1.18.1 Pipelining: GATE CSE 1999 | Question: 13 top

https://gateoverflow.in/1512

An instruction pipeline consists of 4 stages – Fetch (F), Decode field (D), Execute (E) and Result Write (W). The 5 instructions in a certain instruction sequence need these stages for the different number of clock cycles as shown by the table below

Instruction	F	D	E	W
1	1	2	1	1
2	1	2	2	1
3	2	1	3	2
4	1	3	2	1
5	1	2	1	2

unless & until mentioned assume a simple PL with one unit each IF, ID, IE, IW units & when a unit is processing a instruction at cycle i then next instruction which needs same unit must enter it only after current instruction exists from it.

Find the number of clock cycles needed to perform the 5 instructions. = 15

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F	1	2	3	3	4	5									
D		1	1	2	2	3	4	4	4	5	5				
E				1		2	2	3	3	3	4	4	5		
W					1						3	3	4	5	5

1.18.2 Pipelining: GATE CSE 2000 | Question: 1.8 top

https://gateoverflow.in/631

Comparing the time T1 taken for a single instruction on a pipelined CPU with time T2 taken on a non-pipelined but identical CPU, we can say that

A. T1 ≤ T2
B. T1 ≥ T2
C. T1 < T2
D. T1 and T2 plus the time taken for one instruction fetch cycle

↳ for this PL doesn't improve time taken but extra hardware used in PL like latches/inter stage buffer, lock units, forwarding units, etc consume some extra time

so $T_1 = T_{PL}$ = Time for 1 instruction + small time due to extra h/w
$= T + \Delta$

T_2 = Time for 1 instruction $= T$

if h/w very fast $\Delta \simeq 0 \Rightarrow T_1 \simeq T_2$ } $T_1 \geq T_2$

else $T_1 > T_2$

1.18.3 Pipelining: GATE CSE 2000 | Question: 12 top

https://gateoverflow.in/683

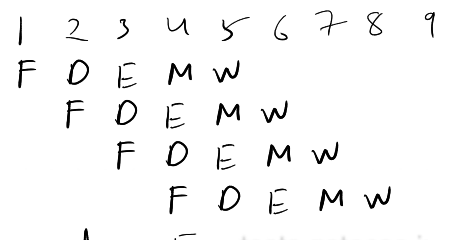
An instruction pipeline has five stages where each stage take 2 nanoseconds and all instruction use all five stages. Branch instructions are not overlapped. i.e., the instruction after the branch is not fetched till the branch instruction is completed. Under ideal conditions,

A. Calculate the average instruction execution time assuming that 20% of all instructions executed are branch instruction. Ignore the fact that some branch instructions may be conditional.
B. If a branch instruction is a conditional branch instruction, the branch need not be taken. If the branch is not taken, the following instructions can be overlapped. When 80% of all branch instructions are conditional branch instructions, and 50% of the conditional branch instructions are such that the branch is taken, calculate the average instruction execution time.

Consider a 5−stage pipeline - IF (Instruction Fetch), ID (Instruction Decode and register read), EX (Execute), MEM (memory), and WB (Write Back). All (memory or register) reads take place in the second phase of a clock cycle and all writes occur in the first phase. Consider the execution of the following instruction sequence:

split phase
Write | Read

I1:	sub $r2, r3, r4$	/* $r2 \leftarrow r3 - r4$ */
I2:	sub $r4, r2, r3$	/* $r4 \leftarrow r2 - r3$ */
I3:	sw $r2, 100(r1)$	/* $M[r1 + 100] \leftarrow r2$ */
I4:	sub $r3, r4, r2$	/* $r3 \leftarrow r4 - r2$ */



A. Show all data dependencies between the four instructions.
B. Identify the data hazards.
C. Can all hazards be avoided by forwarding in this case. yes

I2 R @ cc 3, I1 W @ cc 5

/* r2 ← r3 − r4 */ (H)
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

RAW

I3 R @ cc 4, I1 W @ cc 5

/* r2 ← r3 − r4 */ (H)
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

RAW

for this → I1 W, I4 R correct no Hazard

/* r2 ← r3 − r4 */
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

RAW

I2 W @ cc 6, I1 R @ cc 2

/* r2 ← r3 − r4 */
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

WAR

I4 R @ cc 5, I2 W @ cc 6

/* r2 ← r3 − r4 */ (H)
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

RAW

I4 W @ cc 8, I2 R @ cc 3

/* r2 ← r3 − r4 */
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

WAR

I4 W @ cc 8, I1 R @ cc 2

/* r2 ← r3 − r4 */
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */

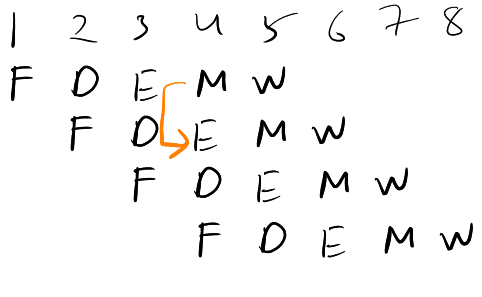
WAR

all instances marked (H) are hazards where Read is happening before Write

I1 produces value of r2 by end of cc 3

I2 R @ cc 3, I1 W @ cc 5

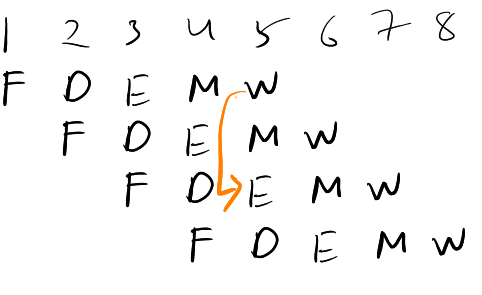
/* r2 ← r3 − r4 */ (H)
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */



I2 needs r2 at start of cc 4 so forwarding possible

I3 R @ cc 4, I1 W @ cc 5

/* r2 ← r3 − r4 */ (H)
/* r4 ← r2 − r3 */
/* M[r1 + 100] ← r2 */
/* r3 ← r4 − r2 */



I1 produces r2 at end of cc 3 & about to write to Reg at cc 5 (start, 1st phase) I3 reads r2 in cc 5 2nd phase, so forward possible

I_1 R @ cc 5, I_2 W @ cc 6

/* $r2 \leftarrow r3 - r4$ */ (H)
/* $r4 \leftarrow r2 - r3$ */
/* $M[r1 + 100] \leftarrow r2$ */
/* $r3 \leftarrow r4 - r2$ */

1	2	3	4	5	6	7	8	9
F	D	E	M	W				
	F	D	E	M	W			
		F	D	E	M	W		
			F	D	E	M	W	

I_2 produces r_4 at cc 4 & writes at cc 6

I_1 need r_4 at cc 6 so forward possible

1.18.5 Pipelining: GATE CSE 2002 | Question: 2.6, ISRO2008-19 top

The performance of a pipelined processor suffers if:

A. the pipeline stages have different delays
B. consecutive instructions are dependent on each other
C. the pipeline stages share hardware resources
~~D.~~ All of the above

A.) out of 5 stages lets say only Ex takes 3 cycles others only take 1 cycle, but to accomodate EX now we have to set clock to take 3 cycles for all if all stages took 1 cycle a single instruction would've finished in 5 cycles, & large no. of instructions would take 1 cycle but due to EX taking 3 cycle, a single inst takes 15 cycles & multiple inst take 3 cycle i.e definitely performance loss compared to ideal case

B.) have to stall till dependent data is ready (time waste)
or
have to forward according (extra checks & data transfer)

C.) i.e structural hazard either have to stall or add extra hardware

all 3 cases effect pipeline performance

1.18.6 Pipelining: GATE CSE 2003 | Question: 10, ISRO-DEC2017-41

For a pipelined CPU with a single ALU, consider the following situations

I. The $j+1^{st}$ instruction uses the result of the j^{th} instruction as an operand — Data Hazard

II. The execution of a conditional jump instruction — control Hazard

III. The j^{th} and $j+1^{st}$ instructions require the ALU at the same time. — structural Hazard

Which of the above can cause a hazard

A. I and II only
B. II and III only
C. III only
D. All the three ✓

1.18.7 Pipelining: GATE CSE 2004 | Question: 69

https://gateoverflow.in/1063

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 *nanoseconds* each. Assuming constant clocking rate, the total time taken to process 1000 data items on this pipeline will be:

A. 120.4 microseconds
B. 160.5 microseconds
C. 165.5 microseconds ✓
D. 590.0 microseconds

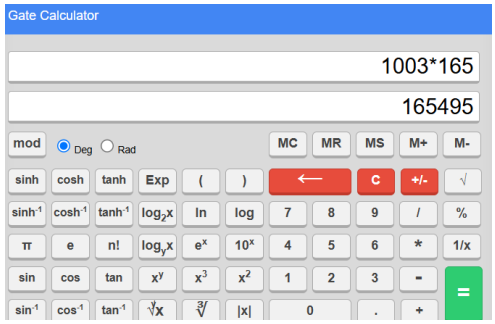
clock cycle = max(stage delays) + inter stage delay

= 160 + 5 = 165 ns

no. of stages = k = 4

no. of stages = n = 1000

total time taken = $(k+n-1) \times t_{cc} = (4+1000-1) \times 165$ ns



$= 1003 \times 165$ ns

$= 165495$ ns

1 microsecond = 1000 nanosecond

1 ns = $\frac{1}{1000}$ μs

$= \frac{165495}{1000}$ μs

$= 165.495$ μs

$\simeq 165.5$ μs

1.18.8 Pipelining: GATE CSE 2005 | Question: 68

A 5 stage pipelined CPU has the following sequence of stages:

- IF – instruction fetch from instruction memory
- RD – Instruction decode and register read
- EX – Execute: ALU operation for data and address computation
- MA – Data memory access – for write access, the register read at RD state is used.
- WB – Register write back

cc	1	2	3	4	5	6	7
IF	1	2	3				
RD		1	2	3			
EX			1	2	3		
MA				1	2	3	
WB					1	2	3

cc 7 not in options & also hazard possible

Consider the following sequence of instructions:

- I_1: L $R0, loc\ 1$; $R0 \Leftarrow M[loc1]$
- I_2: A $R0, R0$; $R0 \Leftarrow R0 + R0$
- I_3: S $R2, R0$; $R2 \Leftarrow R2 - R0$

load-use hazard

all hazards can be resolved using forwarding except load-use to resolve it have to stall which will take 1 cycle extra

7+1 = 8 cycles

Let each stage take one clock cycle.

What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of I_1?

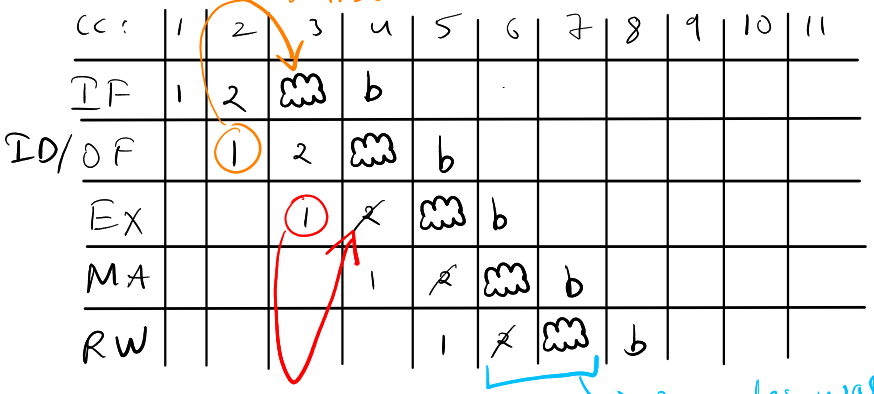
A. 8 ✓
B. 10
C. 12
D. 15

A CPU has a five-stage pipeline and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions out of which 20% are conditional branches. If each instruction takes one cycle to complete on average, the total execution time of the program is:

A. 1.0 second
B. 1.2 seconds
C. 1.4 seconds
D. 1.6 seconds

in 3rd stage or clock cycle 3 i.e in EX we will known whether branch is taken or not

got to know its a branch so insert bubble in next cc



got branch outcome as Taken

so have to stop instructions that already got into pipeline i.e flush them

Recall that only in OF stage opcode gets decoded and we will know whether it is a branch instruction or not

so for each branch inst 2 cycles are wasted / extra

out of 10^9 inst 20% are branches

$10^9 \times 0.2 = 2 \times 10^8$ branch inst

$= 2 \times 10^8 \times 2$ cycles extra

$= 4 \times 10^8$ extra cycles

$= 4 \times 10^8$ cc

$4 \times 10^8 \times \frac{1}{10^9}$ sec

$= \frac{4}{10} = 0.4$ sec extra

The processor runs at a 1 GHz frequency.
This means it can perform 1 billion (10^9) clock cycles per second.

given

1 instruction – 1 cycle

10^9 cc — 1 sec

1 cc — $\frac{1}{10^9}$ sec

1 inst — $\frac{1}{10^9}$ sec

10^9 inst — 1 sec

original + extra = total

1 sec + 0.4 sec = 1.4 sec

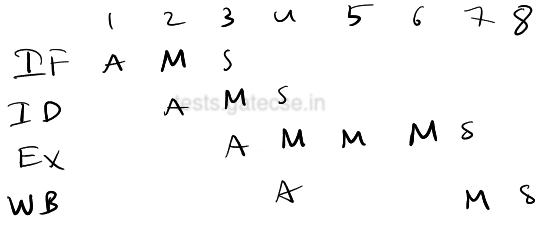
Consider a pipelined processor with the following four stages:

- 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 ADD and SUB instructions need 1 clock cycle and the MUL instruction needs 3 clock cycles 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?

forwarding doesn't help here

ADD	R2, R1, R0	R2 ← R1+R0
MUL	R4, R3, R2	R4 ← R3*R2
SUB	R6, R5, R4	R6 ← R5–R4



A. 7
B. 8
C. 10
D. 14

1.18.11 Pipelining: GATE CSE 2008 | Question: 36 top

Which of the following are NOT true in a pipelined processor?

I. Bypassing can handle all RAW hazards
II. Register renaming can eliminate all register carried WAR hazards
III. Control hazard penalties can be eliminated by dynamic branch prediction

A. I and II only
B. I and III only
C. II and III only
D. I, II and III

II T so II shouldn't be present in correct options

Bypassing = Forwarding
↳ resolves most of RAW hazards except Load-use

control hazard penalties = branch predicted wrongly
penalties can't be eliminated by dynamic branch prediction, as there has to be some penalties for it to learn from

1.18.12 Pipelining: GATE CSE 2008 | Question: 76 top

Delayed branching can help in the handling of control hazards

For all delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false,

A. The instruction following the conditional branch instruction in memory is executed irrespective of branch taken or not taken
B. The first instruction in the fall through path is executed
C. The first instruction in the taken path is executed
D. The branch takes longer to execute than any other instruction

they just asked core idea of delay slot

1.18.13 Pipelining: GATE CSE 2008 | Question: 77 top

Delayed branching can help in the handling of control hazards

The following code is to run on a pipelined processor with one branch delay slot:

I1: ADD $R2 \leftarrow R7 + R8$
I2: Sub $R4 \leftarrow R5 - R6$
I3: ADD $R1 \leftarrow R2 + R3$
I4: STORE Memory $[R4] \leftarrow R1$
BRANCH to Label if $R1 == 0$

Which of the instructions I1, I2, I3 or I4 can legitimately occupy the delay slot without any program modification?

A. I1
B. I2
C. I3
D. I4

```
I2: Sub R4 ← R5-R6
I3: ADD R1 ← R2+ R3
I4: STORE Memory [R4] ← R1
BRANCH to Label if R1 == 0
I1: ADD R2 ← R7+ R8
```

if I1 is moved to delay slot it corrupts results of I3 & I4 since I3 depends on I1

```
I1: ADD R2 ← R7+ R8
I2: Sub R4 ← R5-R6
I4: STORE Memory [R4] ← R1
BRANCH to Label if R1 == 0
I3: ADD R1 ← R2+ R3
```

if I3 is moved some random or garbage value in R1 will be stored instead of R2+R3

```
I1: ADD R2 ← R7+ R8
I3: ADD R1 ← R2+ R3
I4: STORE Memory [R4] ← R1
BRANCH to Label if R1 == 0
I2: Sub R4 ← R5-R6
```

if I2 is moved then in I4 the memory address at which R1 must be stored will be wrong since I2 calculates it

```
I1: ADD R2 ← R7+ R8
I2: Sub R4 ← R5-R6
I3: ADD R1 ← R2+ R3
BRANCH to Label if R1 == 0
I4: STORE Memory [R4] ← R1
```

if I4 is moved to delay slot no effect to intended execution i.e safe instruction

1.18.14 Pipelining: GATE CSE 2009 | Question: 28

https://gateoverflow.in/1314

Consider a 4 stage pipeline processor. The number of cycles needed by the four instructions $I1, I2, I3, I4$ in stages $S1, S2, S3, S4$ is shown below:

	S_1	S_2	S_3	S_4
I1	2	1	1	1
I2	1	3	2	2
I3	2	1	1	3
I4	1	2	2	2

What is the number of cycles needed to execute the following loop?

For $(i = 1 \text{ to } 2)$ {I1; I2; I3; I4;}

A. 16
~~B. 23~~
C. 28
D. 30

loop i=1 starts; loop i=2 starts; loop i=1 finishes; loop i=2 finishes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
S_1	I	I	2	3	3	4	I	I	2	3	3	4												
S_2			I	2	2	2	3	4	4	I	2	2	2	3	4	4								
S_3				I			2	2	3	4	4	I		2	2	3	4	4						
S_4					I				2	2	3	3	3	4	4	I	2	2	3	3	3	4	4	

1.18.15 Pipelining: GATE CSE 2010 | Question: 33

https://gateoverflow.in/2207

A 5−stage pipelined processor has Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Write Operand (WO) stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock 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 clock cycles needed to execute the following sequence of instructions?

Instruction	Meaning of instruction
t_0: MUL R_2, R_0, R_1	$R_2 \leftarrow R_0 * R_1$
t_1: DIV R_5, R_3, R_4	$R_5 \leftarrow R_3 / R_4$
t_2: ADD R_2, R_5, R_2	$R_2 \leftarrow R_5 + R_2$
t_3 : SUB R_5, R_2, R_6	$R_5 \leftarrow R_2 - R_6$

PO:
ADD, SUB 1 cc
MUL 3 cc
DIV 6 cc

3 RAW → only worry about these hazards
1 WAR
2 WAW

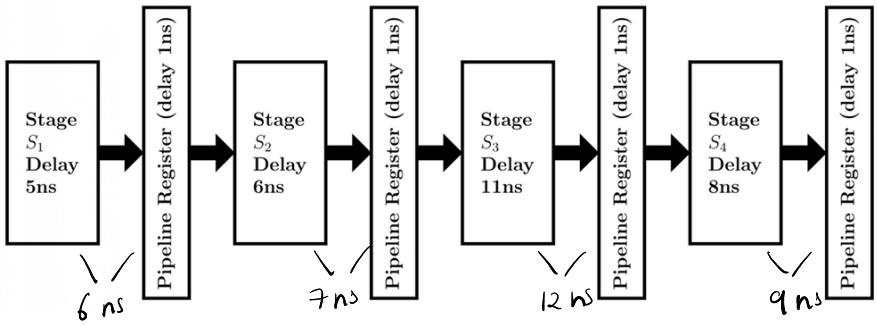
A. 13
~~B. 15~~
C. 17
D. 19

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
IF	M	D	A	S															
ID		M	D	A	S														
OF			M	D	A	A	A	A	A	A	A	A	S						
PO				M	M	M	D	D	D	D	D	D	A	S					
WO							M						D	A	S				

1.18.16 Pipelining: GATE CSE 2011 | Question: 41 top

https://gateoverflow.in/2143

Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays 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 when compared to the corresponding non-pipeline implementation?

A. 4.0
B. 2.5
C. 1.1
D. 3.0

while calculating non-pipeline execution time no need to consider pipeline register delays since those won't be part of Non PL CPU

$T_{non-PL} = 5+6+11+8 = 11+11+8 = 22+8 = 30$

$T_{PL} = \max([5,6,11,8]) + 1 = 11+1 = 12$

$\text{Speedup} = \frac{\text{before}}{\text{after}} = \frac{T_{non-PL}}{T_{PL}} = \frac{30}{12} = 2.5$

1.18.17 Pipelining: GATE CSE 2012 | Question: 20, ISRO2016-23 top

Register renaming is done in pipelined processors:

A. as an alternative to register allocation at compile time
B. for efficient access to function parameters and local variables
C. to handle certain kinds of hazards WAR & WAW in super scalar CPU
D. as part of address translation

1.18.18 Pipelining: GATE CSE 2013 | Question: 45 top

https://gateoverflow.in/330

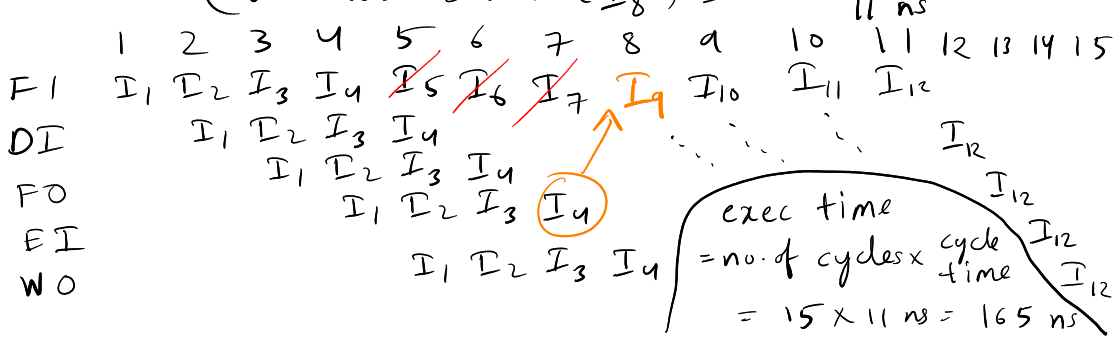
Consider an instruction pipeline with five stages without any branch prediction:

Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 *ns*. A program consisting of 12 instructions I1, I2, I3, . . . , I12 is executed in this pipelined processor. Instruction I4 is the only branch instruction and its branch target is I9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is

A. 132
B. 165
C. 176
D. 328

pipeline cycle time = max([5, 7, 10, 8, 6]) + 1 ns (delay)

(effectively only 1 inst time was saved i.e I_8) = 10 ns + 1 ns = 11 ns



only in EI stage we will know if branch is taken or not till then fetch as usual

[If branch taken convert them → nop else do nothing]

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 ____4____

let N = no. of instructions

t_i = time taken by 1 instruction

k = 6 stages

no cycle time overhead of pipeline ≡ no stage delay ≡ ideal case

in pipeline 25% out of N instructions take 2 cycle extra (maybe due to branch stalling)

$0.25 \times N$ ⟶ each takes 2 cycles each i.e $0.25 \times 2 \times N$ extra cycles

generally in pipeline in 1 cycle 1 instruction comes out ⇒ 1 cc ≡ 1 inst

No. of cycle in pipeline = $k + (N-1)$

No. of cycle in pipeline with extra cycles due to stalls = $k + (N-1) + 0.25 \times 2 \times N$

T_{PL} = no. of cycles × time for each cycle

$= k + 1.5N - 1$

$$\text{speedup} = \frac{T_{non-PL}}{T_{PL}} = \frac{N \times t_i}{(k + 1.5N - 1) \times \frac{t_i}{6}} = \frac{6 \times N}{1.5N} = 4$$

assuming large no. of instructions $N \to \infty$

then $k-1$ becomes insignificant

as N increases so does 1.5 N

An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register writeback (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nanoseconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed. All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____.

breakdown of no. of cycles

$(5 + N-1 + 0.2 \times N \times 2)$

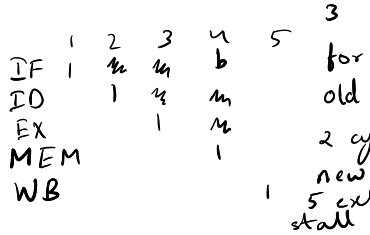
1st instruction goes through all 5 stage it takes 5 cycles

old PL IF: 1 ns, ID/RF: 2.2 ns, EX: 2 ns, MEM: 1 ns, WB: 0.75 ns k = 5

old cycle time = 2.2 ns

new cycle time = 1 ns

new PL ID: $\frac{2.2}{3}$ ns, RF1: $\frac{2.2}{3}$, RF2: $\frac{2.2}{3}$, EX1: 1 ns, EX2: 1 ns k = 8



for every branch old PL stalls for 2 cycles i.e extra.

new PL IF, ID, RF1, RF2, EX1, EX2, MEM

5 extra stall cycles 1 2 2 2 2 2 b

exec time = no. of cycle × cycle time

old exec time = $(5 + N-1 + 0.2 \times N \times 2) \times 2.2$ ns $= 4 + 1.4N = P$

new exec time = $(8 + N-1 + 0.2 \times N \times 5) \times 1$ ns $= 7 + N = Q$

$$\text{speedup} = \frac{P}{Q} = \frac{(4 + 1.4N) \times 2.2}{7 + N} = 1.4 \times 2.2 = 3.08$$

for large no. of instructions $N \to \infty$, +4, +7 insignificant

1.18.21 Pipelining: GATE CSE 2014 Set 3 | Question: 9

Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency.

- P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
- P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
- P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
- P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

clock time
2
1.5
1 → least
1.1

Which processor has the highest peak clock frequency?

A. $P1$
B. $P2$
C. $P3$
D. $P4$

max clock frequency
↳ least clock time

frequency $f = \frac{1}{time}$ as $t \uparrow$ $f \downarrow$
$t \downarrow$ $f \uparrow$

in pipeline clock time = max of all stage delays

1.18.22 Pipelining: GATE CSE 2015 Set 1 | Question: 38

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

nice explanation: https://gemini.google.com/app/66c510a4c6bf239a

clock low high
1 cycle

1 GHz ⇒ clock ticks 10^9 times in 1 second
i.e clock complete 10^9 cycles in 1 second

given:
Non pipeline,

frequency ↓ time ↓

2.5×10^9 cycles in 1 second $= 2.5 \times 10^9 \frac{cycle}{second} \Rightarrow \frac{1}{2.5 \times 10^9} \frac{seconds}{cycle}$

for 1 instruction takes 4 cycles $= 4 \frac{cycles}{instruction}$ → CPI

$\frac{1}{2.5} = \frac{10^2}{25_5} = 0.4$

let N = no. of instruction

Time for execution non PL =

no. of instructions × no. of cycles taken per instruction × time taken per cycle

$= N \text{ instructions} \times \frac{4 \text{ cycles}}{1 \text{ instruction}} \times (0.4 \times 10^{-9}) \frac{\text{seconds}}{1 \text{ cycles}}$

$= N \times 4 \times 0.4 \times 10^{-9}$ seconds

in pipeline, 1 instruction takes 1 cycle to finish ⇒ CPI = 1

Time for execution $_{PL}$

= no. of instructions × no. of cycles taken per instruction × time taken per cycle

clock rate = clock frequency = 2 GHz = 2×10^9 cycles / second

$\Rightarrow$ clock time = $\frac{1}{2 \times 10^9}$ sec/cycles = 0.5×10^{-9} sec/cycle

↳ This way of calculation is same for non-PL & PL since it doesn't involve no. of stages instead involves CPI

So even though no. of stages k is given as 5 it's not needed info

= $N \times 1 \times 0.5 \times 10^{-9}$ seconds

$$\text{Speedup} = \frac{\text{Time for execution}_{\text{non PL}}}{\text{Time for execution}_{PL}} = \frac{N \times 4 \times 0.4 \times 10^{-9} \text{ seconds}}{N \times 1 \times 0.5 \times 10^{-9} \text{ seconds}}$$

$$= \frac{4 \times 0.4}{0.5}$$

$$= \frac{1.6}{(1/2)} = 1.6 \times 2 = 3.2$$

1.18.23 Pipelining: GATE CSE 2015 Set 2 | Question: 44 top

Consider the sequence of machine instruction given below:

3 RAW Hazards

```
MUL   R5, R0, R1
DIV   R6, R2, R3
ADD   R7, R5, R6
SUB   R8, R7, R4
```

In the above sequence, $R0$ to $R8$ are general purpose registers. In the instructions shown, the first register shows the result of the operation performed on the second and the third registers. This sequence of instructions is to be executed in a pipelined instruction processor with the following 4 stages: (1) Instruction Fetch and Decode (IF), (2) Operand Fetch (OF), (3) Perform Operation (PO) and (4) Write back the result (WB). The IF, OF and WB stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD and SUB instruction, 3 clock cycles for MUL instruction and 5 clock cycles for DIV instruction. The pipelined processor uses operand forwarding from the PO stage to the OF stage. The number of clock cycles taken for the execution of the above sequence of instruction is _____.

IF, OF, WB 1 cycle, PO: ADD & SUB – 1 cycle

MUL – 3 cycle

DIV – 5 cycles

forwarding from PO → OF probably can resolve RAW Hazard

cc:	1	2	3	4	5	6	7	8	9	10	11	12	13
IF	M	D	A	S									
OF		M	D	A	A	A	A	A	A	A	S		
PO			M	M	M	D	D	D	D	D	A	S	
WB						M					D	A	S

arrows indicate forwarding of data causing corresponding RAW hazards (color coded)
total no. of cycles required = 13

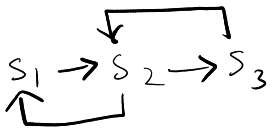
ADD instruction gets stalled in OF stage till correct data for R5 be forwarded from MUL & R6 be forwarded from DIV
only after both dep resolve A proceed
SUB don't need to stall

confirmation of my understanding https://gemini.google.com/app/53d0831c930e83be

1.18.24 Pipelining: GATE CSE 2015 Set 3 | Question: 51 top

Consider the following reservation table for a pipeline having three stages S_1, S_2 and S_3.

Time →	1	2	3	4	5
S_1	X				X
S_2		X			X
S_3			X		



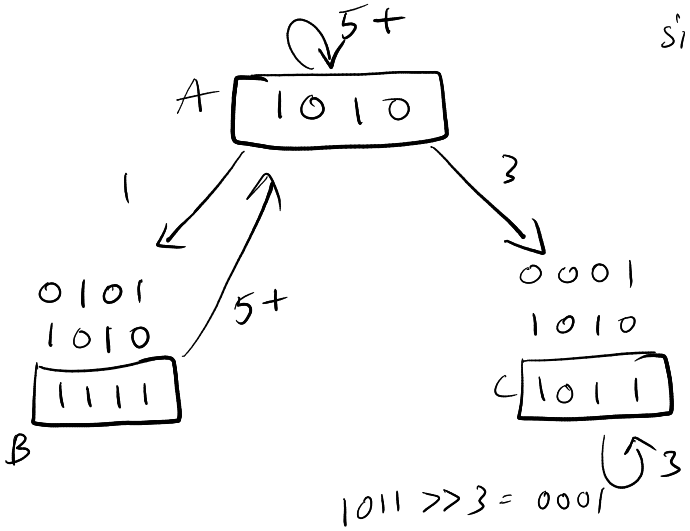
The minimum average latency (MAL) is _____

forbidden latencies 2, 4 → gaps between two X in a stage

max forbidden latency = 4 = m

collision vector of m = 4 bits is 1 0 1 0 (4 3 2 1)

position with non zero 1, 3 i.e permissible latencies



simple cycles:

$A \xrightarrow{1} B \xrightarrow{5} A$, $C \xrightarrow{3} C$

avg latency $\frac{1+5}{2} = \frac{6}{2} = 3$

avg latency $= \frac{3}{1} = 3$

MAL = 3

The stage delays in a 4-stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionality equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is _____ percent.

Throughput: number of instructions completed per unit of time

For a large number of instructions in an ideal pipeline, the throughput is approximately 1 / Cycle Time.

in pipeline 1 instruction takes 1 cycle to complete on average except for first instruction

time for 1 cycle to complete = max of all stage delay

old PL cycle Time = 800 ps → 1 instruction per 800 ps

new PL cycle Time = 600 ps → 1 instruction per 600 ps

old throughput = $\frac{1}{800}$ $\frac{\text{instructions}}{\text{picoseconds}}$

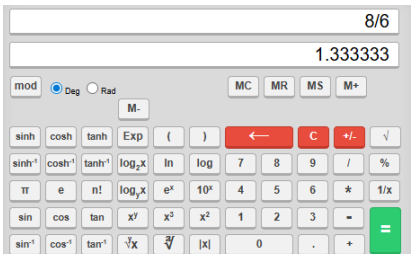
new throughput = $\frac{1}{600}$ $\frac{\text{instructions}}{\text{picoseconds}}$

old throughput x ? = new throughput $\Rightarrow$? = $\frac{\text{new throughput}}{\text{old throughput}}$

overall throughput : decrease (<1) | same ($=1$) | increase (>1)

$= \frac{\left(\frac{1}{600}\right)}{\left(\frac{1}{800}\right)} = \frac{800}{600}$

$\frac{1}{800} \times 1.33 = \frac{1}{600}$ i.e 33% ↑



$= 1.33$ i.e >1

∴ through put ↑

10% of 100 is being added

$100 \times 1.1 = 100 \times (1+0.1) = 100 + 10 = 110$ i.e 10% ↑ from original

$100 \times 2 = 100 \times (1+1) = 100 + 100 = 200$ i.e 100% ↑ from original

$100 \times 1.33 = 133$ i.e 33% ↑ " "

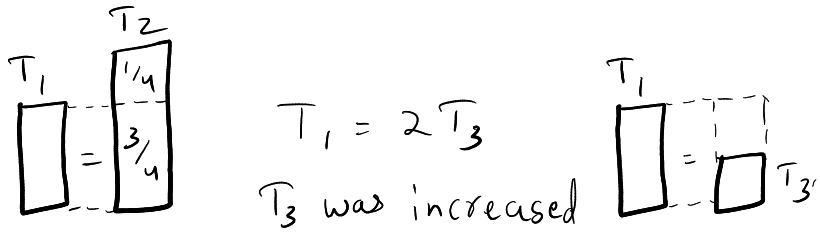
Consider a 3 GHz (gigahertz) processor with a three stage pipeline and stage latencies τ_1, τ_2 and τ_3 such that $\tau_1 = \frac{3\tau_2}{4} = 2\tau_3$.

If the longest pipeline stage is split into two pipeline stages of equal latency, the new frequency is _____ GHz, ignoring delays in the pipeline registers.

$K_{old} = 3$

clock frequency old = 3 GHz $= 3 \times 10^9 \frac{\text{cycles}}{\text{second}}$

clock time old $= \frac{1}{3} \times 10^{-9} \frac{\text{seconds}}{\text{cycle}}$



$T_1 = \frac{3}{4} T_2$

T_2 was decreased to make it equal to T_1

$\Rightarrow T_1 < T_2$

$T_1 = 2T_3$

T_3 was increased to make it equal to T_1

$\Rightarrow T_1 > T_3$

$T_2 > T_3$

$T_2 > T_1 > T_3 \leftarrow \{ T_1 > T_2 > T_3 \text{ or } T_2 > T_1 > T_3 \}$

$\therefore$ max stage delay = T_2

clock time = max stage delay + register delay $= \frac{1}{3} \times 10^{-9} \frac{\text{seconds}}{\text{cycle}}$

$= T_2 + 0$ (given ignore register delay)

$T_2 = \frac{1}{3} \times 10^{-9} \frac{\text{seconds}}{\text{cycle}}$ i.e longest stage's delay

$\Rightarrow T_1 = \frac{3}{4} T_2 = \frac{3}{4} \times \frac{1}{3} \times 10^{-9} \frac{\text{sec}}{\text{cycle}} = \frac{1}{4} \times 10^{-9} \frac{\text{sec}}{\text{cycle}}$

$\Rightarrow T_3 = \frac{T_1}{2} = \frac{1}{8} \times 10^{-9} \frac{\text{sec}}{\text{cycle}}$

given longest stage is split into 2 with equal latency

$T_2 <$ T_{2a} / T_{2b} each with new delay as $\frac{\frac{1}{3} \times 10^{-9}}{2} = \frac{1}{6} \times 10^{-9} \frac{\text{sec}}{\text{cycle}}$

T_1	T_{2a}	T_{2b}	T_3
1/4	1/6	1/6	1/8

new max stage delay is of stage 1 i.e $\frac{1}{4} \times 10^{-9} \frac{\text{sec}}{\text{cycle}}$

clock time $\Rightarrow$ clock frequency $= \frac{1}{t}$

$4 \times 10^9 \frac{\text{cycle}}{\text{sec}}$

i.e 4 GHz //

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 implementation 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 no hazards is _____ .

old clock time = old max stage delay + register delay
= 20 + 2 = 22 ns

new clock time = new max stage delay + register delay
= 12 + 2 = 14 ns

Time for execution non PL or PL =

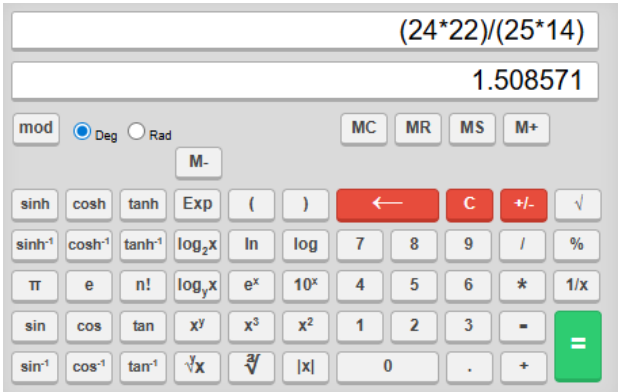
no. of instructions × no. of cycles taken per instruction × time taken per cycle

in pipeline on average it takes 1 cycle to complete 1 instruction
∴ no. of instructions = no. of cycles ∵ no. of cycles taken per instruction = CPI = 1

then simpler way to calculate execution time
= no. of cycles × time taken per cycle

no. of cycles required = K + (n−1)×1 i.e 1st instruction goes through all K stages ⇒ K cycles
∵ remaining n−1 instructions each take only 1 cycle

$$\text{speedup} = \frac{\text{old PL exec time}}{\text{new PL exec time}} = \frac{(5+20-1)\times 22\ ns}{(6+20-1)\times 14\ ns} = \frac{24\times 22}{25\times 14}$$



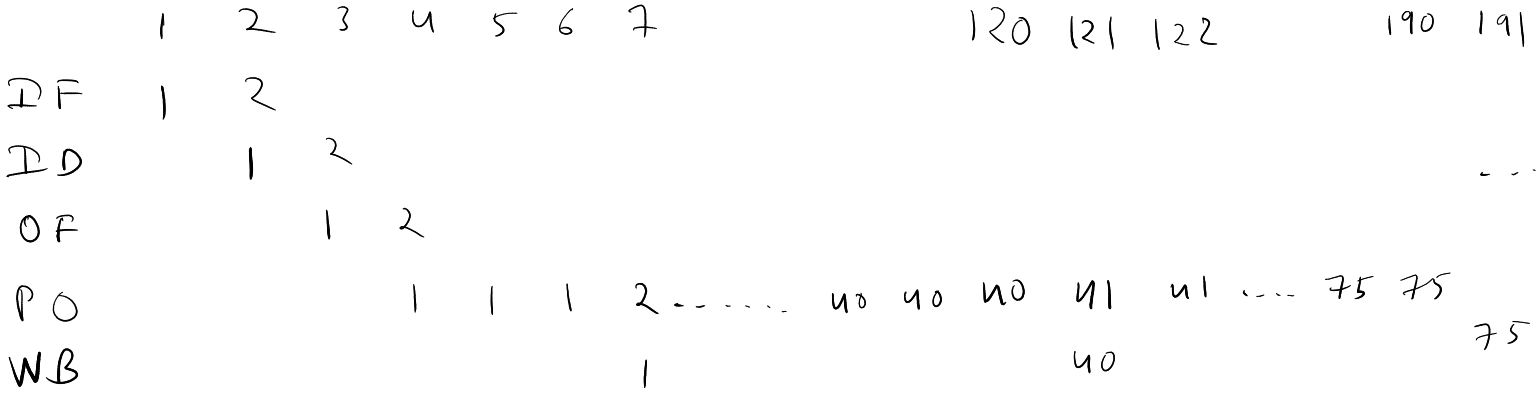
= 1.508

∴ speedup = 1.5 (as asked upto 2 decimal places)

1.18.28 Pipelining: GATE CSE 2018 | Question: 50 top https://gateoverflow.in/204125

The instruction pipeline of a RISC processor has the following stages: Instruction Fetch (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 _____.



$3 + 40\times3 + 35\times2 + 25\times1 + 1 = 219$ cycles

1st instruction: IF, ID, OF

Last 100th instruction: WB

1.18.29 Pipelining: GATE CSE 2020 | Question: 43 top https://gateoverflow.in/333188

Consider a non-pipelined processor operating at 2.5 GHz. It takes 5 clock cycles to complete an instruction. You are going to make a 5- stage pipeline out of this processor. Overheads associated with pipelining force you to operate the pipelined processor at 2 GHz. In a given program, assume that 30% are memory instructions, 60% are ALU instructions and the rest are branch instructions. 5% of the memory instructions cause stalls of 50 clock cycles each due to cache misses and 50% of the branch instructions cause stalls of 2 cycles each. Assume that there are no stalls associated with the execution of ALU instructions. For this program, the speedup achieved by the pipelined processor over the non-pipelined processor (round off to 2 decimal places) is_____.

N instructions

- 0.3 N — Mem — 1 — $0.05\times0.3N$ cause 50 cycles stall
- 0.6 N — ALU — 1 — No stall
- 0.1 N — Branch — 1 — $0.5\times0.1N$ cause 2 cycles stall

cycles required to execute N instructions in pipeline

$5 + N - 1 + \underbrace{50\times0.05\times0.3\times N}_{\text{extra cycle due to Mem stall}} + \underbrace{2\times0.5\times0.1\times N}_{\text{due to branch stall}}$

$= 4 + N + 0.85N$

$= 4 + 1.85N$

execution time = no. of cycles × time taken for each cycle

Non PL exec time = $5N$ cycles $\times \frac{1}{2.5\times10^9}$ $\frac{\text{second}}{\text{cycle}} = 2N$ nano seconds

PL exec time = $4 + 1.85N$ cycles $\times \frac{1}{2\times10^9}$ $\frac{\text{second}}{\text{cycle}} = \frac{4 + 1.85N}{2}$ nano second

$$\text{speedup} = \frac{\text{Non PL exec time}}{\text{PL exec time}} = \frac{2N}{\frac{(4+1.85N)}{2}} = \frac{4N}{(4+1.85N)} = \frac{4}{1.85} = 2.162$$

as $N \to \infty$

$+4$ is insignificant

1.18.30 Pipelining: GATE CSE 2021 Set 1 | Question: 53

https://gateoverflow.in/357398

A five-stage pipeline has stage delays of 150, 120, 150, 160 and 140 nanoseconds. The registers that are used between the pipeline stages have a delay of 5 nanoseconds each.
The total time to execute 100 independent instructions on this pipeline, assuming there are no pipeline stalls, is _____ nanoseconds.

clock time = max of all stage delay + register delay

= 160 + 5 nano seconds

execution time = no. of cycles required to execute N instructions in pipeline × time taken for each cycle

= (5 + 100 − 1) × 165 ns

= 104 × 165 ns

= 17160 ns

1.18.31 Pipelining: GATE IT 2004 | Question: 47

Consider a pipeline processor with 4 stages $S1$ to $S4$. We want to execute the following loop:

```
for (i = 1; i < = 1000; i++)
    {I1, I2, I3, I4}
```

where the time taken (in ns) by instructions $I1$ to $I4$ for stages $S1$ to $S4$ are given below:

	S_1	S_2	S_3	S_4
I1	1	2	1	2
I2	2	1	2	1
I3	1	1	2	1
I4	2	1	2	1

1st 2nd 3rd 4th

since instructions are in loop I_1 will be 5th instruction in pipeline in $i=2$ iteration

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

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

	1	2	3	4	5	6	7	8	9	10	11	12	13
S_1	1	2	2	3	4	4	5						
S_2		1	1	2	3		4	5	5				
S_3				1	2	2	3	3	4	4	5		
S_4					1	1	2		3		4	5	5

output of I_1 in $i=2$ after 13 ns

1.18.32 Pipelining: GATE IT 2005 | Question: 44 top

https://gateoverflow.in/3805

We have two designs $D1$ and $D2$ for a synchronous pipeline processor. $D1$ has 5 pipeline stages with execution times of 3 nsec, 2 nsec, 4 nsec, 2 nsec and 3 nsec while the design $D2$ has 8 pipeline stages each with 2 nsec execution time How much time can be saved using design $D2$ over design $D1$ for executing 100 instructions?

A. 214 nsec
~~B. 202 nsec~~
C. 86 nsec
D. −200 nsec

D_1 exec time $= (5 + 100 - 1) \times 4$ ns $= 416$ ns

D_2 exec time $= (8 + 100 - 1) \times 2$ ns $= 214$ ns

D_2 is faster than D_1

time saved $= 416 - 214 = 202$ ns

1.18.33 Pipelining: GATE IT 2006 | Question: 78 top

https://gateoverflow.in/3622

A pipelined processor uses a 4−stage instruction pipeline with the following stages: Instruction fetch (IF), Instruction decode (ID), Execute (EX) and Writeback (WB). The arithmetic operations as well as the load and store operations are carried out in the EX stage. The sequence of instructions corresponding to the statement $X = (S - R * (P + Q))/T$ is given below. The values of variables P, Q, R, S and T are available in the registers $R0, R1, R2, R3$ and $R4$ respectively, before the execution of the instruction sequence.

ADD	R5, R0, R1	; R5 ← R0 + R1
MUL	R6, R2, R5	; R6 ← R2 * R5
SUB	R5, R3, R6	; R5 ← R3 - R6
DIV	R6, R5, R4	; R6 ← R5/R4
STORE	R6, X	; X ← R6

eliminate options that don't start with 4

ADD	R5, R0, R1
MUL	R6, R2, R5
SUB	R5, R3, R6
DIV	R6, R5, R4
STORE	R6, X

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

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

1.18.34 Pipelining: GATE IT 2006 | Question: 79 top

https://gateoverflow.in/3623

A pipelined processor uses a 4-stage instruction pipeline with the following stages: Instruction fetch (IF), Instruction decode (ID), Execute (EX) and Writeback (WB). The arithmetic operations as well as the load and store operations are carried out in the EX stage. The sequence of instructions corresponding to the statement $X = (S - R * (P + Q))/T$ is given below. The values of variables P, Q, R, S and T are available in the registers $R0, R1, R2, R3$ and $R4$ respectively, before the execution of the instruction sequence.

WAR & WAW occur in super scaler i.e like cpu with multiple EX unit or other units, so Ignore

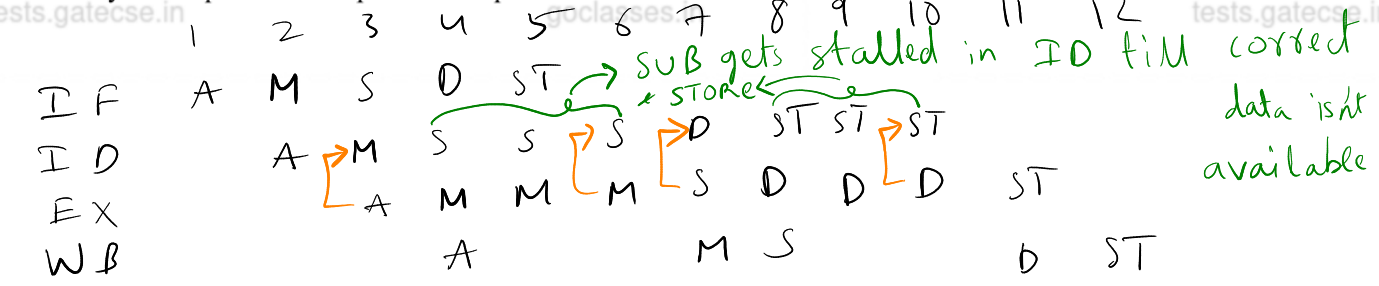
So only worry about RAW

ADD	$R5, R0, R1$	; R5 ← R0 + R1
MUL	$R6, R2, R5$	; R6 ← R2 * R5
SUB	$R5, R3, R6$	; R5 ← R3 - R6
DIV	$R6, R5, R4$	; R6 ← R5/R4
STORE	$R6, X$	; X ← R6

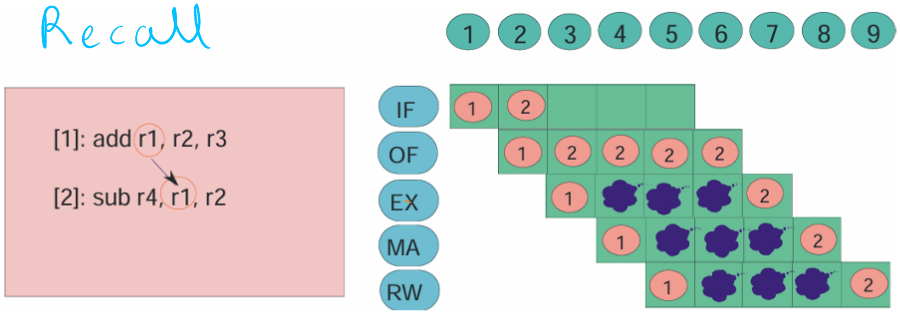
ways to handle data hazards like RAW
stalling
or forwarding (given)
EX → ID

The IF, ID and WB stages take 1 clock cycle each. The EX stage takes 1 clock cycle each for the ADD, SUB and STORE operations, and 3 clock cycles each for MUL and DIV operations. Operand forwarding from the EX stage to the ID stage is used. The number of clock cycles required to complete the sequence of instructions is

A. 10
~~B. 12~~
C. 14
D. 16



Recall



We have a RAW hazard

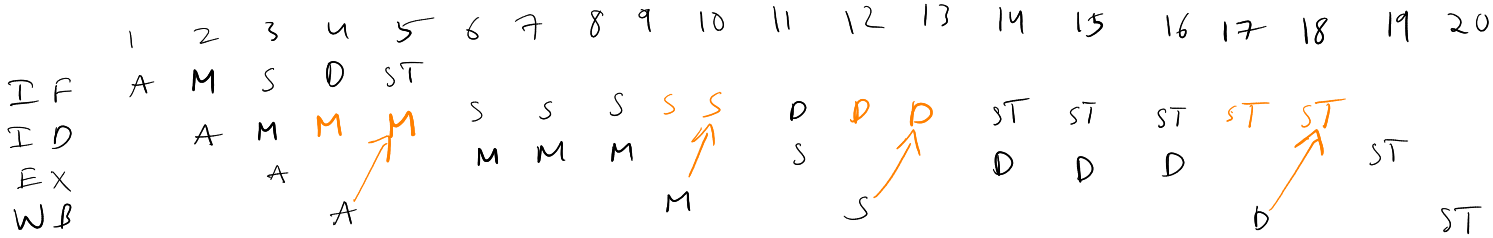
We need to stall, instruction [2] at the OF stage for 3 cycles.

We need to keep sending nop instructions to the EX stage during these 3 cycles

III^ly to s/w soln ① but that is done by compiler this is done by H/w itself

an extension to this, instead of stalling completely, due to forwarding we can save one or more cycle for each hazard.

lets see pipeline diagram for same instructions without Forward



instructions in orange waited/stalled till correct data was written to register & made available for them.

1.18.35 Pipelining: GATE IT 2007 | Question: 6, ISRO2011-25 top https://gateoverflow.in/3437

A processor takes 12 cycles to complete an instruction I. The corresponding pipelined processor uses 6 stages with the execution times of $3, 2, 5, 4, 6$ and 2 cycles respectively. What is the asymptotic speedup assuming that a very large number of instructions are to be executed?

A. 1.83
~~B. 2~~
C. 3
D. 6

no info on cycle time or register delays

& asked about speedup, since it is ratio these ungiven info would cancel out

$$\text{Speedup} = \frac{\text{no. of cycles required in non PL for 1 instruction}}{\text{no. of cycles required in PL for 1 instruction}}$$

$$= \frac{12}{6 \hookrightarrow \text{max of all PL stages so that all stages can be accomodated in 1 cycle}}$$

$$= 2$$

confirmation regarding my understanding https://gemini.google.com/app/254691a396ecabdb

A non pipelined single cycle processor operating at 100 MHz is converted into a synchronous pipelined processor with five stages requiring 2.5 $nsec$, 1.5 $nsec$, 2 $nsec$, 1.5 $nsec$ and 2.5 $nsec$, respectively. The delay of the latches is 0.5 $nsec$. The speedup of the pipeline processor for a large number of instructions is:

A. 4.5
B. 4.0
C. 3.33
D. 3.0

non PL clock time $= \frac{1}{100 \times 10^6} \frac{\text{seconds}}{\text{cycle}} = 10^{-8} \frac{\text{seconds}}{\text{cycle}}$

$= \frac{10}{10} \times 10^{-8}$

$= 10 \times 10^{-9} \frac{\text{second}}{\text{cycle}}$

$= 10 \text{ ns}$

PL clock time = max of all stage delays + register or latch delay

$= 2.5 \text{ ns} + 0.5 \text{ ns}$

$= 3 \text{ ns}$

$\text{Speed up} = \frac{\text{before}}{\text{after}} = \frac{10 \text{ ns}}{3 \text{ ns}} = 3.33$