Module-6

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

Flynn’s Classification

- Based on
  - Internal organization of processors
  - Interconnection Structures

  Instruction Stream

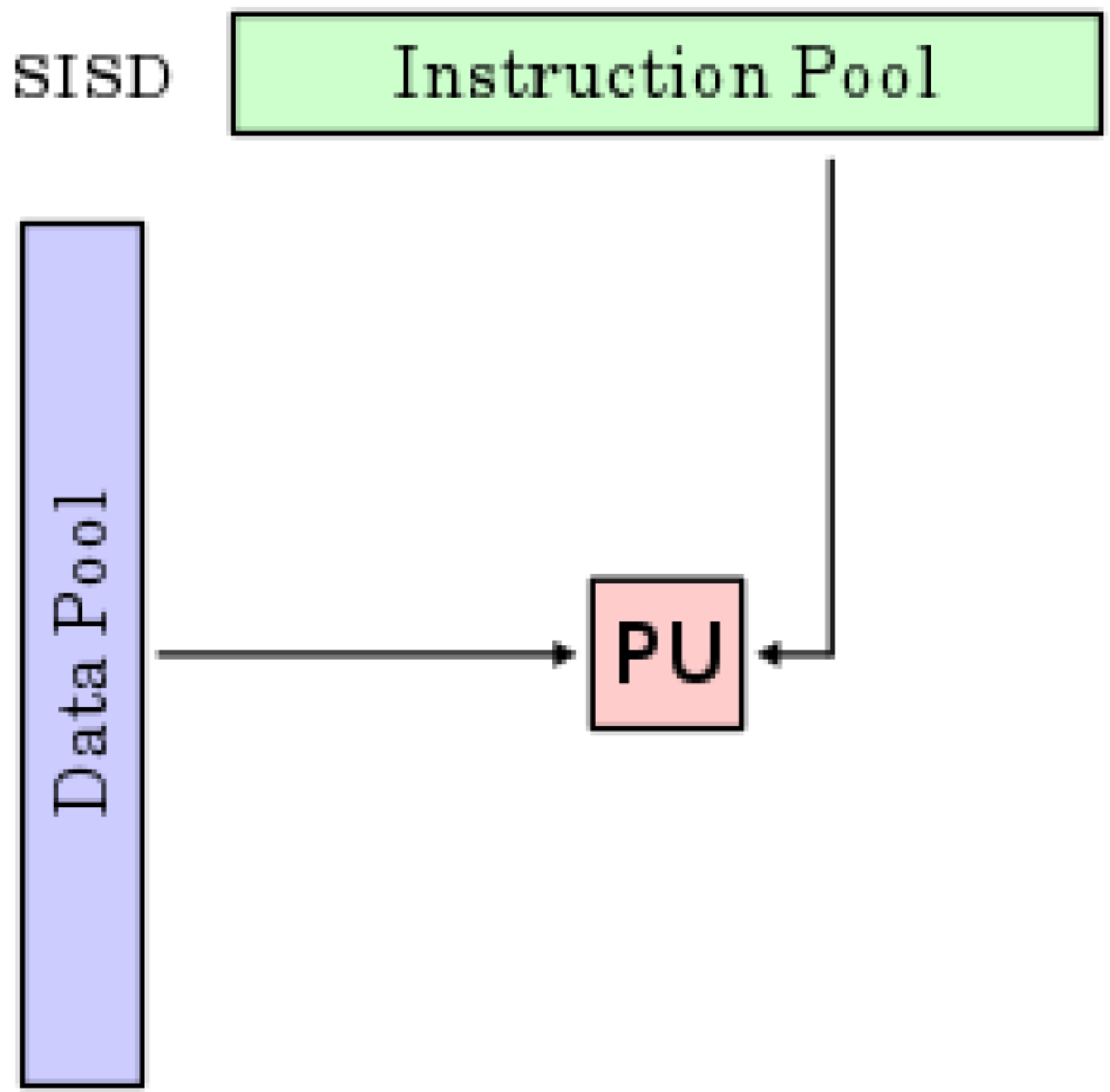
  Instruction from main memory to processor

  Data Stream

  Operands flowing to and from the processor

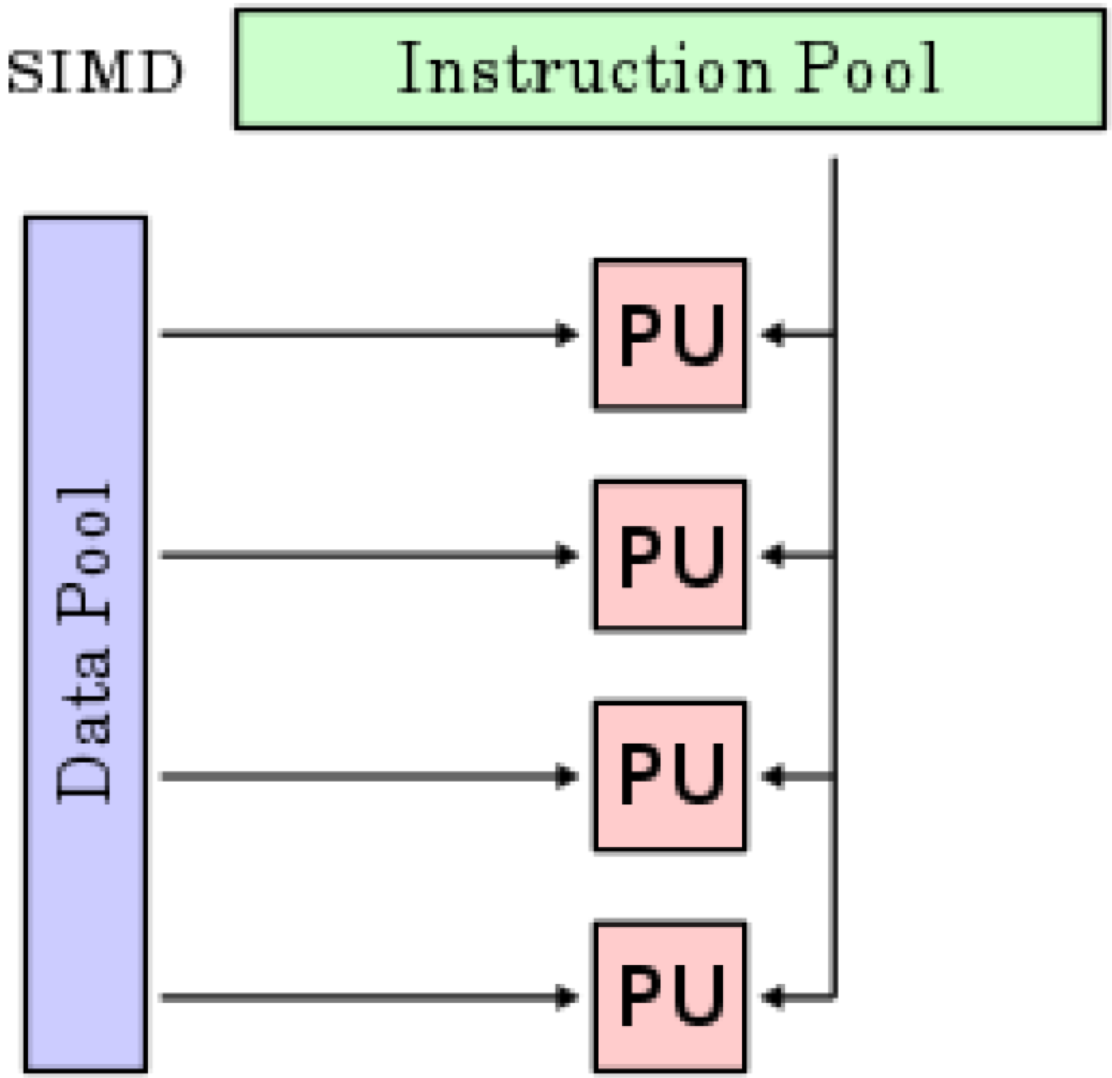
FLYNN’s CLASSIFICATION
Multiple Processor Organization

- 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

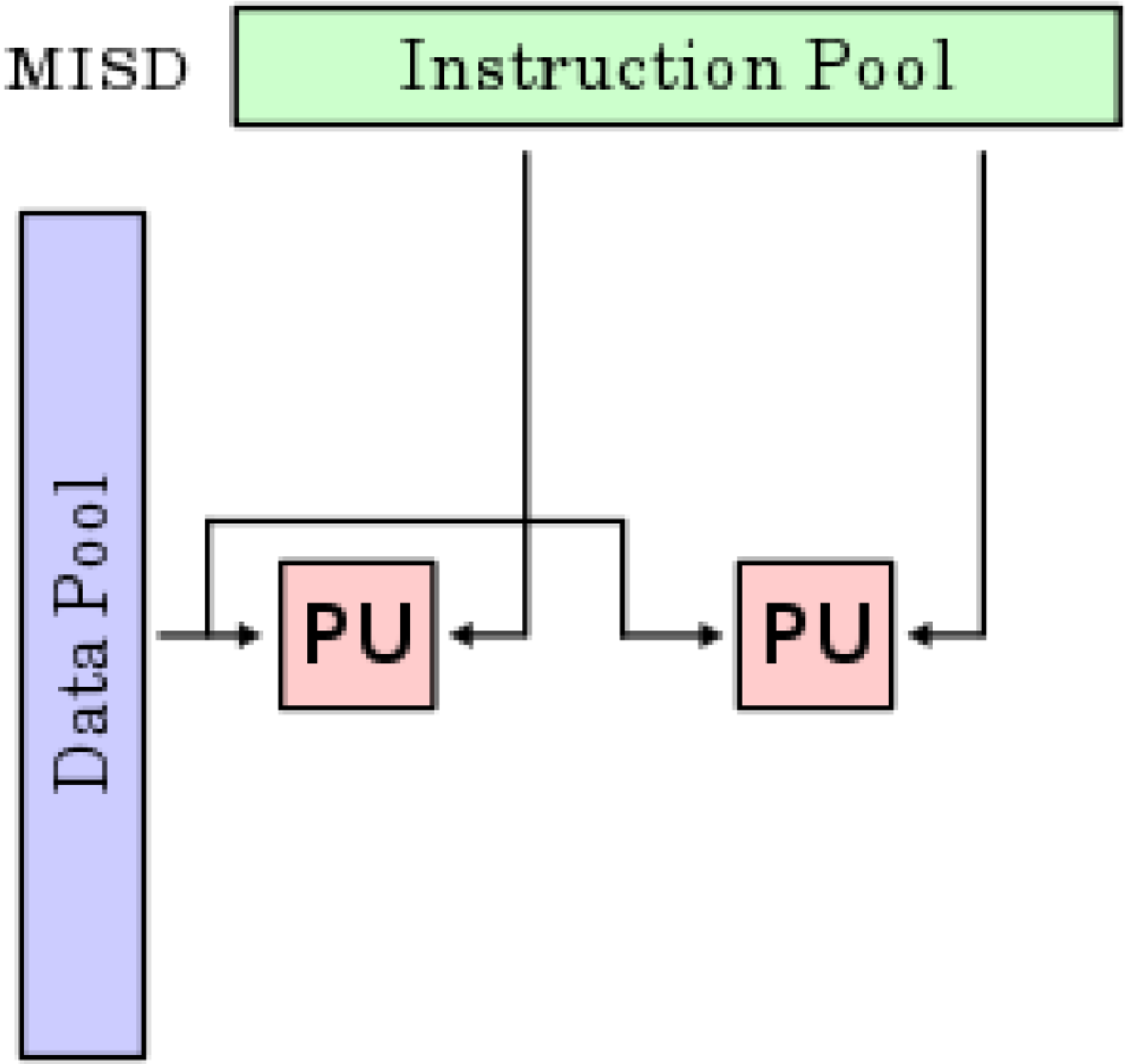
Single Instruction, Single Data Stream - SISD

- Single processor
- Single instruction stream
- Data stored in single memory
- Uni-processor


SIMD
Instruction Pool
Data Pool
PU
PU
PU
PU

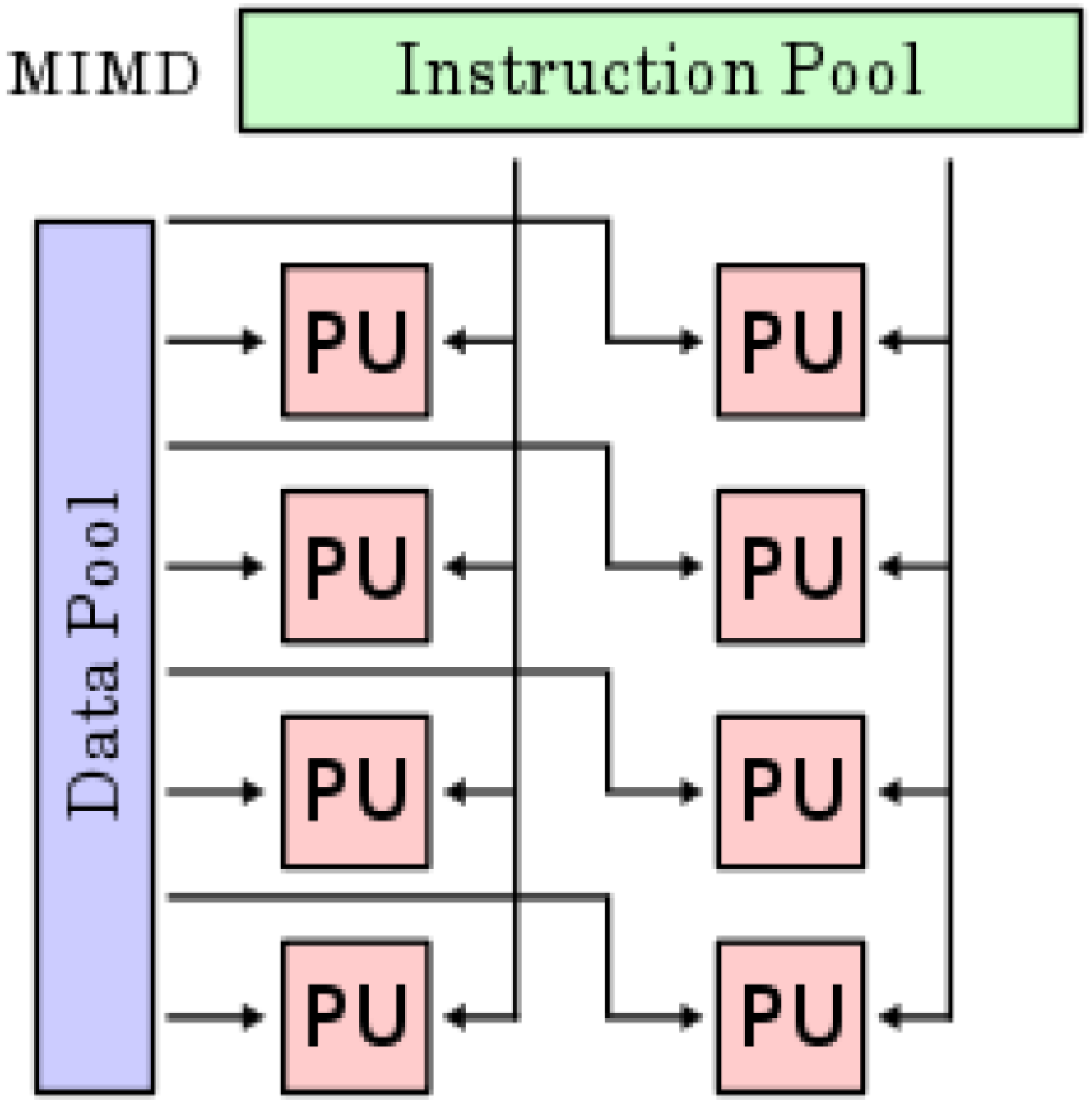
Single Instruction, Multiple Data Stream - SIMD

- 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
- Vector and array processors


MISD
Instruction Pool
Data Pool
PU
PU

Multiple Instruction, Single Data Stream - MISD

- Sequence of data
- Transmitted to set of processors
- Each processor executes different instruction sequence
- Never been implemented


MIMD
Instruction Pool
Data Pool
PU
PU
PU
PU
PU
PU
PU
PU

Multiple Instruction, Multiple Data Stream- MIMD

- Set of processors
- Simultaneously execute different instruction sequences
- Different sets of data
- SMPs, NUMA 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.

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)**

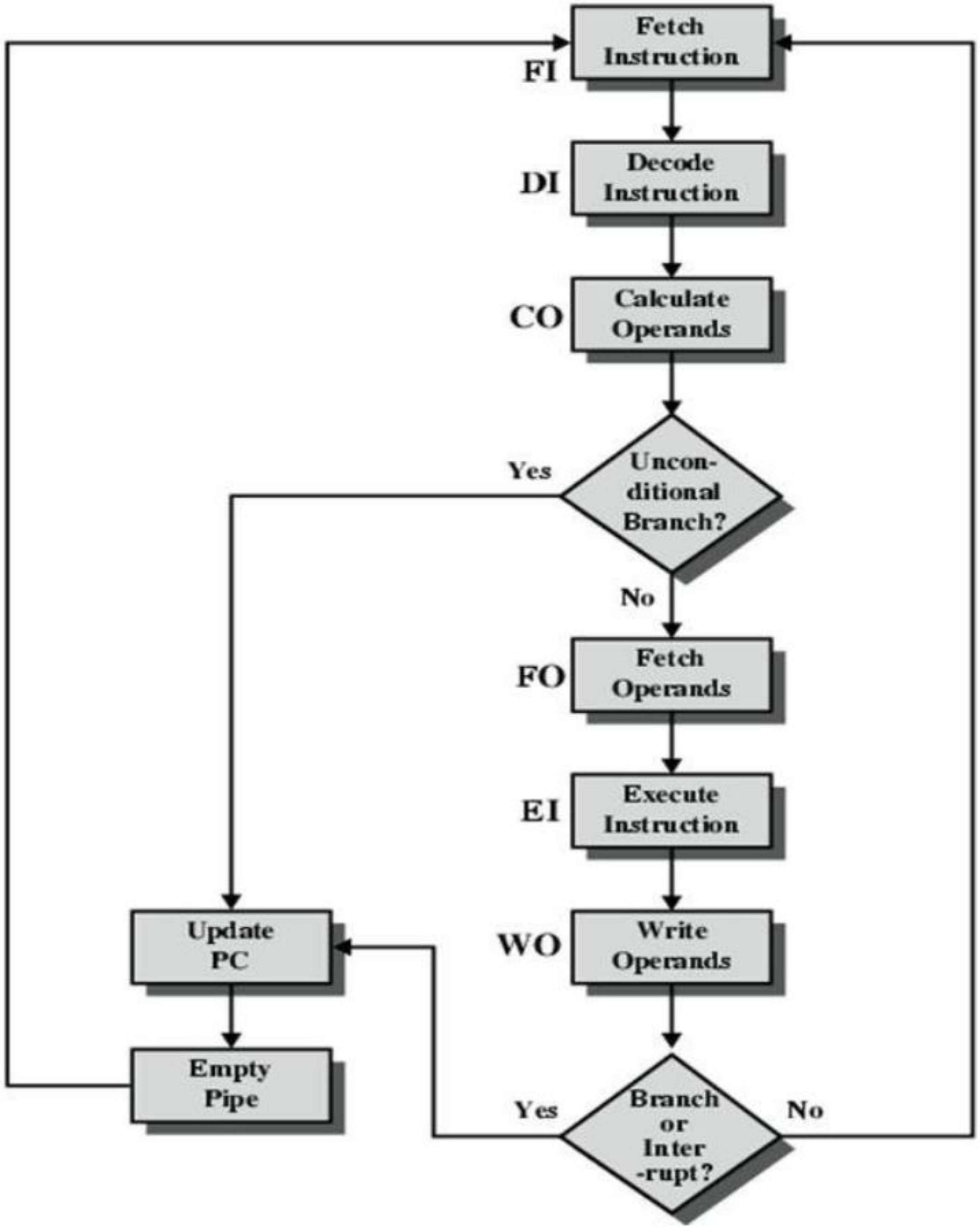
  Calculate the effective address of each source operanc
- **Fetch Operands(FO)**

  Fetch each operand from memory. Operands in regist 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

Time →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

Figure 12.10 Timing Diagram for Instruction Pipeline Operation

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
- DATA or DATA DEPENDENCY CONFLICTS
  - instruction in pipeline depend on result of previous instruction which still in pipeline yet to be completed
- BRANCH DIFFICULTIES-Contents of PC get changed

3 Types of HAZARD

- RESOURCE HAZARDS(Structural hazard)
- DATA HAZARDS
- CONTROL HAZARDS

Resource hazard

- If an operand for an instruction is in memory , rather than a register. Operand read to or write from memory cannot be performed in parallel with an instruction Fetch.

- Another example of a resource conflict is a situation in which multiple Instructions are ready to enter the execute instruction phase and there is a single ALU.

Resource Hazard

Clock cycle

Instrutcion	1	2	3	4	5	6	7	8	9
I1	FI	DI	FO	EI	WO				
I2		FI	DI	FO	EI	WO			
I3			FI	DI	FO	EI	WO		
I4				FI	DI	FO	EI	WO	

(a) Five-stage pipeline, ideal case

Clock cycle

Instrutcion	1	2	3	4	5	6	7	8	9
I1	FI	DI	FO	EI	WO				
I2		FI	DI	FO	EI	WO			
I3			Idle	FI	DI	FO	EI	WO	
I4					FI	DI	FO	EI	WO

(b) I1 source operand in memory

Figure 12.15 Example of Resource Hazard

DATA HAZARD

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

DATA HAZARD

- Occurs when-conflict in the access of an operand location.
- Two instructions in a program are to be executed in sequence and both access a particular memory or register operand.
- The operand value to be updated –producing a different result than would occur with strict sequential execution.
- The destination of one instruction should not be a source of immediate next instruction.
- Example, :
- ADD EAX, EBX
- /* EAX = EAX + EBX
- SUB ECX, EAX
- /* ECX = ECX - EAX

DATA HAZARD

- To maintain correct operation, the pipeline must stall for two clocks cycles.

- Otherwise-results in inefficient pipeline usage.
- Eg;-add
- Nop
- Nop
- Sub

3-types of Data Hazards

- Read after write (RAW), or true dependency: An instruction modifies a register or memory location and a succeeding instruction reads the data in that memory or register location. A hazard occurs if the read takes place before the write operation is complete.

- Write after read (RAR), or antidependency: An instruction reads a register or memory location and a succeeding instruction writes to the location. A hazard occurs if the write operation completes before the read operation takes place.

- Write after write (WAW), or output dependency: Two instructions both write to the same location. A hazard occurs if the write operations take place in the reverse order of the intended sequence.

BRANCH HAZARDS

- Flush Pipeline
- Delayed branching
- Conditional branching

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