Pipelining

Pipelining: It is an effective way of organizing concurrent activity in a computer system.

Types of Pipelining

There are various types of pipelining such as Arithmetic pipelining, Instruction Pipelining, Processor Pipelining. Out of which Instruction pipelining is discussed as follows.

2. Instruction Pipelining

The number of instruction are pipelined and the execution of current instruction is overlapped by the execution of the subsequent instruction. It is also called **instruction lookahead**.

There are various types of Pipelining

- 2- Stage Pipelining (Already explained given notes in class)
- 4-Stage Pipelining
- 5-Stage Pipelining
- 6-Stage Pipelining

Four Stage Pipelining

: A 4-stage pipeline is a concept often used in computer architecture to describe the organization of processing units in a CPU (Central Processing Unit) to improve its performance and efficiency.

Each stage represents a specific task or operation that an instruction goes through as it is processed by the CPU.

The purpose of using a pipeline is to allow multiple instructions to be in various stages of execution simultaneously, which can significantly increase the overall throughput of the CPU.

In a 4-stage pipeline, there are four primary stages through which instructions progress:

1. Instruction Fetch (IF):

- In this stage, the CPU fetches the next instruction from memory. The program counter (PC) is used to determine the memory address of the next instruction to be executed.

- The fetched instruction is then placed in an instruction buffer or queue to be processed in subsequent stages.

2. Instruction Decode (ID):

- In this stage, the CPU decodes the fetched instruction. It determines the operation to be performed and identifies the operands involved.

- The CPU also checks for any hazards or data dependencies between the current instruction and previously fetched instructions.

3. Execute (EX):

- In the execution stage, the CPU performs the actual operation specified by the instruction. This stage can involve arithmetic or logic operations, memory access, and other computations.

- If the instruction involves data processing, this is where the computation takes place. For memory access instructions, it could involve fetching or storing data in memory.

4. Write Back (WB):

- In the final stage, the results of the executed instruction are written back to the appropriate registers or memory locations. This stage completes the instruction and prepares the CPU for the next instruction in the pipeline.

- For instructions that produce results, this stage ensures that the results are properly stored, making them available for future instructions.

Advantages of a 4-stage pipeline

- Pipelining organizes the execution of the multiple instructions simultaneously.
- Pipelining improves the throughput of the system.
- Pipelining increase the overall performance of computer system.

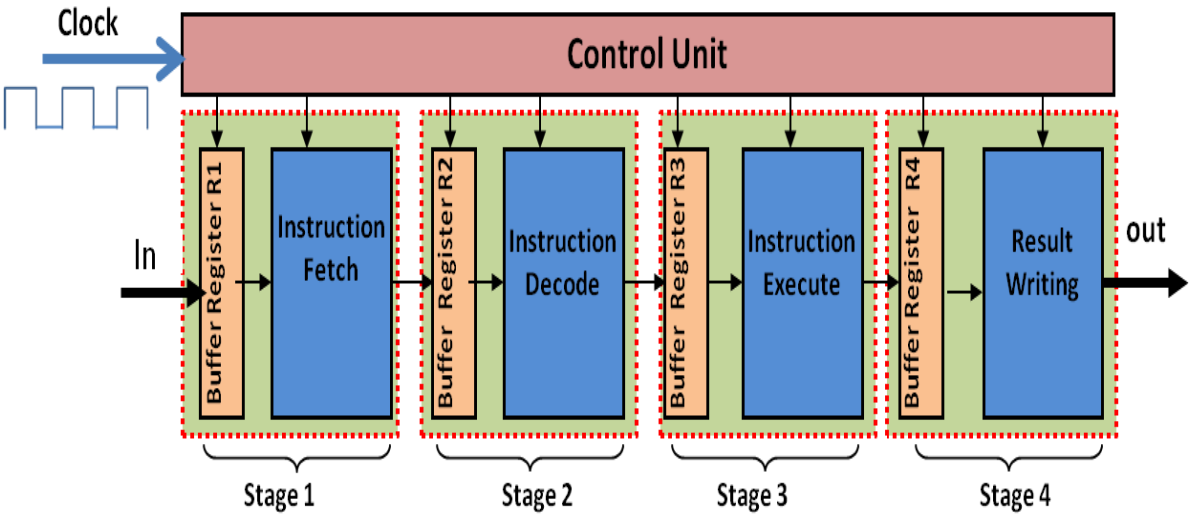


Fig1: Four stage pipeline

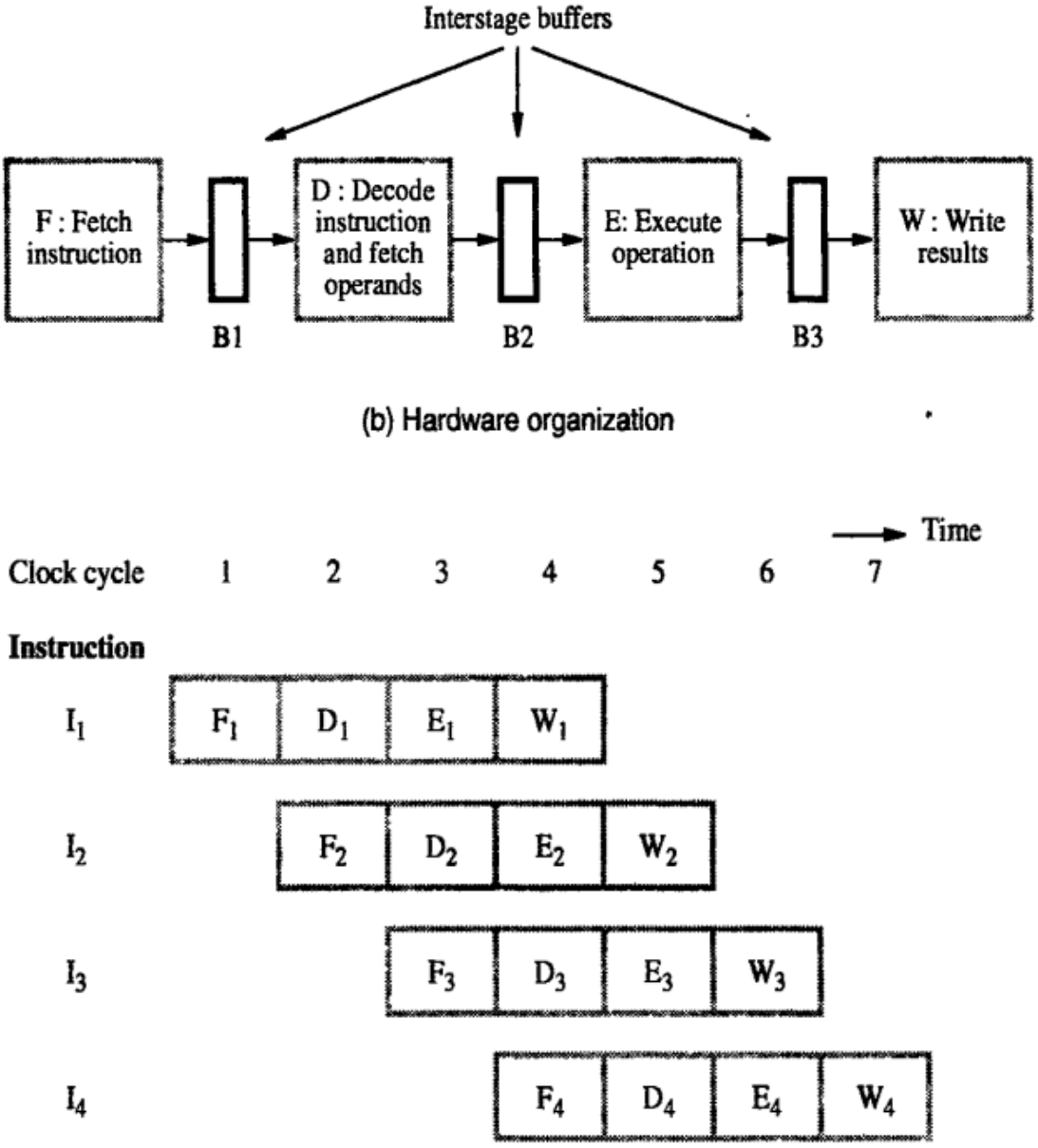
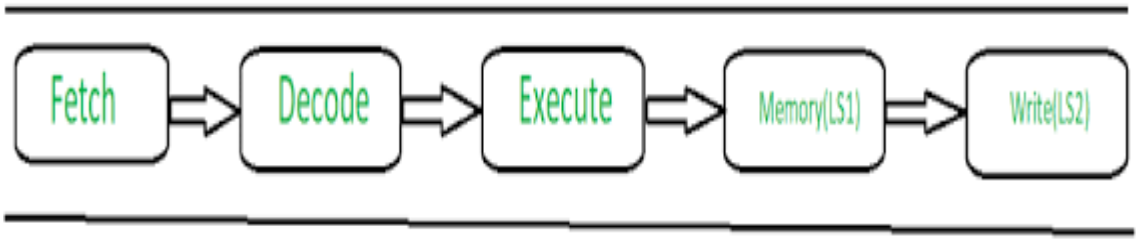


Fig.2: Execution of 4 instructions in pipeline

Fig. 2 shows how the four instructions are executing in pipeline structure.When instruction I1 is in Decode unit, Instruction I2 will be in fetch unit. The assumption is that each instruction in each stage will take only one clock cycle. At the clock 4, All instructions are in some of the unit. By using the concept of pipelining ,these 4 instruction will be executed in 7 clock cycles , otherwise It would take 16 clock cycles in sequential execution.

Five Stage Pipelining



Pipeline Stages RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of the RISC pipeline with their respective operations:

- **Stage 1 (Instruction Fetch)** In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.
- **Stage 2 (Instruction Decode)** In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.
- **Stage 3 (Instruction Execute)** In this stage, ALU operations are performed.
- **Stage 4 (Memory Access)** In this stage, memory operands are read and written from/to the memory that is present in the instruction.
- **Stage 5 (Write Back)** In this stage, computed/fetched value is written back to the register present in the instructions.

Instr No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Six Stage Pipelining

1. Fetch instruction (FI): Read the next expected instruction into a buffer.

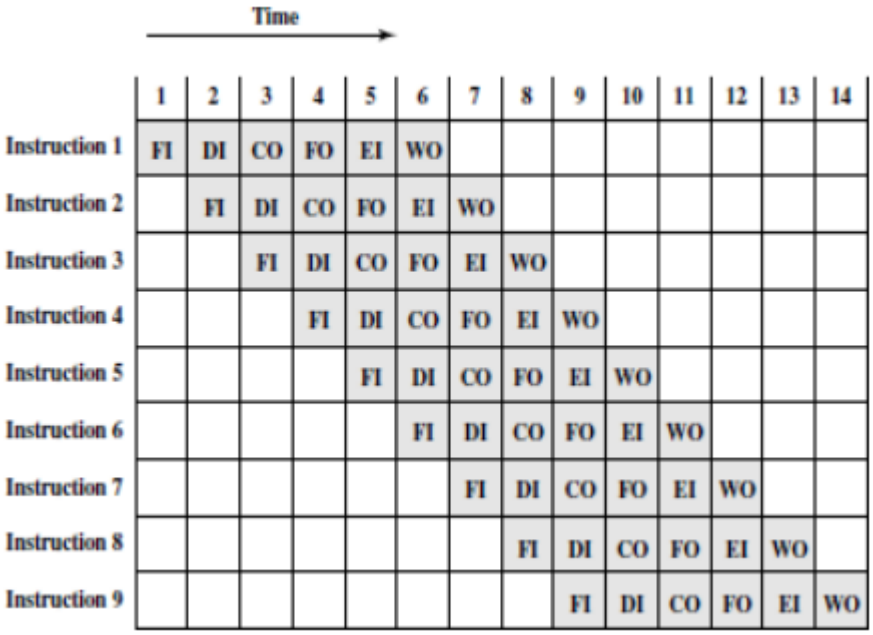
2. Decode instruction (DI): Determine the opcode and the operand specifiers.

3. Calculate operands (CO): Calculate the effective address of each source operand. This may involve displacement, register indirect, indirect, or other forms of address calculation.

4. Fetch operands (FO): Fetch each operand from memory.

5. Execute instruction (EI): Perform the indicated operation and store the result, if any, in the specified destination operand location.

6. Write operand (WO): Store the result in memory.



Pipelining Hazards

Definition: A pipeline has to stall(stop) due to some reason it is called pipeline hazards. In other words, pipeline hazards are situations that can occur in a CPU's pipeline architecture, leading to delays in the execution of instructions or potential conflicts between instructions.

. There are three primary types of pipeline hazards:

1. Data Hazards:

- Data Hazard is any condition in which either the source or destination operands are not available at the expected time in the pipeline.
- Data hazards, also known as data dependencies, occur when an instruction depends on the result of a previous instruction that has not yet completed.

-There are three subtypes of data hazards:

a. Read-After-Write (RAW) Hazard: This happens when an instruction reads data that a previous instruction is still writing to.

b. Write-After-Read (WAR) Hazard: This occurs when an instruction writes data that a subsequent instruction reads before the write operation is complete.

c. Write-After-Write (WAW) Hazard:In this case, two instructions attempt to write to the same register or memory location in close succession.

- Data hazards can be resolved using techniques like forwarding (also known as data forwarding or bypassing) to provide the dependent instruction with the required data without waiting for it to be written back to a register. If forwarding is not possible, a pipeline stall or a bubble (inserting a no-op instruction) may be used to resolve the hazard

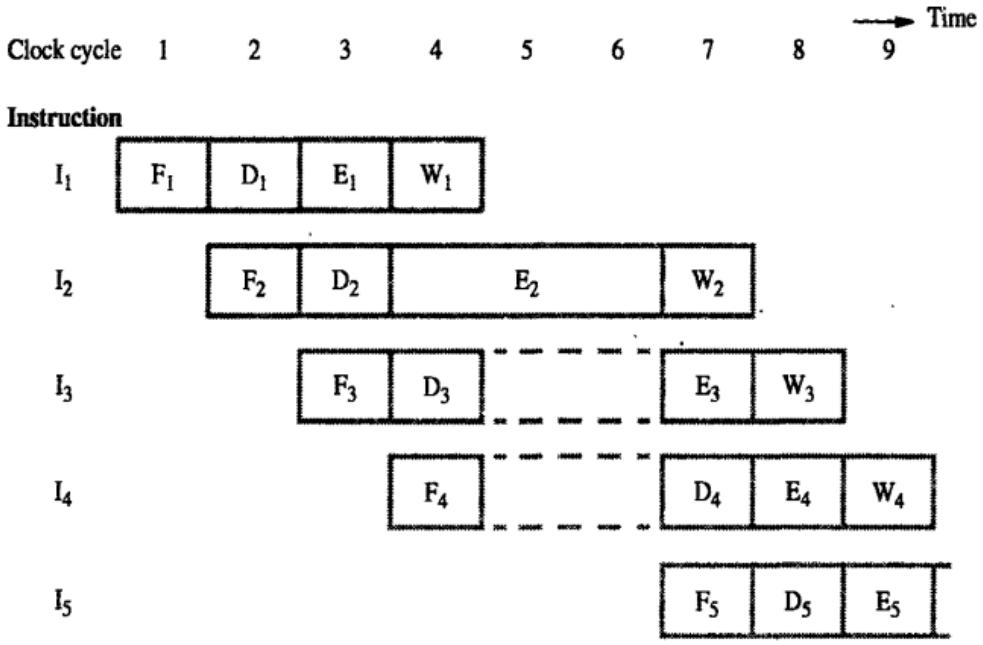


Fig 4 : Example of data hazard

As shown in fig4, Instruction I2 takes more clock cycles for execution than expected and hence the pipeline get stalled as rest of the instructions are progressing further.

2. Instruction Hazard

The pipeline may also be stalled due the delay in availability of instruction, this may be result due to the miss in cache, requiring the instruction to be fetched from the main memory,such hazards are called as Instruction hazard or control hazard.

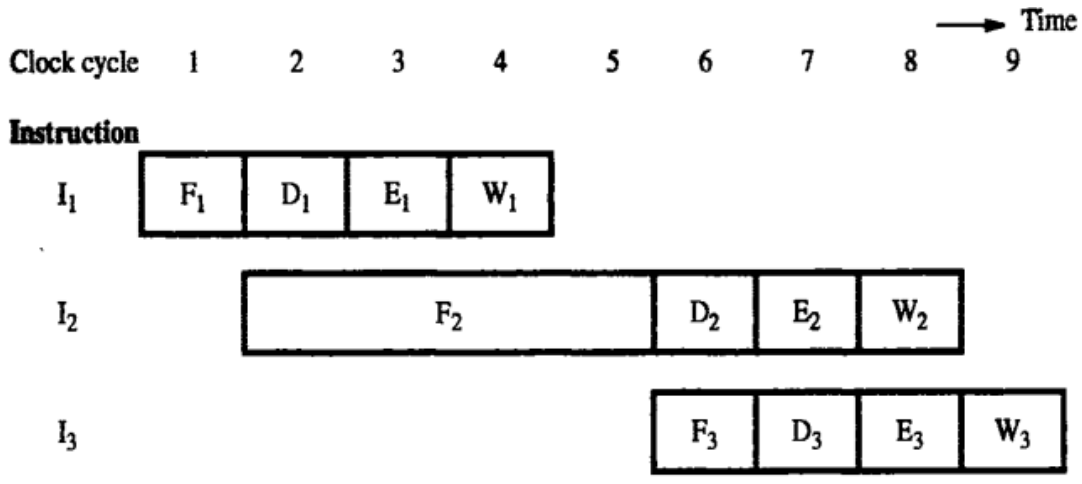


Fig.5 : Instruction Hazard

The above fig. 5 shows that instruction I2 takes more time to get read as it was unavailable due to some cache miss, due to this pipeline get stalled.

3. Structural Hazards:

- Structural hazards occur when multiple instructions in the pipeline need access to the same hardware resource simultaneously, but the resource can only be used by one instruction at a time.

- Common examples include conflicts over access to the same register file, memory unit, or ALU (Arithmetic Logic Unit).

- These conflicts can cause pipeline stalls, where one or more instructions must wait until the resource becomes available, slowing down the entire pipeline.

4. Control Hazards:

- Control hazards, also known as control dependencies or branch hazards, occur when there's a conditional branch instruction (e.g., a branch or jump) and the CPU doesn't yet know whether to take the branch or continue with the next sequential instruction.

- This can cause issues because the pipeline has already fetched and started processing instructions following the branch instruction. If the branch is taken, the instructions fetched after the branch should be discarded, which results in wasted work and a pipeline flush.

- Various techniques are used to handle control hazards, including branch prediction, speculative execution, and out-of-order execution. These methods aim to minimize the impact of control hazards by making educated guesses about the outcome of branch instructions.

Handling pipeline hazards is crucial in modern CPU design to ensure that the pipeline operates efficiently and that instructions are processed in the correct order. Advanced pipeline architectures use various mechanisms to detect and mitigate these hazards, allowing for higher performance and throughput while maintaining program correctness.