DAY 13 -111 DAYS VERIFICATION CHALLENGE

Topic: Processor pipelining

Skill: Computer Architecture

DAY 13 CHALLENGE:

1. Why do we need processor pipelining?

- **Increase Speed**: It allows multiple instructions to be processed simultaneously, boosting overall speed.
- **Improve Efficiency**: Keeps different parts of the CPU busy, reducing idle time.
- **Higher Clock Speed**: Breaking tasks into smaller steps lets the CPU run faster.
- **Better Performance**: More instructions are completed in less time, enhancing performance.

2. What will happen if we don't use any processor pipelining at all?

□ **Reduced Instruction Throughput**:

- The CPU will process only one instruction at a time from start to finish, significantly lowering the number of instructions executed per unit of time.

□ **Lower CPU Utilization**:

- Many parts of the CPU will be idle at any given time, waiting for the current instruction to complete before starting the next one, leading to inefficient use of resources.

□ **Longer Instruction Latency**:

- Each instruction will take the full length of the instruction cycle (fetch, decode, execute, memory access, write back) to complete before the next one begins, increasing the time it takes to execute a program.

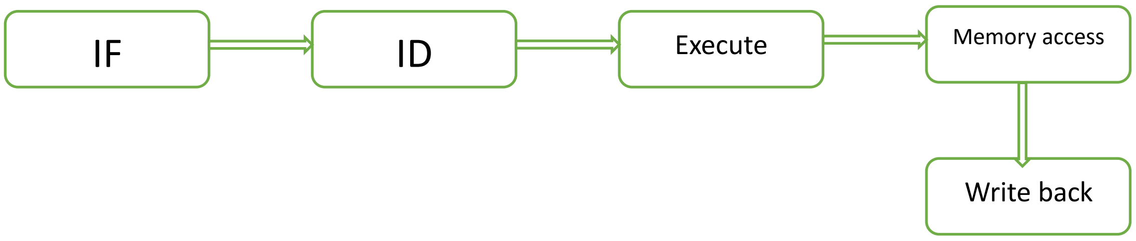
□ **Lower Clock Speeds**:

- Without pipelining, the CPU can't be optimized to run at higher clock speeds since the critical path (longest time-consuming path through the CPU) isn't minimized.

 Poor Scalability:

- Scaling performance improvements would be more difficult without pipelining, as each performance increase would require more significant changes to the CPU architecture.

3. Draw 5 stage pipeline & explain operation of each stage.



IF : instruction fetch

ID : instruction Decode

- **Fetch**:
  - **Action**: Retrieve the next instruction from memory.
  - **Purpose**: Get the instruction to be executed next.
- **Decode**:
  - **Action**: Interpret the fetched instruction to understand what action is required.
  - **Purpose**: Prepare for executing the instruction by identifying the operation and the operands.
- **Execute**:
  - **Action**: Perform the operation specified by the instruction (e.g., arithmetic or logic operation).
  - **Purpose**: Carry out the instruction's action.
- **Memory Access**:
  - **Action**: Read data from or write data to memory if the instruction involves a memory operation.

    - **Purpose**: Handle data transfer between CPU and memory.

- **Write Back**:

    - **Action**: Write the result of the executed instruction back to the appropriate register or memory location.
    - **Purpose**: Update the CPU's state with the results of the instruction.

4. List the advantages & disadvantages of processor pipelining.

Advantages of Processor Pipelining

1. **Increased Throughput**: More instructions are completed in less time.
2. **Improved Efficiency**: Better utilization of CPU resources.
3. **Higher Clock Speeds**: Each stage can operate faster, allowing a higher overall clock speed.
4. **Parallelism**: Multiple instructions are processed simultaneously at different stages.
5. **Scalability**: Easier to enhance performance by adding more pipeline stages.

Disadvantages of Processor Pipelining

1. **Complexity**: More complex CPU design and control logic.
2. **Hazards**: Issues like data hazards, control hazards, and structural hazards can occur, requiring additional mechanisms to handle.
3. **Stall Cycles**: Pipeline stalls or delays can reduce efficiency, such as when waiting for data to be available.
4. **Resource Conflicts**: Increased demand for CPU resources can lead to conflicts.
5. **Increased Power Consumption**: More active components can lead to higher power usage.

5. Consider a pipeline having 5 phases with duration as below:

Fetch - 60 ns

Decode - 50 ns

Execute - 90 ns

Memory Access - 100 ns

Write back - 150 ns.

Given latch delay is 25 ns.

Calculate: -

I. **Pipeline cycle time (Tip: Pipeline Cycle Time = Max of time taken by each stage + Latch delay)**
   = 150 + 25
   = 175 ns
II. **Non-pipeline execution time (Tip: Non-pipeline execution time = Sum of time taken by each stage)**
   = 60 + 50 + 90 + 100 + 150
   = 450 ns