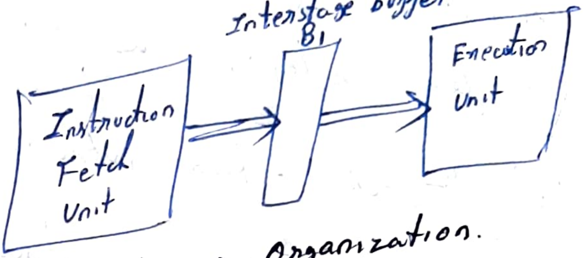
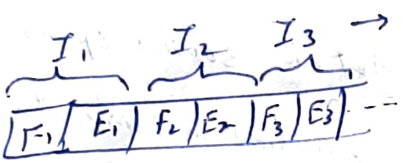
PIPELINING-

Pipeling is a Particularly effective way of organizing Concurrent activity in a Computer system.

In this, the execution of Next instruction is started before compleating the execution of current instruction. (ie) the Next instruction will be fetched while executing the current instruction.



a) Hardware Organization.



Sequential execution.

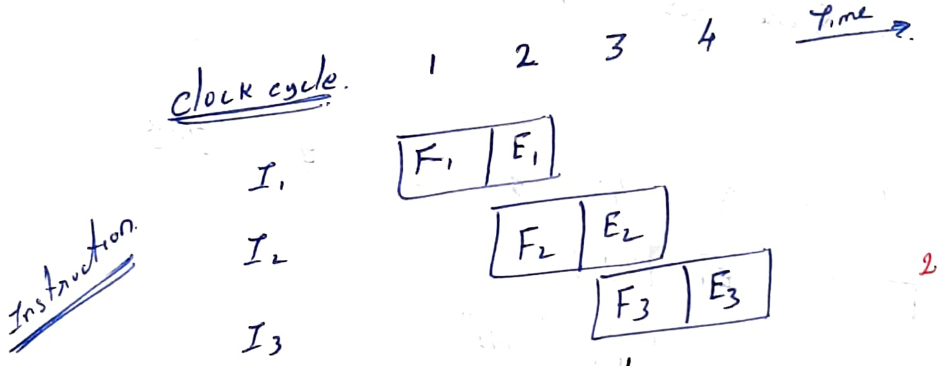


Fig 8.1

2 stage Pipelining.

b) Pipelined Execution.

It has 2 h/w (hardware) unit for Fetching & executing instructions seperately.

Thus the Fetched instruction(I1) is stored in Interstage buffer. & it will be used for execution. so, the Next instruction(I2) can be fetched during its execution.

In this manner, both the fetch & execute units are kept busy all the time. The completion rate of instruction execution will be twice that of Sequential operation.

* Interstage storage Buffer (B1) ⇒ is needed to hold the information being passed from one stage to the next.

* New information is loaded into this buffer at the end of each clock cycle.

A Pipelined Processor may process each instruction in four steps, as Follows.

(F) Fetch - Read instruction from memory.

(D) Decode - Decode instruction & fetch source operand(s).

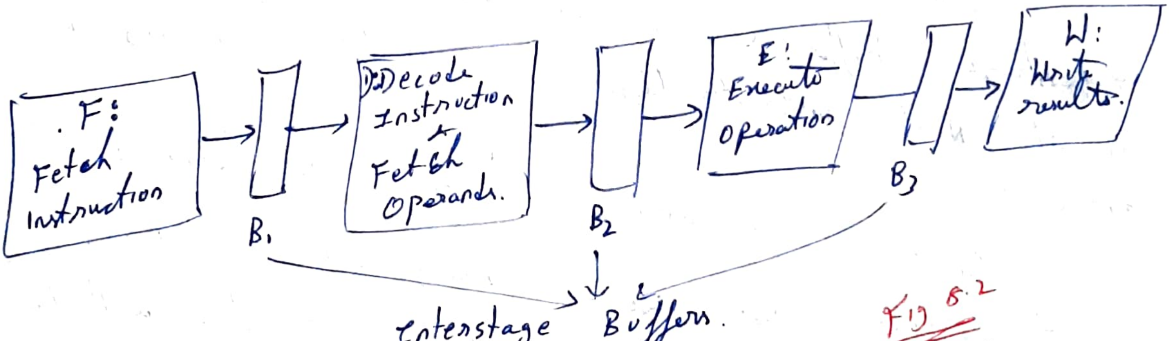
(E) Execute - Perform the operation specified in instruction.

(W) Write - Store result in the destination location.

Clockcycle Instruction	1	2	3	4	5	6	7
I_1	F_1	D_1	E_1	W_1			
I_2		F_2	D_2	E_2	W_2		
I_3			F_3	D_3	E_3	W_3	
I_4				F_4	D_4	E_3	W_4

Time →

a) Instruction Execution.



b) Hardware Organization.

Fig 8.2
4 stage Pipeline.

Role of cache memory -

Each stage in a pipeline is expected to complete its operation in one clock cycle. Hence, the clock period should be sufficiently long to complete the task being performed in any stage. If different units require different amounts of time, the clock period must allow the longest task to be completed. A unit that completes its task early is idle for the remainder of the clock period.

Hence, Pipelining is most effective in improving performance if the tasks being performed in different stages require about the same amount of time.

During fetch cycle, the access time for main memory will be greater compared to execution of other steps inside the processor. So, a cache is included to balance the fetch & processing steps to be more or less equal in duration.

Pipeline Performance -

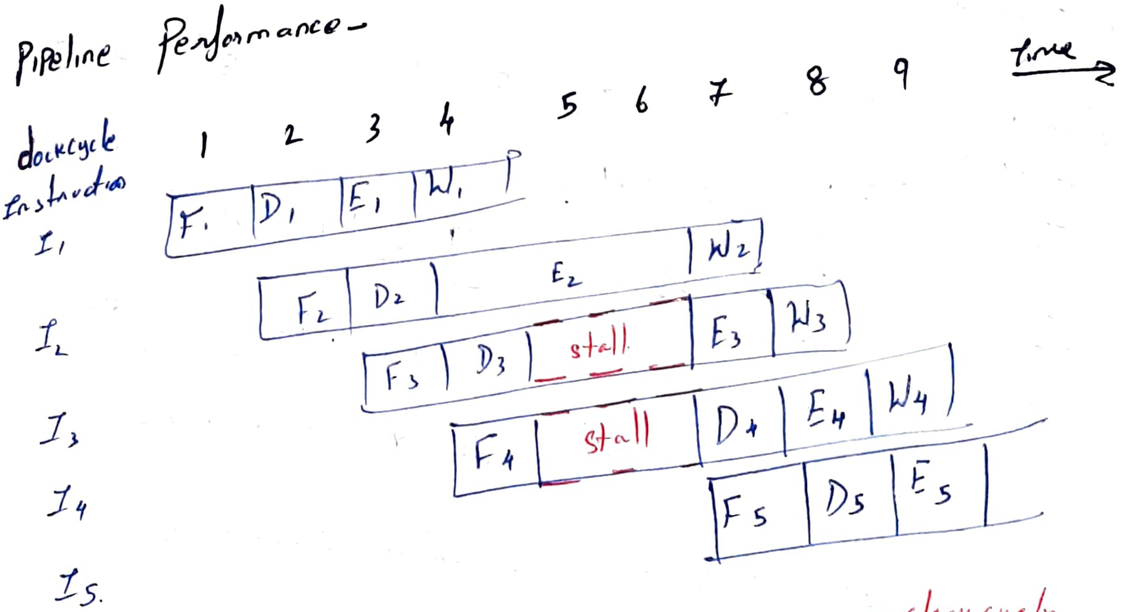


Fig 8.3 execution operation taking more than one clock cycle.

In the Fig 8.3 the pipelined operation is said to have been stalled for two clock cycles.

Any Condition that causes the pipeline to stall is called as hazard.

1) Data Hazard -

It is the condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result Some operation has to be delayed and the pipeline stalls.

2) Instruction Hazards / Control Hazards -

The pipeline may stall because of a delay in availability of an instruction. this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. such hazards are often called control hazards or instruction hazards.

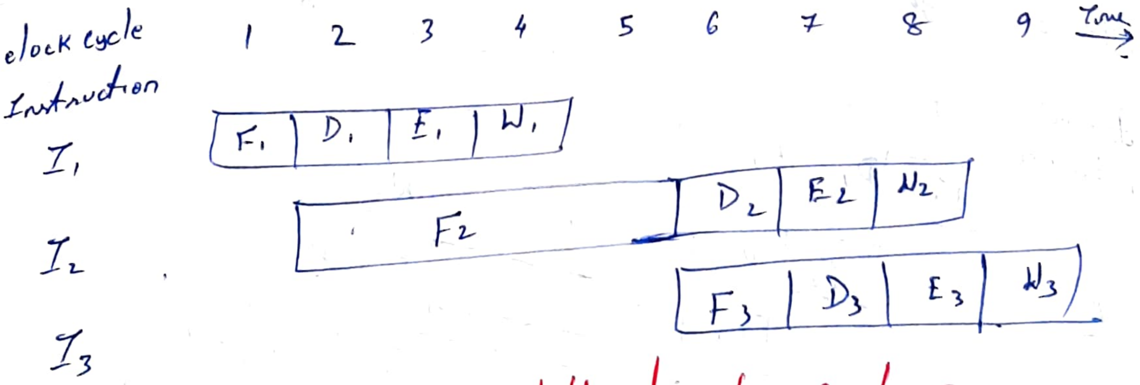


Fig 8.4 Pipeline stall due to cache miss.

a) Instruction Execution clock cycles.

clock cycle. Stage.	1	2	3	4	5	6	7	8	9
F: Fetch.	F1	F2	F2	F2	F2	F3			
D: Decode		D1	idle	idle	idle	D2	D3		
E: Execute			E1	idle	idle	idle	E2	E3	
W: Write				W1	idle	idle	idle	W2	W3.

Time →

b) Function performed by each processor in successive clock cycles.

The idle periods of the processor units are called **stalls**. These are often referred to as **bubbles** in the pipeline.

Once created bubble as a result of delay in one of the pipeline stages, moves downstream until it reaches the last unit.

3) **Structural Hazard** -

This is the situation when 2 instructions require to use the given hardware resource at the same time. The most common case for this hazard may arise in access to memory.

Many processor use seperate instruction and data caches to avoid this delay.

Load x (R1), R2.

eg

clock cycle. Instruction.	1	2	3	4	5	6	7
I1	F1	D1	E1	W1			
I2 (Load)		F2	D2	E2	M2	W2	
I3			F3	D3	E3	- -	W3
I4				F4	D4	- -	E4 - - -
I5						F5	D5 - - -

Time →