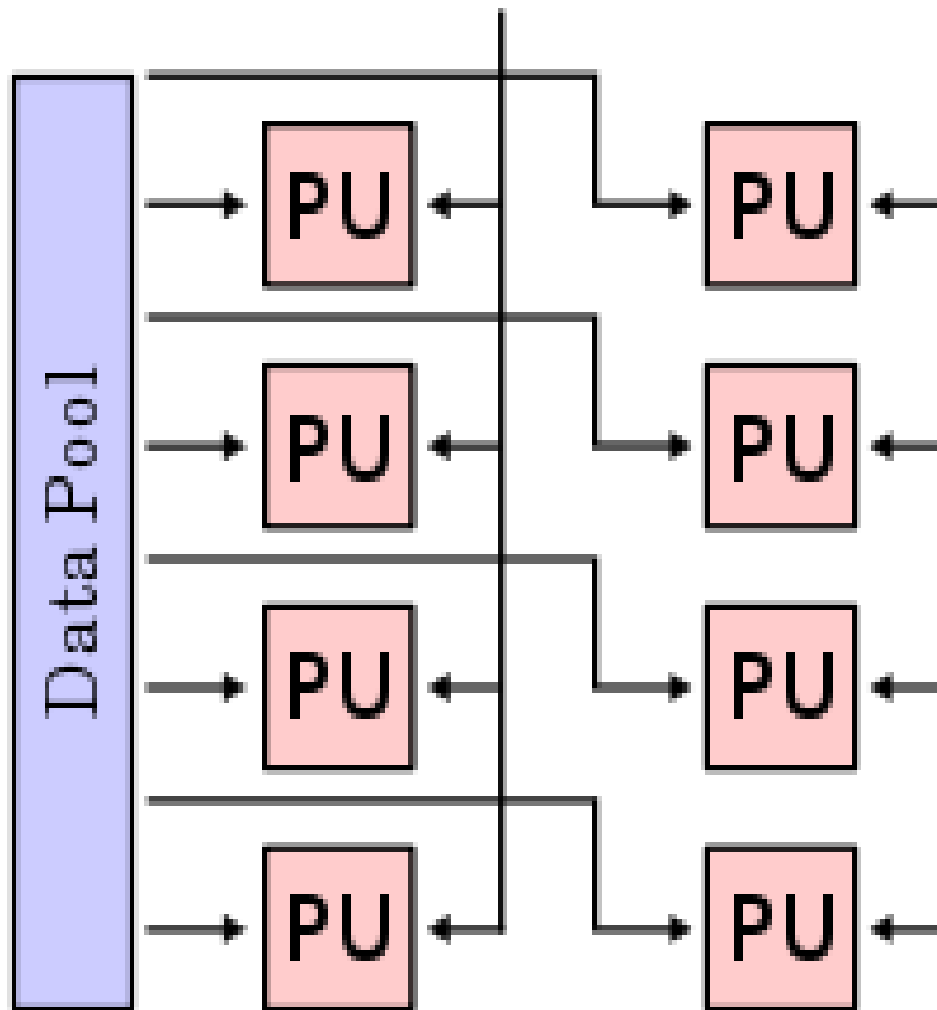
Fig. 6.7 : MISD architecture with shared memory

Multiple Instruction, Single Data Stream - MISD

- Sequence of data
- Transmitted to set of processors
- Each processor executes different instruction sequence
- A single data stream can flow through a linear array of processors executing different programs.
- Never been implemented

MIMD

Instruction Pool



Multiple Instruction, Multiple Data Stream- MIMD

independent programs simultaneously. Most parallel/ vector computer are based on MIMD model. There are two major classes of parallel computers :

- (1) Shared-memory multiprocessors.
- (2) Message-passing multicomputers.

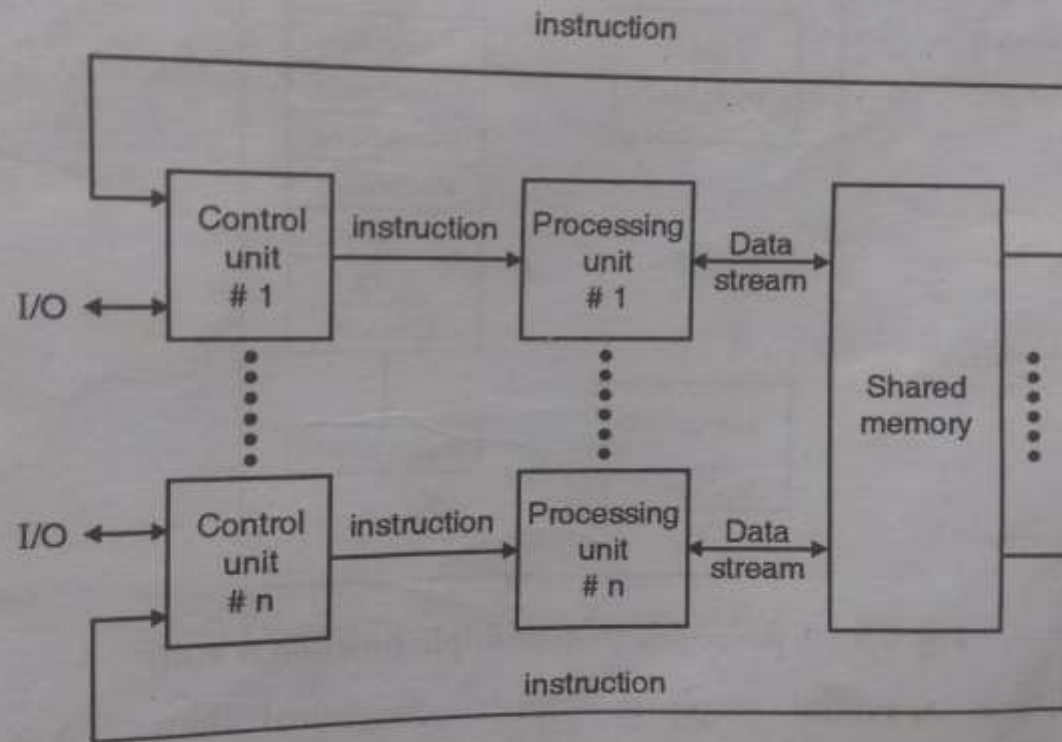


Fig. 6.8 : MIMD architecture with shared memory

Multiple Instruction, Multiple Data Stream-MIMD

- Set of processors
- Computer with multi-processors.
- These processors can be execute independent programs simultaneously.
- Simultaneously execute different instruction sequences
- Different sets of data
- Shared Memory multiprocessors and message-passing multicomputers.
- E.g. Tightly coupled ,loosely coupled systems

Introduction to pipeline processing and pipeline hazards

- **Pipelining**- Temporal overlapping of processing
- Subdivide input task(process) into a sequence of subtasks
- Each executed by **specialized hardware** that operates concurrently with other stages of pipeline

Pipelining allows the next **instructions** to be fetched while the processor is performing arithmetic operations

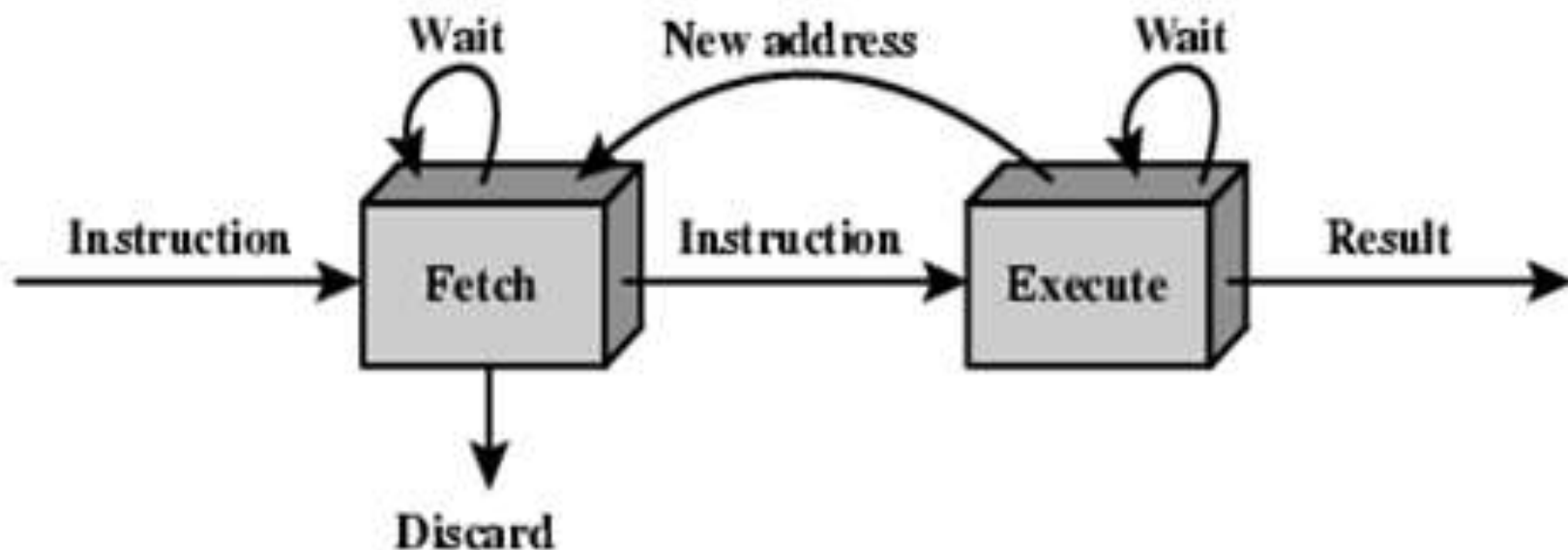
Holds them in a buffer close to the processor until each **instruction** operation can be performed.

The staging of **instruction** fetching is continuous.

Two Stage Instruction Pipeline

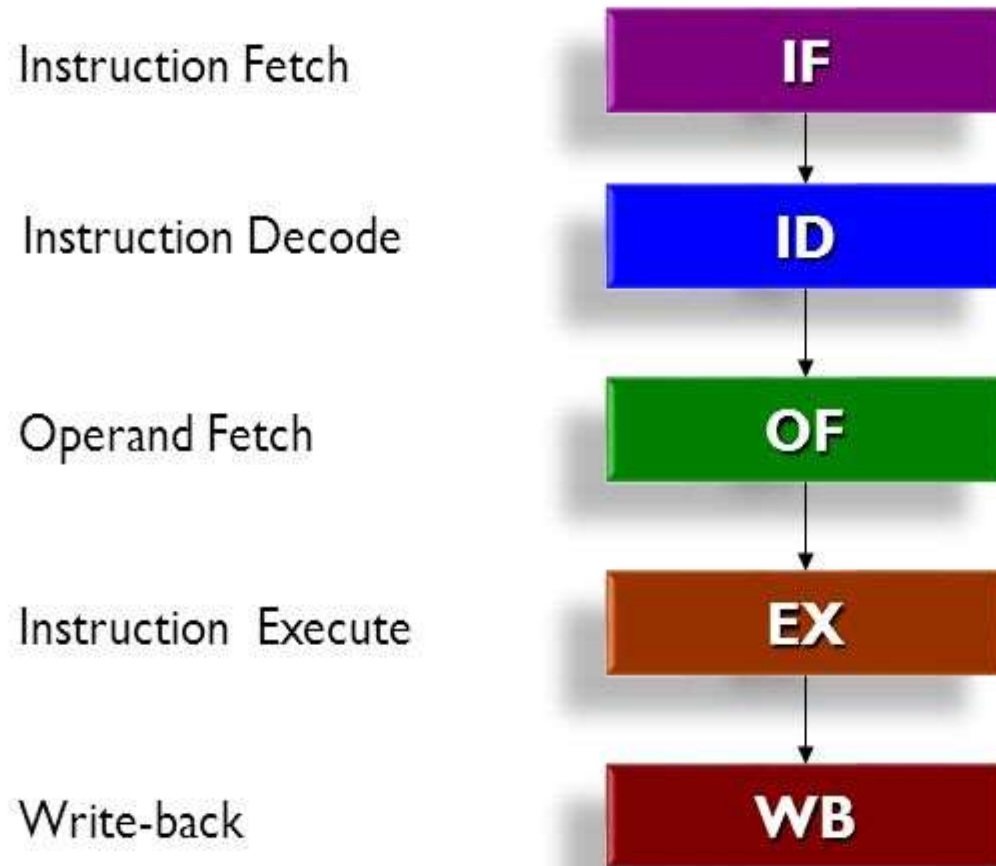


(a) Simplified view



(b) Expanded view

The Generic Instruction Pipeline



SIX STAGE OF INSTRUCTION PIPELINING

- **Fetch Instruction(FI)**

Read the next expected instruction into a buffer

- **Decode Instruction(DI)**

Determine the opcode and the operand specifiers.

- **Calculate Operands(CO)**

Calculate the effective address of each source operand.

- **Fetch Operands(FO)**

Fetch each operand from memory. Operands in registers need not be fetched.

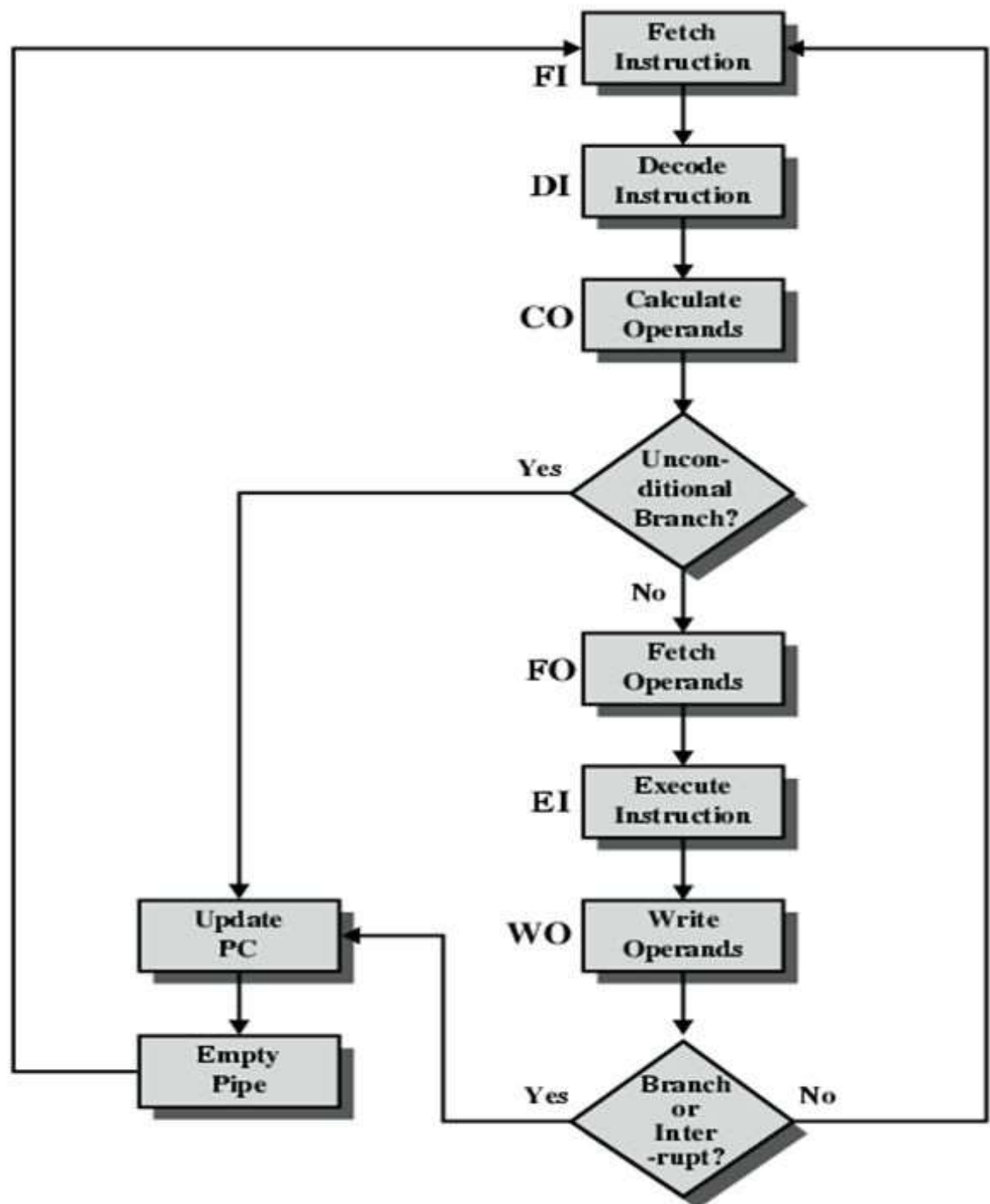
- **Execute Instruction(EI)**

Perform the indicated operation and store the result

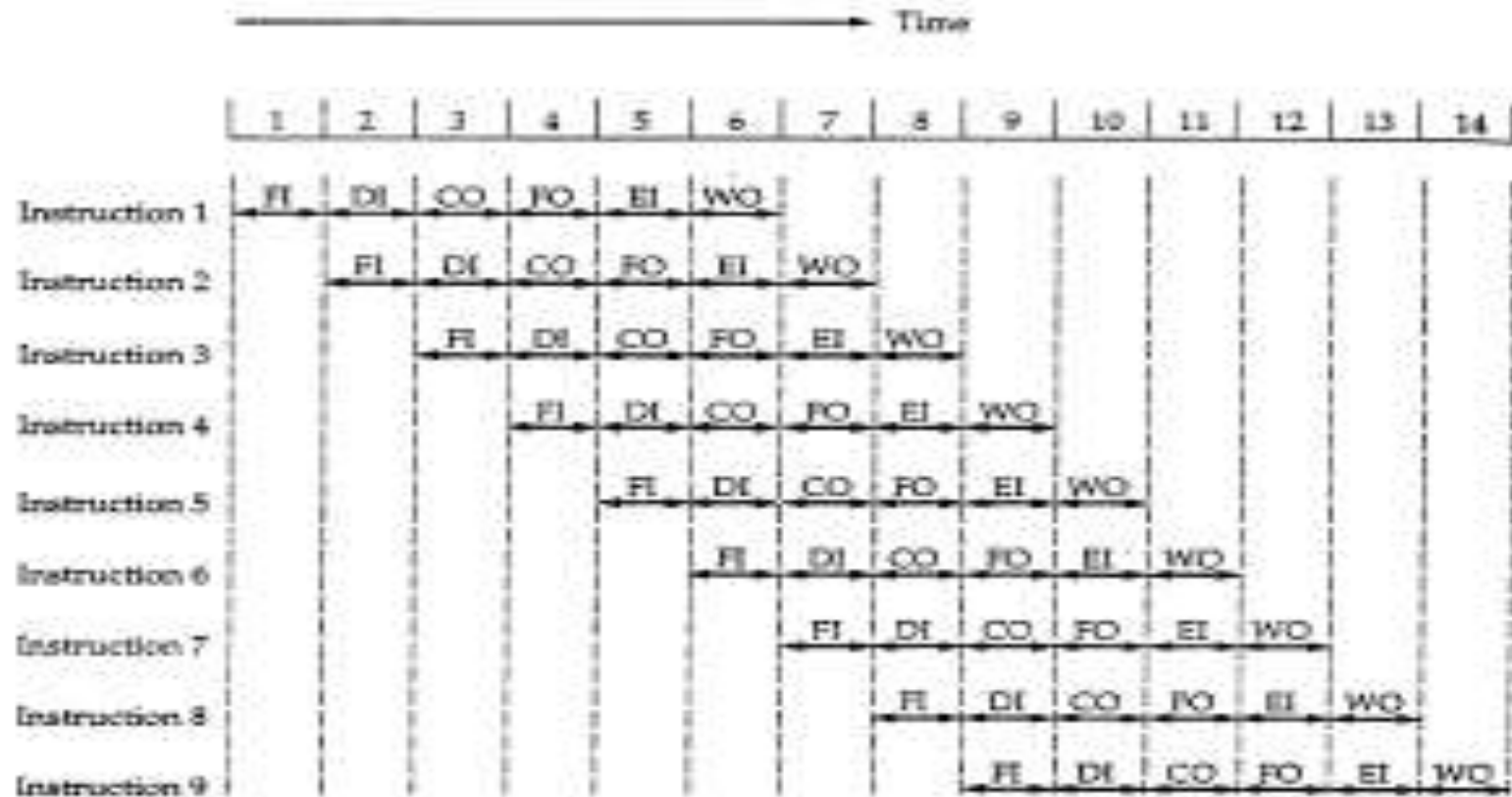
- **Write Operand(WO)**

Store the result in memory.

Six Stage Instruction Pipeline



Timing diagram for 6 stage instruction pipeline



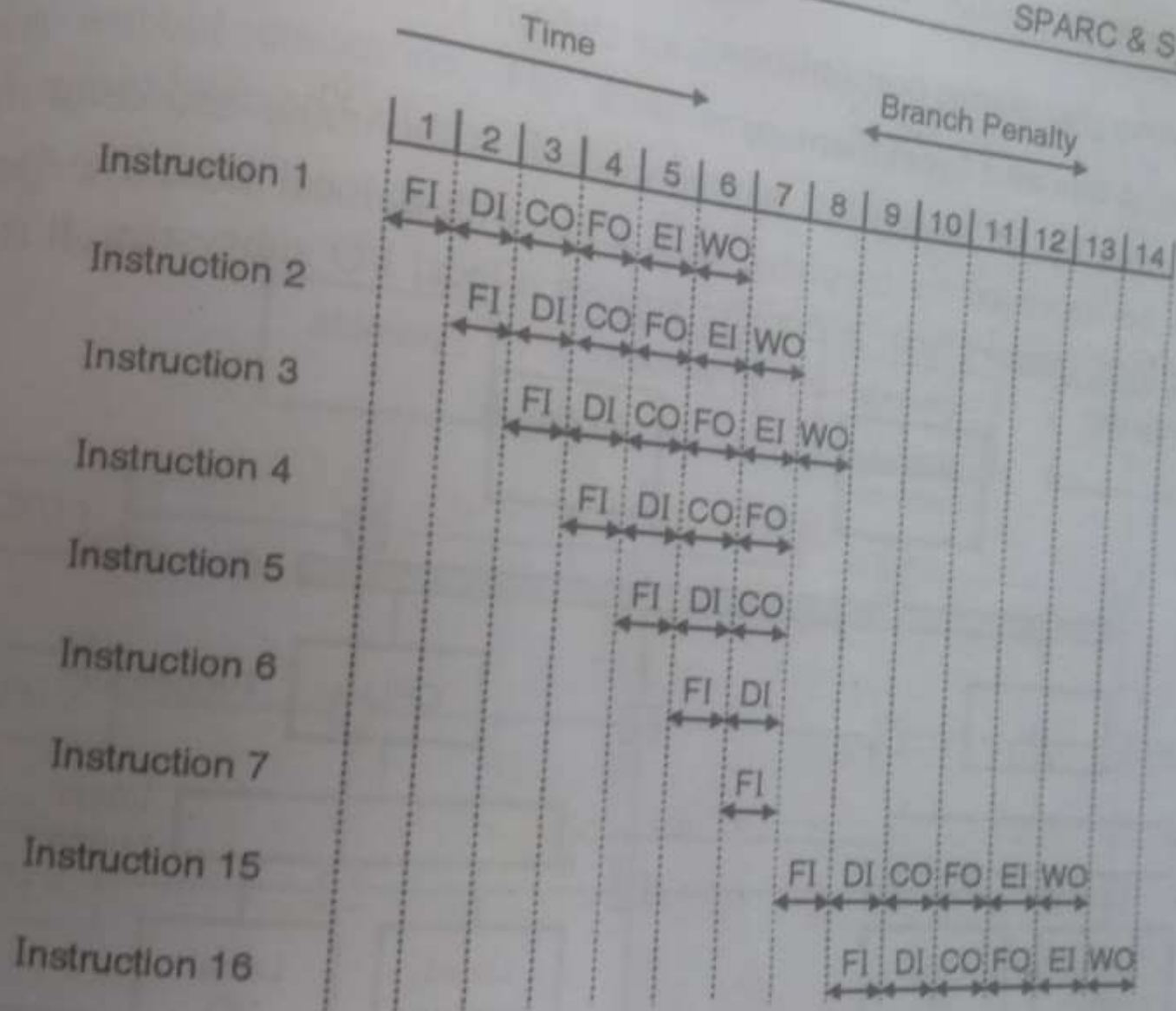


Fig. Ex-3(b) : The effect of a conditional branch on instruction pipeline operation

3000 Add AX, BX

3001 jmp 8000

3002 mov AX,BX

8000 mul BX

Pipeline hazards

- Any reason that causes the pipeline to stall is called hazard or conflict
- Resource usage
(inter instruction dependencies)
- Job scheduling problems

3 types of Hazards


- RESOURCE CONFLICTS
 - insufficient resources
 - Solution is to use separate instruction and data memory
- DATA or DATA DEPENDENCY CONFLICTS
 - instruction in pipeline depend on result of previous instruction which still in pipeline yet to be completed
 - Solution shuffle program instructions

3 types of Hazards

- Branch Difficulties:

--to change PC with new address

RESOURCE HAZARD

- Stalling pipeline () causes **Structural Hazard**
- Commonly need Memory Access
 - Use separate cache for **instruction** and **data**

DATA HAZARD

- RAW
 - WAW
 - WAR
 - RAR?
- Rearrange the pipeline- pipeline scheduling

RAW:

Add AX,BX

Mov DX,AX

DATA HAZARD

- RAW
- WAW
- WAR
- RAR?

- Rearrange the pipeline- pipeline scheduling

WAW:

Add AX,BX

SUB DX,AX

WAR:

Mov AX,DX

Add AX BX

BRANCH HAZARDS

- Multiple streams
- Prefetch branch target
- Loop buffer (contains most recently fetched instructions)
- Branch prediction
- Delayed branch

Design issues of pipeline architecture

- Instruction Pipeline design
 - Instruction Issuing
 - In order issuing, out of order issuing or re order issuing
- Arithmetic Pipeline design
 - Fixed point , floating point , integer arithmetic

Principles of designing pipelined processors

- Proper data buffering to avoid congestion and smooth pipeline operations
- Instruction dependence relationship
- Logic hazards should be detected and resolved
- Avoid Collisions and structural hazards by proper sequencing
- Reconfiguration of the pipeline should be possible