"PIPELINING"

Pipelining is a phenomena or method using which, we will be able to run more than one instruction at the same time, on singe processor.

Eik instruction execute hone mein 5 clock cycles (ya 5 secs) leta hai to agr do execute krni ho to 10 lega. To isi cheez ko reduce krne k liye pipelining ka concept use hota hai.

Time taken for n instruction to execute without pipeline = total no. of phases $*$ n $\times$ time of one clock.

with pipleline = [total no. of phases + (n-1)] $\times$ time of one clock

How much the system is speedup:
Speedup = (Time without pipline) / Time with piplene

HAZARDS

A hazard (conflict) is created whenever there is a dependence b/w instr., & instr. are close enough that overlap caused by pipelining would change the order of acess to the operands involved in the dependence.

Types of Hazards
1. Structural Hazards
2. Data Hazards
3. Control Hazards

STRUCTURAL HAZARDS :

→ Attempt to use same resource (memory) from diff instr. simultaneously.

→ Agr do instr. same memory use krrhi hai to ek instruction ko rukna porega q k ek waqt mein ek hi instr. memory use kr sakti hai.

Q: Why in MIPS Architecture there are no structural Hazards?

In MIPS architecture, structural hazards

In MIPS architecture, structural hazards are minimized because the design ensures that diff parts of the processor don't need to use the same hardware at the same time. Instr. are avoided by ensuring that instruction in different stages of execution dont conflict with eachother for same resources.

(2) DATA HAZARDS :

Attempt to use a result before it's ready.
eg: Instr. depending on a result of a previous instr. still in pipeline

- Types of Structural Hazards :

→ RAW :

Read After Write (Flow / True dependency).
RAW hazard occurs if an instruction I_2 tries to read on R_2 before instruction I_1 writes it.

I_1 : $R_2 \leftarrow R_1 + R_3$

I_2 : $R_4 \leftarrow R_2 + R_3$

I_2 is trying to read R_2 before it has been written in memory by I_1.

- **WAR :**

Write After Read (Anti Data Dependency)
This occurs when instr. I_2 tries to write data before instruct I_1 read it.

I_1 : $R_2 \leftarrow R_1 + R_3$

I_2 : $R_3 \leftarrow R_4 + R_5$ → Agr ye phele execute hogai to R_2 mein ghalat value dali jayegi.

- **WAW :**

Write After Write (Output Data Dependency).
When I_2 tries to write output before I_1 write it.

I_1 : $R_2 \leftarrow R_1 + R_3$

I_2 : $R_2 \leftarrow R_4 + R_5$

Possible Solutions :

① Compilation Technique :

a) Insertion of "nop" (no operation instr.)

b) Instruct. sheduling to avoid that the corellating instruction are too close. ↙

Is mein hum un instructions ko phele likhdenge jinka Instruction I" se koi relation nhi hoga.

CamScanner

```
sub $2, $1, $3  →
nop
nop
nop
and $12, $2, $3
 or $13, $6, $2
add $14, $2, $2
sw  $15, 100($2)
```

CamScanner