|  |
| --- |
| Experiment No. 10 |
| Implement a program to demonstrate the concept of pipeline |
| Date of Performance: |
| Date of Correction: |

**Aim:** To implement a program that demonstrates the concept of instruction pipelining in computer architecture.

**Objective:** To understand and simulate the concept of pipelining used in modern processors to improve instruction throughput and overall CPU performance.

**Theory:**

**Introduction:**

In traditional (non-pipelined) processors, each instruction is processed one at a time—fetch, decode, execute, and write-back. This results in significant idle time for various parts of the CPU during each cycle.

Pipelining is a technique that allows overlapping of these stages, like an assembly line, so multiple instructions are in different phases of execution simultaneously. This improves the CPU's throughput (number of instructions completed per unit time) without increasing clock speed.

**Pipelining Stages:**

**A basic instruction pipeline has the following stages:**

|  |  |
| --- | --- |
| **Stage** | **Description** |
| **IF (Fetch)** | Instruction is fetched from memory |
| **ID (Decode)** | Instruction is decoded, operands identified |
| **EX (Execute)** | Operation is performed |
| **MEM** | Memory is accessed (if needed) |
| **WB (Write Back)** | Result is written to register/memory |

In pipelining, while one instruction is in the "Execute" stage, the next may already be in the "Decode" stage, and a third one in the "Fetch" stage.

**Visual Example (3-Stage Pipeline):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Cycle** | **Instruction 1** | **Instruction 2** | **Instruction 3** |
| **1** | **Fetch** |  |  |
| **2** | **Decode** | **Fetch** |  |
| **3** | **Execute** | **Decode** | **Fetch** |
| **4** |  | **Execute** | **Decode** |
| **5** |  |  | **Execute** |

This overlapping results in faster instruction execution overall.

**Types of Pipelining:**

1. **Instruction Pipelining –** Overlaps fetch-decode-execute stages.
2. **Arithmetic Pipelining –** Used in floating-point units to process parts of arithmetic operations.
3. **Processor-Level Pipelining –** Multiple cores with pipelined execution for parallelism.

**Challenges in Pipelining:**

* Data Hazards: When an instruction depends on the result of a previous one.
* Control Hazards: Occur due to branch or jump instructions.
* Structural Hazards: When hardware resources are insufficient to execute all stages concurrently.

**These can be handled using:**

* Stalling
* Forwarding (Bypassing)
* Branch Prediction

**Real-World Importance:**

* All modern processors (Intel, AMD, ARM) use pipelining to enhance performance.
* Forms the basis of superscalar and parallel processing architectures.
* Essential concept in compiler optimization, instruction scheduling, and microarchitecture design.

**Solution:**

**Conclusion:** We simulated instruction pipelining and observed how overlapping stages can improve instruction throughput and processor performance.