

00:00

Structural Hazards in Pipelining

00:30

Structural Hazards in Pipelining

→ Let's take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB)

01:01

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁							
I ₂							
I ₃							
I ₄							



01:30

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁	IF	ID	EX				
I ₂		IF	ID				
I ₃							
I ₄							



02:00

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁	IF	ID	EX	WB			
I ₂		IF	ID	EX	WB		
I ₃			IF	ID			
I ₄							



02:10

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁	IF	ID	EX	WB			
I ₂		IF	ID	EX	WB		
I ₃			IF	ID	EX		
I ₄				IF	ID		



03:03

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁	IF	ID	EX	WB ← Mem			
I ₂		IF	ID	EX	WB		
I ₃			IF	ID	EX	WB	
I ₄				IF	ID	EX	WB

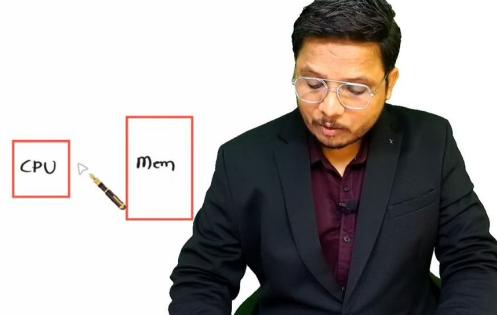


03:33

→ Lets take four stage pipeline :

1. Instruction Fetch (IF)
2. Instruction decode + Operand decode (ID)
3. Execute Instruction (EX)
4. Writeback (WB).

	1	2	3	4	5	6	7
I ₁	IF	ID	EX	WB ← Mem			
I ₂		IF	ID	EX	WB		
I ₃			IF	ID	EX	WB	
I ₄				IF	ID	EX	WB



structural hazard basically means that we can't use the same resource agar ek fetch mei memory use kr rhi hai and ek write

back mei

05:14

→ Solⁿ - Harvard Architecture.

	1	2	3	4	5	6	7
Mem Inst.	I ₁	IF	ID	EX	wB	Mem (data)	
	I ₂	IF	ID	EX	wB		
	I ₃	IF	ID	EX	wB		
	I ₄	IF	ID	EX	wB	Mem (code)	

Diagram illustrating the Harvard Architecture timing. The timeline shows four instructions (I₁, I₂, I₃, I₄) being processed. Instruction I₁ is in the EX stage, reading memory for data. Instruction I₄ is in the IF stage, reading memory for code. A separate diagram shows the CPU connected to two memory units: one for code and one for data.

07:39

	1	2	3	4	5	6	7	8
(MUL) I ₁	IF	ID	EX	EX	EX	wB		
(ADD) I ₂	IF	ID	EX	EX	wB			

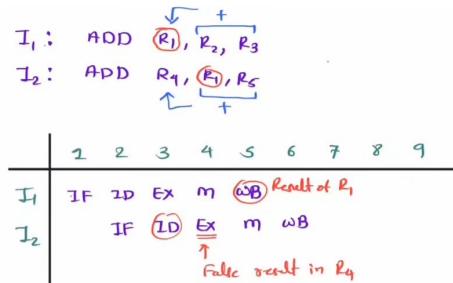
Diagram illustrating pipeline stalls. Instruction I₁ (MUL) is in the EX stage, while instruction I₂ (ADD) is stalled in the ID stage due to a stall. An annotation indicates "2 clock cycles are stalled".

multiplication takes more time for execution as compared to addition so addition wali instruction have to wait for the multiplication to free up execution unit to proceed hence adding a stall

Data Hazard

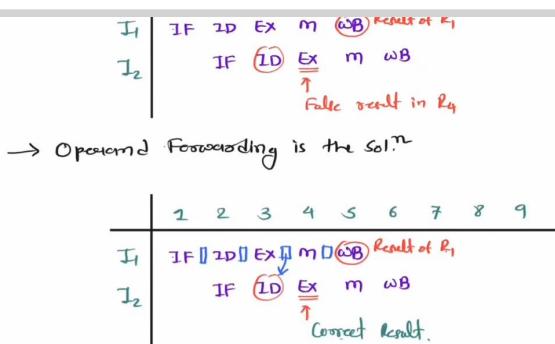
03:31

Data Hazards in Pipelining



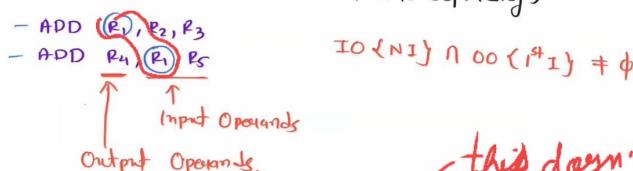
I2 decode stage will take the not updated value of R1 and create false result

06:23

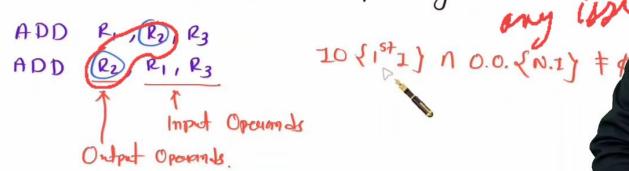


basically buffer register is used to fetch the value in operator forwarding instead of the actual memory location

→ RAW - Read After Write (Actual / True dependency)



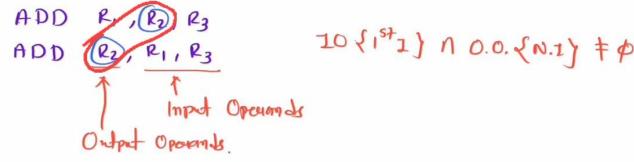
→ WAR - Write After Read (Anti dependency).



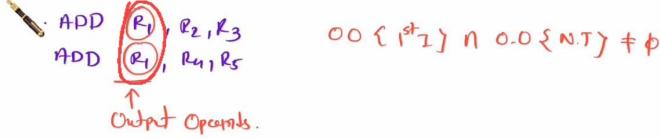
this doesn't cause any issue



→ WAW - write after write (Anti dependency).



→ WAW - write After write (Output dependency)



02:06

Control Hazards in Pipelining

→ Control Hazards in pipeline is happening due to branch Inst."

→ Lets have four stage pipeline: $\frac{\text{IF}}{\uparrow}$, $\frac{\text{ID}}{\uparrow}$, $\frac{\text{EX}}{\uparrow}$, $\frac{\text{WB}}{\uparrow}$
 1st stage 2nd stage 3rd stage 4th stage .

Address	Inst ⁿ
1000	I ₁
1004	I ₂
1008	I ₃
1012	I ₄
⋮	⋮



04:40

→ Lets have four stage pipeline: $\frac{\text{IF}}{\uparrow}$, $\frac{\text{ID}}{\uparrow}$, $\frac{\text{EX}}{\uparrow}$, $\frac{\text{WB}}{\uparrow}$
 1st stage 2nd stage 3rd stage 4th stage .

Address	Inst ⁿ
1000	I ₁
1004	I ₂
1008	I ₃
1012	I ₄
⋮	⋮
2000	I _n

1	2	3	4	5	6	7
IF	1D	EX	WB			
	IF	1D	EX	WB		
X	IF	1D	EX	WB		
			IF	1D	EX	WB
						Flush pipeline.



during branch instruction we get to know where to jump (that is the address of jump location) during execution stage

07:40

→ During branch no cycles stall = EX Stage - 1

03:5407:52

- **GATE CS** – Consider a 5 stage pipeline. Delay of each stage is given as per 10ns, 16ns, 12ns, 11ns & 14ns, respectively. Calculate the execution time of 100 instructions and Speed up due to pipeline.

$$\begin{aligned} \rightarrow K &= 5 \\ t_c &= 16 \text{ nsec} \\ n &= 100 \\ \rightarrow T_p &= (n+K-1)t_c = (100+5-1)16 = 1664 \text{ nsec} = 1.664 \text{ usec} \\ \rightarrow S &= \frac{T_{nop}}{T_p} = \frac{100(10+16+12+11+14)}{1664} = 3.786 \end{aligned}$$



Examples on Pipelining in COA

- **GATE CS** – Consider a 5 stage pipeline with cycle time of 5ns. Calculate the execution time of 100 instructions and Speed up due to pipeline. Also find the utilization.

$$\begin{aligned} \rightarrow K &= 5 \\ t_c &= 5 \text{ ns} \\ n &= 100 \\ \rightarrow T_p &= (n+K-1)t_c = (100+5-1)5 = 520 \text{ nsec} \\ \rightarrow S &= \frac{nK}{n+K-1} = \frac{100 \times 5}{100+5-1} = 4.8077 \\ \rightarrow U &= \frac{n}{n+K-1} = \frac{100}{100+5-1} = 0.9615 \end{aligned}$$



- GATE 2011 CS** – Consider an instruction pipeline with four stages (S1, S2, S3 & S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delay for the stages and for the pipeline registers are as given in the figure.



- What is the approximate speed up of the pipeline in steady state under ideal conditions w/ compared to the corresponding non pipeline implementation?

$$S = \frac{T_{nop}}{T_p} = \left\lceil \frac{5 + 6 + 11 + 8}{11 + 1} \right\rceil = 2.5$$



01:53

Examples on Pipelining in COA

- GATE CS** – Consider a pipeline having 5 stages with duration 10ns, 30ns, 45ns, 80ns and 35ns. If the buffer delay is 20ns, calculate the speed up of pipelined processor.

$$S = \frac{T_{nop}}{T_p} = \left\lceil \frac{10 + 30 + 45 + 80 + 35}{80 + 20} \right\rceil$$



05:50

- GATE CS** – Assume we have two pipelines P1 and P2, respectively. P1 has 6 stages, having execution time of 12ns, 14ns, 19ns, 20ns, 22ns and 25ns. P2 has 4 stages each having execution time of 10ns. Calculate the time that can be saved while using P2 pipeline over P1 pipeline, If 2000 instructions are executed.

$$\rightarrow P_1, K_1 = 6, t_{c_1} = 25 \text{ nsec}$$

$$\rightarrow P_2, K_2 = 4, t_{c_2} = 10 \text{ nsec}$$

$$\rightarrow n = 2000$$

$$\rightarrow T_{P_1} = (n + K_1 - 1) t_{c_1} = (2000 + 6 - 1) 25 = 50125 \text{ nsec}$$

$$\rightarrow T_{P_2} = (n + K_2 - 1) t_{c_2} = (2000 + 4 - 1) 10 = 20030 \text{ nsec}$$

$$\rightarrow \Delta T = T_{P_1} - T_{P_2} = 50125 - 20030 = 30095 \text{ nsec}$$



07:53

- ❑ GATE CS – Assume a pipeline P which operates at 3GHz clock rate. It has a speed up factor of 10 and efficiency of 40%. Calculate the number of stages in the above pipeline.

$$\rightarrow f_{clock} = 3 \text{ GHz}$$

$$S = 10$$

$$\eta = 40\% = 0.4$$

$$\rightarrow S = K\eta \Rightarrow K = \frac{10}{0.4} = 25$$



03:21

Examples on Pipelining in COA

- ❑ GATE 2015 CS – Consider a non pipelined processor with a clock rate of 2.5GHz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to internal pipeline delay, the clock speed is reduced to 2GHz. Assume that there are no stalls in the pipeline. The speed up achieved in this pipelined processor is

$$\rightarrow f_{c1} = 2.5 \text{ GHz}, CPI = 4$$

$$\rightarrow K = 5, f_{c2} = 2 \text{ GHz}$$

$$\rightarrow S = \frac{T_{avg}}{T_p} = \frac{4/2.5 \times 10^9}{1/2 \times 10^9} = \frac{8}{2.5} = \underline{\underline{3.2}}$$



10:14

- ❑ GATE CS – The stage delays in a 4 stage pipeline are 5ns, 6ns, 4ns and 5ns. The second stage is replaced with a functionally equivalent design involving two stages with respective delays 4ns and 5ns. The throughput increase of the pipeline is %

$$\rightarrow \underline{\underline{P-1}} \{ 5\text{ns}, \underline{6\text{ns}}, 4\text{ns}, 5\text{ns} \} \quad \rightarrow \underline{\underline{P-2}} \{ \underline{5\text{ns}}, 4\text{ns}, 5\text{ns}, 4\text{ns}, 5\text{ns} \}.$$

$$\rightarrow \text{Throughput} = \frac{n}{(n+k-1)t_c} = \frac{1}{t_c} = \underline{\underline{n \text{ ideally}}}.$$

$$\rightarrow t_{c1} = 6\text{ns}, t_{c2} = 5\text{ns}$$

$$\rightarrow TP_1 = \frac{1}{t_{c1}} = \frac{1}{6\text{ns}}$$

$$\rightarrow TP_2 = \frac{1}{t_{c2}} = \frac{1}{5\text{ns}}.$$

$$\rightarrow \% \uparrow_a = \frac{TP_2 - TP_1}{TP_1} \times 100$$

$$\approx \frac{1/5 - 1/6}{1/6} \times 100$$

$$\approx \frac{6}{5} - 1 = 0.2 \times 100 = 20\%.$$



12:26

- GATE 2004 CS** – A 4 stage pipeline has the stage delays as 150, 120, 160 and 140 nanoseconds, respectively. Registers that are used between the stages have a delay of 5 ns each. Assuming constant clock rate, the total time taken to process 1000 data items on this pipeline will be Microseconds

$$n = 1000$$

$$K = 4$$

$$t_c = 160 + 5 = 165 \text{ nsec}$$

$$\begin{aligned} \rightarrow T_p &= (n+K-1) t_c \\ &= (1000+4-1) 165 \\ &= 165495 \text{ nsec} \\ &= 165.495 \text{ microsec} \end{aligned}$$

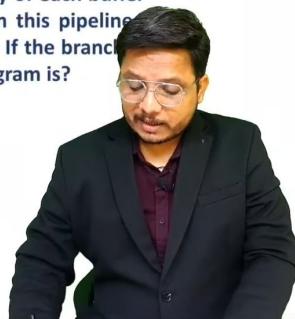


05:53

Examples on Pipelining in COA

- GATE 2009 CS** – Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delay for FI, DI, FO, EI and WO are 5ns, 7ns, 10ns, 8ns and 6ns, respectively. There are intermediate storage buffer after each stage and the delay of each buffer is 1ns. A program consisting of 12 instructions I1, I2, I3, ..., I12 is executed in this pipeline processor. Instruction I4 is only the branch instruction and its branch target is I9. If the branch is taken during the execution of this program, the time needed to complete the program is?

$$\begin{aligned} - &\frac{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8}{n = 8} & - T_p = t_p + t_s \\ - & \text{FI, DI, FO, EI, WO} & = (n+K-1) t_c + x t_c \\ & \uparrow & = (8+5-1) 11 + 3 \times 11 \\ & \text{4th Stage.} & = \\ - & \text{No of stalled cycles} = 4-1 = 3 \end{aligned}$$



10:01

- GATE CS** – Consider the pipelined processor with the following four stages:

IF: Instruction Fetch

ID: Instruction Decode and Operand Fetch

EX: Execute

WB: Write Back

- The IF, ID and WB stages take one clock cycle each to complete the operation. The number of clock cycles for the EX stage depends on the instruction. The number ADD and SUB instructions need 1 clock cycle and MUL instruction needs 3 clock cycle in the EX stage. Operand forwarding is used in the pipelined processor. What is the number of clock cycles taken to complete the following sequence of instructions? = 8

ADD R2, R1, R0	;R2 \leftarrow R1 + R0
MUL R4, R3, R2	;R4 \leftarrow R3 \times R2
SUB R6, R5, R4	;R6 \leftarrow R5 + R4

I₁- IF ID EX WB
I₂- IF ID EX EX EX WB
I₃- IF ID — EX WB
2 cycles stalled.



11:18

GATE CS – Consider the pipelined processor with the following four stages:

- IF: Instruction Fetch
- ID: Instruction Decode and Operand Fetch
- EX: Execute
- WB: Write Back

The IF, ID and WB stages take one clock cycle each to complete the operation. The number of clock cycles for the EX stage depends on the instruction. The number ADD and SUB instructions need 1 clock cycle and MUL instruction needs 3 clock cycle in the EX stage. Operand forwarding is used in the pipelined processor. What is the number of clock cycles taken to complete the following sequence of instructions? = 8

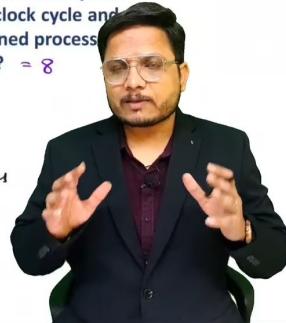
```

ADD R2, R1, R0      ;R2 ← R1 + R0
MUL R4, R3, R2      ;R4 ← R3 × R2
SUB R6, R5, R4      ;R6 ← R5 + R4

```

I_1 : IF ID EX WB
 I_2 : IF ID EX EX EX WB
 I_3 : IF ID — — EX WB
 2 cycles stalled.

$$\begin{aligned}
 \text{cycles} &= \text{Pipe Cycles} + \text{Hazards Cycles} \\
 &= (n+k-1) + 1 \times 2 \\
 &= 3+4-1 + 2 = 8
 \end{aligned}$$



04:34

Examples on Pipelining in COA

GATE 2017 CS – Instruction execution in a processor is divided into 5 stages. Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Execute (EX), and Write Back (WB). These stages take 5, 4, 20, 10 and 3 nanoseconds (ns), respectively. A pipelined implementation of the processor requires buffering between each pair of consecutive stages with a delay of 2 ns. Two pipelined implementations of the processor are contemplated:

- (i) a naive pipeline implementation (NP) with 5 stages and
- (ii) an efficient pipeline (EP) where the OF stage is divided into stages OF1 and OF2 with execution times of 12 ns and 8 ns respectively.

The speedup (correct to two decimal places) achieved by EP over NP in executing 20 independent instructions with hazards is _____.

✓ (A) 1.50-1.51	NP: { 5, 4, 20, 10, 3 }	EP: { 5, 4, 12, 8, 10, 3 }
(B) 1.51-1.52		
(C) 1.52-1.53	$k_1 = 5$	$k_2 = 6$
(D) 1.53-1.54	$t_{c1} = 20 + 2 = 22 \text{ ns}$	$t_{c2} = 12 + 2 = 14 \text{ ns}$

$$\Rightarrow S = \frac{T_{NP}}{T_{EP}} = \frac{(n+k-1)t_{c1}}{(n+k-1)t_{c2}} = \frac{(20+5-1)22}{(20+5-1)14} = 1.508$$

2 Example of Pipelining



09:38

GATE 2010 CS – A 5 stage pipelined processor has instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and write back (WO) stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 cycle for ADD and SUB instructions, 3 clock cycles for MUL instruction and 6 clock cycles for DIV instruction respectively. Operand forwarding is used in the pipeline. What is the number of cycles needed to execute the following sequence of instruction? (15)

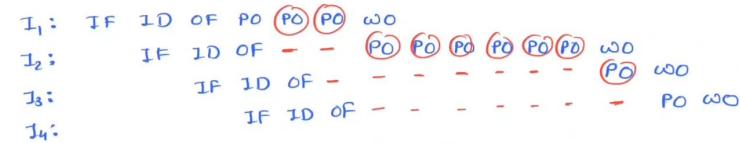
I_1 : ③ MUL R2, R0, R1 ;R2 ← R0 × R1
 I_2 : ⑥ DIV R5, R3, R4 ;R5 ← R3 / R4
 I_3 : ① ADD R2, R5, R2 ;R2 ← R5 + R2
 I_4 : ⑤ SUB R5, R2, R6 ;R5 ← R2 - R6

I_1 : IF ID OF PO PO WO
 I_2 : IF ID OF — — PO PO PO PO PO WO
 I_3 : IF ID OF — — — — PO WO
 I_4 : IF ID OF — — — — — — PO WO



11:12

Pipeline: What is the number of cycles needed to execute the following sequence of instructions?		
J1	<u>③</u> MUL R2, R0, R1	;R2 \leftarrow R0 x R1
J2	<u>⑥</u> DIV R5, R3, R4	;R5 \leftarrow R3 / R4
J3	<u>①</u> ADD R2, R5, R2	;R2 \leftarrow R5 + R2
J4	<u>①</u> SUB R5, R2, R6	;R5 \leftarrow R2 - R6



$$\begin{aligned}
 \rightarrow \text{Cycles} &= \text{Normal Cycles} + \text{Horseshoe Cycles} \\
 &= (n+k-1) + (5+2) \\
 &= (4+5-1) + 5+2 \\
 &= (15)
 \end{aligned}$$



16:35

What is the number of cycles needed to execute the following sequence of instruction without data forwarding in above question?

I_1	MUL R2, R0, R1	(3)	$R2 \leftarrow R0 \times R1$
I_2	DIV R5, R3, R4	(6)	$R5 \leftarrow R3 / R4$
I_3	ADD R2, R5, R2	(1)	$R2 \leftarrow R5 + R2$
I_4	SUB R5, R2, R6	(1)	$R5 \leftarrow R2 - R6$

Output Input
 Operands Operands

I_1 :	IF	ID	OF	PO	PO	PO	WB
I_2 :	IF	ID	OF	-	-	PO	WB
I_3 :	IF	ID	-	-	-	-	WB
I_4 :	IF	ID	-	-	-	-	WB

03:38

Examples on Pipelining Hazards in COA

GATE 2018 CS – The Instruction pipeline of a RISC processor has the following stages: Instruction (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Writeback (WB). The IF, ID, OF and WB stages take 1 clock cycle each for every instruction. Consider a sequence of 100 instructions. In the PO stage, 40 instructions take 3 clock cycles each, 35 instructions take 2 clock cycles each and the remaining 25 instructions take 1 clock cycle each. Assume that there are no data hazards and no control hazards. The number of clock cycles required for completion of execution of the sequence of instruction is

$$\rightarrow \text{No of cycles} = \text{Normal pipeline cycles} + \text{cycles due to structural hazard}$$

$$= (n+k-1) + \frac{35 \times 1}{ } + \frac{40 \times 2}{ }$$

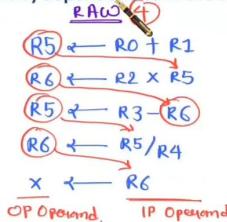


10:29

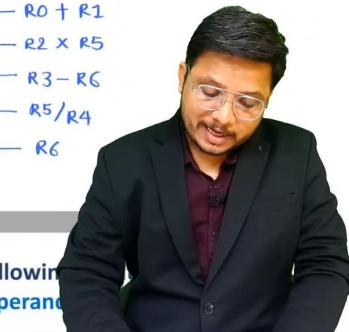
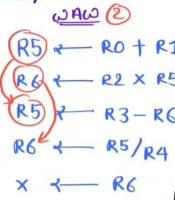
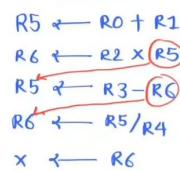
JSD R3, R2, R0 ;R3 ← R2 - R0
 DIV R6, R5, R4 ;R6 ← R5 / R4
 STORE R6, X ;X ← R6

- The number of Read After Write (RAW) dependencies, Write After Read (WAR) dependencies and Write After Write (WAW) dependencies in the sequence of instruction are, respectively.

- A. 2, 2, 4
 B. 3, 2, 3
 C. 4, 2, 2
 D. 3, 3, 2



WAR (2)



- GATE 2003 CS – For a pipelined CPU with a single ALU, consider the following:

- The $j + 1^{\text{st}}$ instruction uses the result of the j^{th} instruction as an operand
- The execution of a conditional jump instruction

12:08

- GATE 2003 CS – For a pipelined CPU with a single ALU, consider the following situations

- The $j + 1^{\text{st}}$ instruction uses the result of the j^{th} instruction as an operand
- The execution of a conditional jump instruction
- The j^{th} and $j + 1^{\text{st}}$ instruction require the ALU at the same time

- Which of the above can cause a hazard

- A. I and II Only
 B. II and III Only
 C. III Only
 D. All of above

(1) → Data Hazard.
 (2) → Control Hazard.
 (3) → Structural Hazard.

**08:30**

Non Synchronized Pipelining in COA

GATE CS – A four stage pipeline is given below:

	S1	S2	S3	S4
I1	2	1	1	3
I2	1	3	2	1
I3	1	1	3	1
I4	2	1	1	1

What is the performance gain achieved by the above pipeline structure over non pipeline execution

- A. 1.84 B. 1.72 C. 1.92 D. 2.08

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I1	S1	S1	S2	S3	S4	S4	S4							
I2		S1	S2	S2	S2	S3	S3	S4						
I3			S1	—	—	S2	—	S3	S3	S3	S4			
I4				S1	S1	—	S2	—	—	—	S3	S4		



09:14

Non Synchronized Pipelining in COA

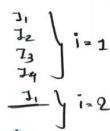
GATE 2004 CS – Consider a pipeline processor with 4 stages S1 to S4. We want to execute the following loop:

For ($i=1, i \leq 1000, i++$)

(I1, I2, I3, I4)

Where the time taken (in ns) by instruction I1 to I4 for stages S1 to S4 are given below:

	S1	S2	S3	S4
I1	1	2	1	2
I2	2	1	2	1
I3	1	1	2	1
I4	2	1	2	1



The Output of I1 for $i = 2$ will be available after

A. 11ns B. 12ns C. 13ns D. 28ns

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I1	s1	s2	s3	s4	s5									
I2		s1	s2	s3	s4									
I3			s1	s2	–	s3	s3	s4						
I4				s1	s1	s2	–	s3	s3	s4				
I1					s1	s2	s2	–	s3	s4	(s4)			

04:22

CPI & Speed Up in Pipelining with Hazard

→ Without Hazard.

$$CPI = \frac{\text{Total Cycles}}{\text{Total Inst.} n} = \frac{(n+k-1)}{n} = \textcircled{1} \text{ ideal.}$$

→ With Hazard { x cycles due to hazard }.

$$CPI = \frac{(n+k-1) + x}{n} = \left[1 + \frac{x}{n} \right] \text{ ideal.}$$

05:57

CPI & Speed Up in Pipelining with Hazard

→ Without Hazard.

$$CPI = \frac{\text{Total Cycles}}{\text{Total Inst.} n} = \frac{(n+k-1)}{n} = \textcircled{1} \text{ ideal.}$$

→ With Hazard { x cycles due to hazard }.

$$CPI = \frac{(n+k-1) + x}{n} = \left[1 + \frac{x}{n} \right] \text{ ideal.}$$

→ without Hazard.

$$S = \frac{T_{op}}{T_p} = \frac{T_{op}}{CPI T_p}$$



09:29

- GATE CS** – Consider a 5 stage pipeline which is executing a program of 100 instructions. Among all instructions 10 instructions cause 3 stall cycles each.

- A. Calculate CPI of Pipeline?
- B. If Pipeline cycle time is 3ns then what is average instruction execution time?
- C. Calculate CPI of pipeline in ideal conditions with hazards?
- D. If pipeline cycle time is 3ns then what is average instruction execution time in ideal conditions?

$$\rightarrow k = 5 \\ n = 100 \\ x = 10 \times 3 = 30$$

$$1) CPI = \frac{(n+k-1)+x}{n} = \frac{100+5-1+30}{100} = 1.34$$

$$2) t_{inst} = CPI \times t_c = 1.34 \times 3 = 4.02 \text{ nsec}$$

$$3) CPI_{wst} = 1 + \frac{x}{n} = 1 + \frac{30}{100} = 1.3$$

**10:07**

- GATE CS** – Consider a 5 stage pipeline which is executing a program of 100 instructions. Among all instructions 10 instructions cause 3 stall cycles each.

- A. Calculate CPI of Pipeline?
- B. If Pipeline cycle time is 3ns then what is average instruction execution time?
- C. Calculate CPI of pipeline in ideal conditions with hazards?
- D. If pipeline cycle time is 3ns then what is average instruction execution time in ideal conditions?

$$\rightarrow k = 5 \\ n = 100 \\ x = 10 \times 3 = 30$$

$$1) CPI = \frac{(n+k-1)+x}{n} = \frac{100+5-1+30}{100} = 1.34$$

$$2) t_{inst} = CPI \times t_c = 1.34 \times 3 = 4.02 \text{ nsec}$$

$$3) CPI_{wst} = 1 + \frac{x}{n} = 1 + \frac{30}{100} = 1.3$$

$$4) t_{inst(wst)} = CPI_{wst} t_c = 1.3 \times 3 = 3.9 \text{ nsec.}$$

**03:13**

Examples on Pipelining Hazards in COA

- GATE 2014 CS** – Consider a 6 stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle time overhead of pipelining. When an application is executing on this 6 stage pipeline, the speedup achieved with respect to non pipelined execution if 25% of the instructions incur 2 pipeline stall cycles is?

$$\begin{aligned} \rightarrow k &= 6 \\ \rightarrow CPI &= 1 + \frac{x}{n} \\ &= 1 + 0.25 \times 2 \\ &= 1.5 \\ \rightarrow S &= \frac{T_{nop}}{CPI T_p} = \frac{6 t_c}{1.5 t_c} = 4 \end{aligned}$$



09:29

GATE CS – Consider a 3 stage pipeline having delays of 100ns, 200ns and 500ns, respectively. The third stage is spliced into two stages of 250ns and 250ns respectively. Calculate the throughput increase or decrease in percentage.

- A. 100% increase
- B. 60% increase
- C. 60% decrease
- D. 50% decrease

$$\underline{P-1} \quad \{ 100\text{ns}, 200\text{ns}, 500\text{ns} \}$$

$$k_1 = 3$$

$$t_{c1} = 500\text{ns}$$

$$\underline{P-2} \quad \{ 100\text{ns}, 200\text{ns}, \underline{250\text{ns}}, \underline{250\text{ns}} \}$$

$$k_2 = 4$$

$$t_{c2} = 250\text{ns}$$

$$\rightarrow TP = \frac{n}{(n+k-1)t_c} = \frac{1}{t_c}$$

$$\rightarrow TP1 = \frac{1}{t_{c1}} = \frac{1}{500\text{ns}} \quad \rightarrow TP2 = \frac{1}{t_{c2}} = \frac{1}{250\text{ns}}$$

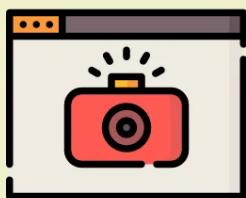
$$\rightarrow \% = \frac{\frac{1}{250} - \frac{1}{500}}{\frac{1}{500}} = \frac{2-1}{1} \times 100 = 100\%$$



No Ads

Remove Ads from pdf and websites

Pricing



Now you can use Askify in any websites

See How