MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh, Tangail-1902

Assignment

Department of : Information and Communication Technology

Assignment No : 01

Name of the assignment : Parallel processing, Flynn's Classification, pipelining, and Data Hazards

Course Title : Computer Architecture and Organization

Course Code : ICT-3207

Submitted by	Submitted to
Name : Kuldip Saha Mugdha	Md. Anowar Kabir
ID : IT22018 Session : 2021-22	Lecturer
Year : 3rd Semester : 2nd	
Dept. of ICT, MBSTU	Dept. of ICT, MBSTU

Date of Performance : 11/01/2026 Date of Submission : 08/03/2026

1. Parallel Processing:

Parallel processing is a computing technique in which multiple processing units perform tasks simultaneously. Instead of executing one instruction at a time in sequence, parallel processing allows two or more operations to be carried out at the same time.

The main objective of parallel processing is to increase system performance and throughput. By executing several instructions concurrently, the total execution time is reduced, and more work can be completed within the same period.

Parallel processing is especially useful in:

- Scientific computations.
- Engineering simulations.
- Graphics processing
- Large database systems
- Artificial intelligence applications

As technology advances, increasing clock speed alone is not sufficient to achieve higher performance. Therefore, designers improve performance by adding multiple functional units or processors that operate simultaneously.

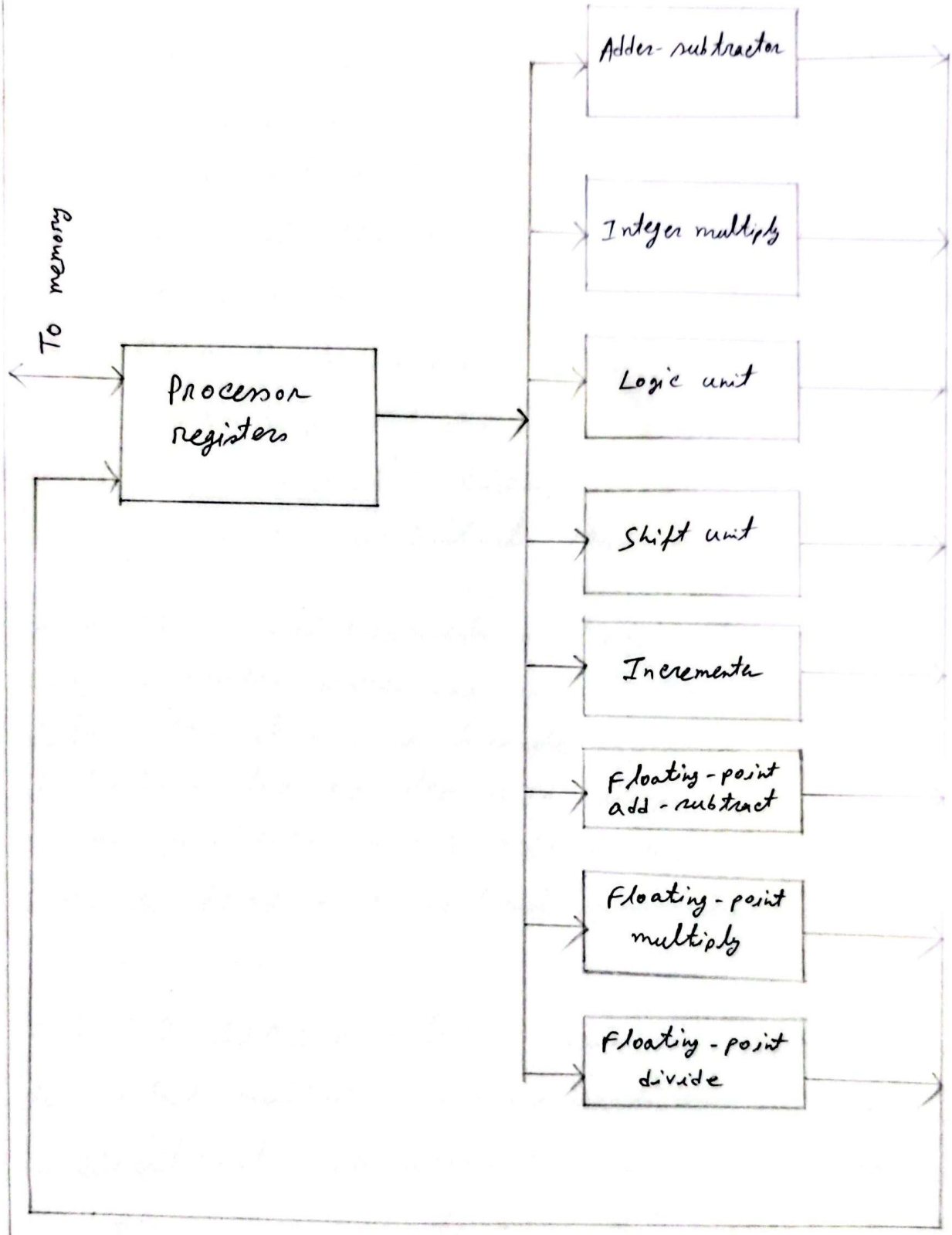

Figure 9.1 : Processor with multiple functional units

2. Flynn's Classification of Computer:

Flynn's classification categorizes computer architectures based on the number of instruction streams and data streams they process simultaneously. It focuses on how instructions streams and data flow through a system.
The four categories are:

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

1. SISD: SISD represents a traditional sequential Computer system where one instruction operates on one data item at a time. A single processor executes instructions step by step in a linear manner. There is no parallelism in this architecture, and it is Commonly found in conventional single-core computers.

2. SIMD: SIMD architecture uses one instruction stream to control multiple processing units that operates on different data items simultaneously. The same operation is applied to many data elements at the same time. This type of architecture is widely used in vector processors and graphics processing

units (GPUs) for tasks like image processing and scientific calculations.

3. MISD: MISD architecture involves multiple instruction streams operating on a single data streams. In this system, the same data is processed by different instructions or algorithms. It is rarely used in general-purpose computing but may be applied in specialized systems such as fault-tolerant or safety-critical applications.

4. MIMD: MIMD architecture allows multiple processors to execute different instructions on different data simultaneously. Each processor works independently, making it suitable for parallel and distributed computing systems. Modern multi-core processors and high-performance computing systems commonly use this architecture.

3. Pipelining :

Pipelining is a technique used to increase instruction throughput by overlapping multiple instruction phases. Instead of completing one instruction before starting another, pipelining divides the instruction cycle into stages, and multiple instructions are processed simultaneously in different stages.

It is similar to an assembly line in a factory.

Basic Concept: A pipeline consists of multiple stages connected in sequence. Each stage performs a part of the operation.

Typical instruction pipeline stages:

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execute (EX)
4. Memory Access (MEM)
5. Write Back (WB)

Each ~~has~~ stage has:

→ A register

→ A Combinational circuit

Data moves from one stage to another at each clock pulse.

Example of Pipeline Operation:

Consider the arithmetic operation:

$A_i \times B_i + C_i$

This operation can be divided into smaller steps such as loading operands, performing multiplication, and then performing addition. In a pipelined system, each of these steps is assigned to a different stage.

For example:

- Stage 1: Load A_i and B_i
- Stage 2: Multiply A_i and B_i
- Stage 3: Add C_i to the product

While one set of data is being multiplied, another set can be loaded, and a previous result can be added. Thus, multiple operations proceed in parallel within different stages of the pipeline.

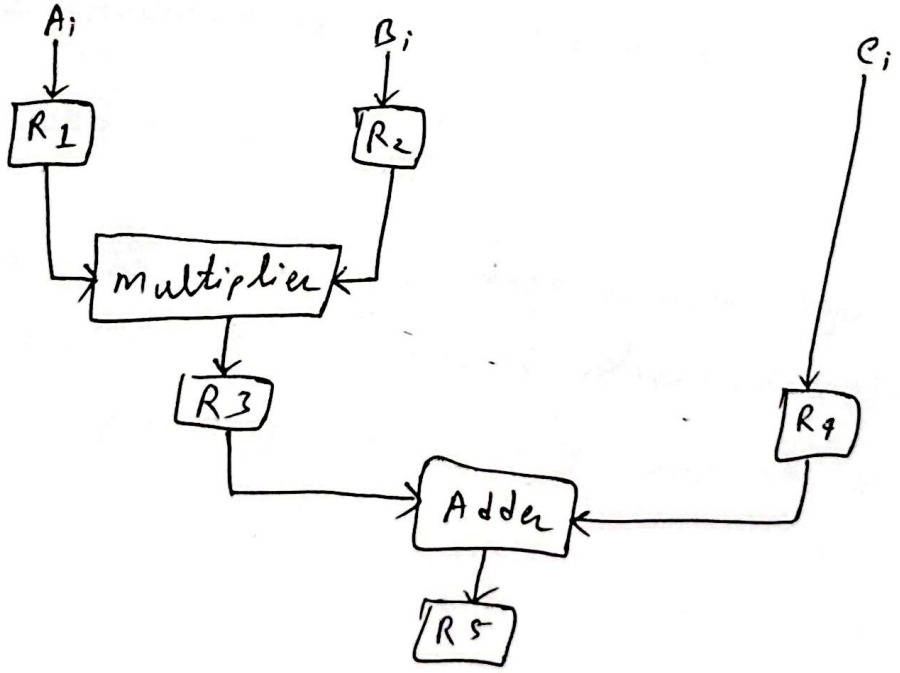

Figure : Example of Pipeline processing

Space-Time Diagram:

The space-time diagram represents the operation of a pipeline over time. The horizontal axis shows clock cycle, and the vertical axis represents different pipeline stages. Initially, the pipeline requires a few clock cycles to fill. After that, one output is produced in each clock cycle. The overlapping of operations increases throughput significantly.

Performance of Pipeline:

If a pipeline has k stages and n tasks are to be processed, the total time without pipelining is $n \times k$ ~~cycles~~ clock cycles. With pipelining, the total time becomes $k+(n-1)$ clock cycles.

The speedup of a pipeline is give by:

$$\text{speedup} = \frac{n \times k}{k+n-1}$$

When the number of task is very large, the speedup approaches the number of stages (k). This shows that pipelining can greatly improve performance.

4. Data Hazards and its Handling Methods:

Data hazards occur when an instruction depends on the result of a previous instruction, and that result of the instruction has not yet been computed.
whenever two different instructions use the same storage. The location must appear as if it is executed in sequential order.

There are four types of data dependencies:

1. Read after write (RAW)
2. Write after Read (WAR)
3. Write after write (WAW)
4. Read after Read (RAR)

These are explained below.

Read after Write (RAW):

It is also known as True dependency or Flow dependency. It occurs when the value produced by an instruction is required by a subsequent instruction.
For example,

```
ADD  R1, --, --;
SUB  --, R1, --;
```

stalls are required to handle these hazards.

Write after Read (WAR):

It is also known as anti dependency. These hazards occur when the output register of an instruction is used right after read by previous instruction. For example,

```
ADD --, R1, --;
SUB R1, --, --;
```

Write after Write (WAW):

It is also known as output dependency. These hazards occur when the output register of an instruction is used for write after written by previous instruction. For example,

```
ADD R1, --, --;
SUB R1, --, --;
```

Read after Read (RAR):

It occurs when the instruction both read from the same register. For example,

```
ADD --, R1, --;
SUB --, R1, --;
```

Since reading a register value does not change the register value, these Read after Read (RAR) hazards don't cause a problem for the processor.

Data hazards occur when instructions in a pipeline depend on the results of previous instructions. To ensure smooth execution, various hazard-handling techniques like forwarding and stalling are used. provides an in-depth exploration of these techniques, ensuring you're well-prepared for competitive exams and real-world applications.

Handling Data Hazards:

These are various methods we use to handle ~~haz~~ hazards:

1. Forwarding:

It adds special circuitry to the pipeline. This method works because it takes less time for the required values to travel through a wire than it does for a ~~t~~ pipeline segment to compute its ~~resul~~ results.

2. Code reordering:

We need a special type of software to reorder code. We call this type of software a ~~hardware-depard~~ hardware-dependent compiler.

3. Stall Insertion:

It inserts one or more stall (no-op instructions) into pipeline, which delays the execution of the current ~~inst~~ instruction untill the required operand is written to the register file, but this method decreases pipeline efficiency and throughput.