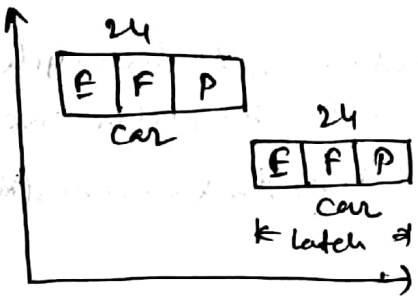
Pipelining

used for to increase the overall performance of CPU

Non-Pipelining - Single H/W component which can be take only 1 task at a time from its input and produce the result at the output.

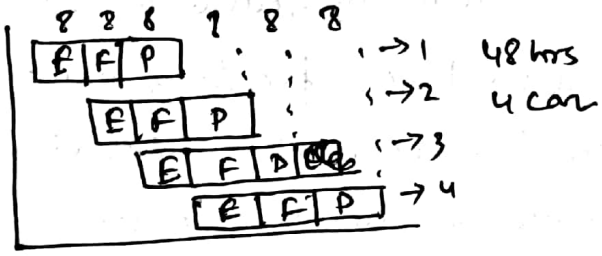
I/P → [H/W Design] → O/P

Drawback :- ⇒ only 1 input can be processed at a time
⇒ Partial o/p or segment o/p is not possible.



E- engine
F- Fitting
P- painting

It is sequencial work every person complete the work at first then started next Car making.

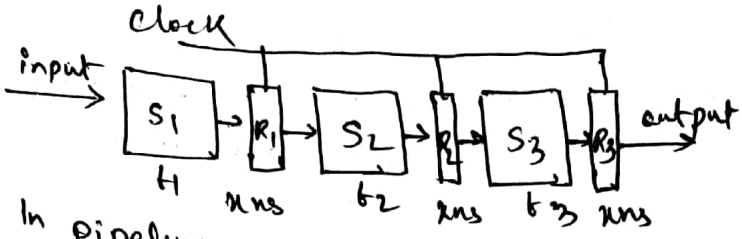


48 hrs
4 car

Pipeline — pipeline is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as pipeline processing.

Instead of a single H/W component we can split the H/W design into small component as segment.

The segment are connected with each other through an interface register & they can execute multiple tasks independently or parallely.



S → stage segment
t → stage time

In pipeline system each segment consists of an i/p register followed by a combination circuit. The register is used to hold data and combinational circuits performs operation on it. The o/p of combinational circuits is applied to the i/p register of next segment.

Type of pipeline —

1. Arithmetic pipeline
2. Instruction pipeline

→ calculation of 1 clock cycle Time:
Let tp → cycle time for pipeline system
tp = Max of (t1, t2, t3 & soon) + x ns (Register delay)

1. Arithmetic Pipeline :—

Its are usually found in most of the computers. They are used for floating point operators, multiplication of fixed point number etc.
For example — The input to the floating point Adder pipeline

$$X = A * 2^a \qquad Y = B * 2^b$$

Here A & B are mantissas (Significant digit of floating point no.), while a & b are exponent.

The floating point addition & substraction is done in 4 Parts

1. Compare the exponent.
2. Align the mantissas
3. Add or substract mantissas
4. Product the result.

Instruction Pipelining → It is a pipeline process which executes different instructions belonging to a process in its different stages.

Stages in an Instruction Pipeline

1. Instruction Fetch (IF) → In this stage the CPU reads instruction from the address in the memory whose value is present in the program counter.
   e.g. ADD, SUB.

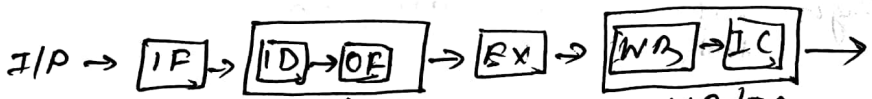
2. a) Instruction Decode (ID) → Instruction is decided and the register file is accessed to get the values from the register used in the instructions.

   b) Operand Fetch (OF) → e.g. A + B → Fetch

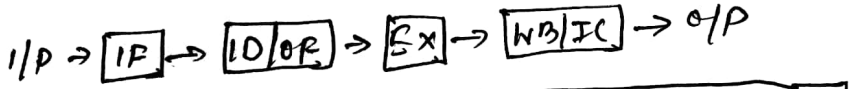
3. Instruction Execution (IE) → ALU operations are performed

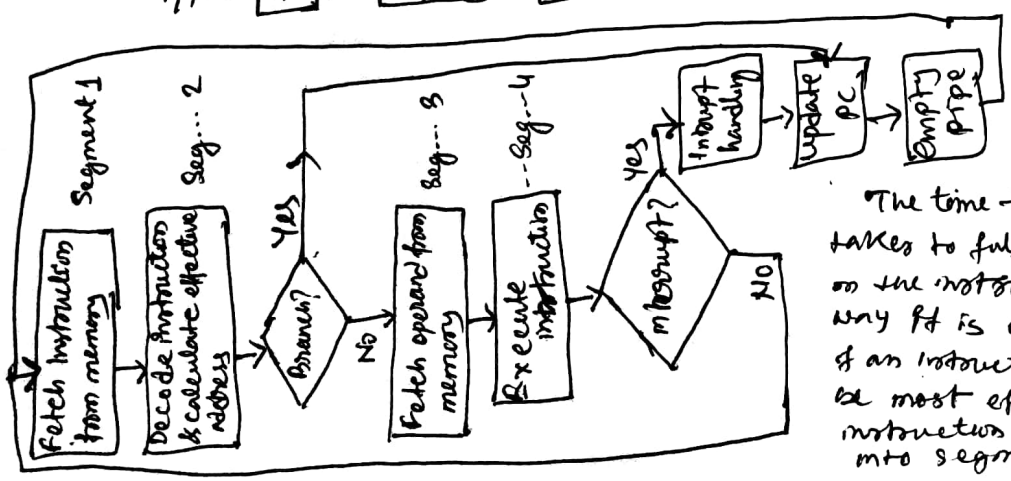
4. Result Write back to Main Memory or Register (WB)

5. Instruction checking (IC)



∵ ID & IC are very fast operations.





example of instruction pipeline

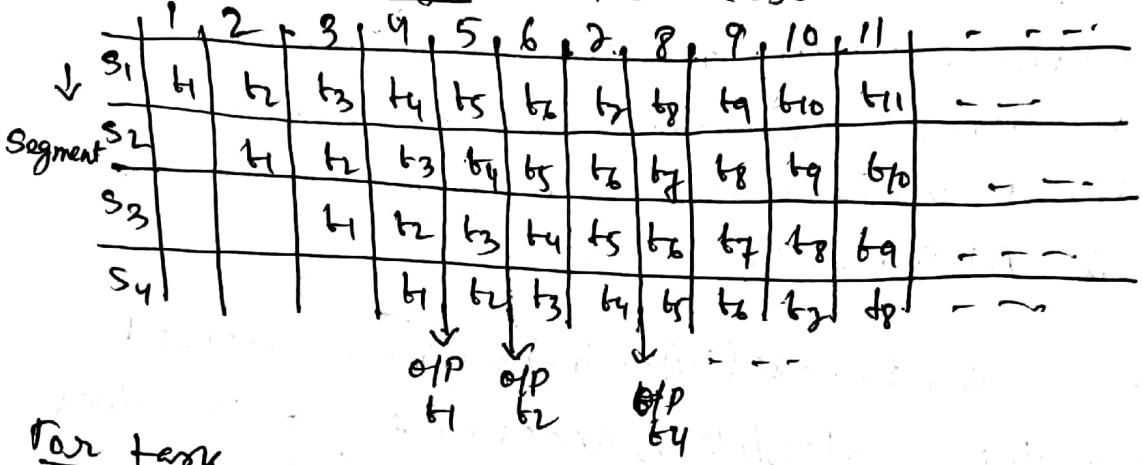
The time that each step takes to fulfill its f° depends on the instructions and the way it is executed. The design of an instruction pipeline would be most efficient if the instruction cycle is divided into segments to equal durations

(4)

Calculation of No. of clock cycles Required to process a group of Tasks

→ Let there be 100 task & they have to be execute in 4 stage pipelining system.

Time space Diagram → clock cycle



For task

2 → 5 (2+3) ← no of task

3 → 6 (3+3)

4 → 7 (4+3)

↑ Stages-1

$n = n + (K-1)$

$= K + (n-1)$ → Each task 1 clock cycle time each

Stages, no of task → above diagram.

Total no of clocks for 100 tasks

No. of clock cycles = $4 + 99 = 103$

(4 = K, 99 = n-1)

Q) 4- Instructions are executed in a 4-stage pipelining. Assume that each instruction requires different stage delay (in terms of no. of clock cycles) as mentioned in the table below :-

	IF	ID	EX	WB	
I_1	1	3	2	1	→ 7
I_2	2	2	3	1	→ 8
I_3	1	1	1	1	→ 4
I_4	2	1	1	2	→ 6

How many clock cycles will be required to complete these 4 instructions in the given pipeline system.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
IF	I_1	I_2	I_2	I_3	I_4	I_4							
ID		I_1	I_1	I_1	I_2	I_2	I_3	I_4					
EX					I_1	I_1	I_2	I_2	I_2	I_3	I_4		
WB							I_1			I_2	I_3	I_4	I_4

Answer → 13 clock cycles

Let's see the speed up in this case

$$\text{speed up} = \frac{t_{np}}{t_p} = \frac{7+8+4+6}{13} = \frac{25}{13} \approx 2$$

Instruction execution in pipeline are :-

- Speed up ratio (S)
- Frequency (f)
- Efficiency (E)
- Throughput (H)

④

⑤

Speed up — If we have n task & they have to be execu in a k-stage pipeline system & clock cycle time of pipeline system is t_p, then no. of clock cycles required to execute n tasks = $n + (k-1)$

$= k + (n-1)$

Total time required to execute n tasks

$= (n + k - 1)\, t_p$

or t_p = time of 1 clock cycle

If the clock cycle time for a non-pipeline system (time required by a non-pipeline system to execute 1 task) = $t_{np} = k \times t_p$

So, to execute n tasks in non-pipeline system, time required = $n \times t_{np}$

Speed up factors of pipeline system

$= \frac{\text{Time required by non-pipeline system}}{\text{Time required by pipeline system}}$

$\therefore S = \frac{n \cdot t_{np}}{(n+k-1)\, t_p}$

if $n \gg k$ then $(k+n-1) \approx n$

speedup ratio $\approx \frac{n \cdot t_{np}}{n\, t_p} \simeq \frac{t_{np}}{t_p}$

- Frequency(f) = $\frac{1}{t_p}$

- Efficiency $E_k = \frac{S_k}{k}$

$S_k = \frac{n \times t_{np}}{(k+n-1)\, t_p} = \frac{n \times k \times t_p}{(n+k-1)\, t_p}$

$= \frac{n \times k}{k+n-1}$

So, $E_k = \frac{S_k}{k} = \frac{n}{k+n-1}$

Throughput :- no. of instructions execute per ~~meter~~ clock cycles

$\eta = \frac{n}{(k+n-1)\, t_p} = \frac{\text{no of instructions}}{\text{Total time to complete the instructions}}$

$= \frac{nf}{(k+n-1)}$

CA-P5

Pipeline Model (Four stage)

Q) A 4 stage pipeline has the stage delay of 150 ns, 120 ns, 160 ns and 140 ns respectively. Registers that are used b/w the stages have a delay of 5 ns each. Assuming constant clocking rate, what is the total time taken by pipeline to process 1000 data items.

t_p = max (120, 150, 160, 140) + Register stage delay
= 160 + 5 = 165 ns

Total times = 165 (n + k − 1)
= 165 (1000 + 3) = 165.5 μs

Q) 5 Functional units operate using pipelining with clock cycle delay 10 ns, 8 ns, 10 ns, 10 ns, 7 ns. Assume pipeline have as 1 ns overhead. Find the speed up factors.

t_p = Max (10, 8, 10, 10, 10, 7) ns + 1 ns
= 10 + 1 = 11 ns.

For non-pipeline system, t_{np} = 10 + 8 + 10 + 10 + 10 + 7
= 45 ns

speedup factors = $\frac{45}{11} \approx 4$

Operand Forwarding Technique –

Main Topic – Hazard During Pipelining

Hazard: Circumstances that would cause incorrect execution if the next instruction were launched.

Data Hazar – When an instruction attempts to use a resource before it is ready. It is classified types, depending on the order of read & write access in the instructions.

There are three types of Data Hazards possible:-

↳ Read after Write Hazard [RW or True dependency problem] WAR

↳ Write after Read Hazard [RW] or Anti dependency RAW WAR

storing the data ↳ Write after Write Hazard [WW] output dependency WAW

* READ AFTER WRITE HAZARD –

Example – destination operands

i: ADD R_1, R_2, R_3 ; // $R_1 \leftarrow R_2 + R_3$

j: SUB R_4, R_1, R_6 ; // $R_4 \leftarrow R_1 - R_6$ (source)

	1	2	3	4	5
IF	i	j			
ID		ADD	SUB		
OF			$R_2 R_3$	$R_1 R_6$	
EX				R_2+R_3	R_1-R_6
WB					R_1

Result

→ Inconsistency

∴ R_1 Loaded at clock pulse 4 is not the correct value. It should have been done only after the result had been stored in R_1 at clock pulse 5.

Solution:-

Delay the Pipe: Solution for WR Hazard

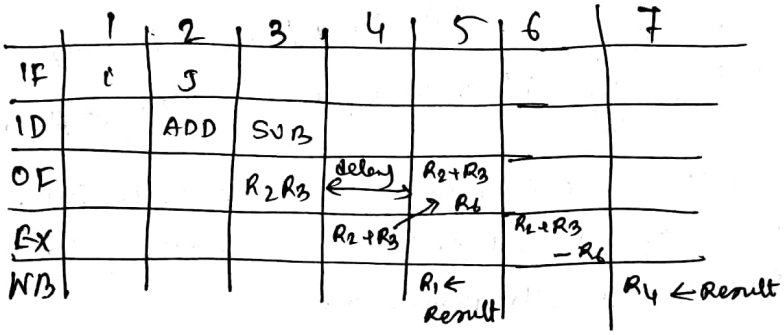
Idea:- Delay the execution of operand fetch till the Result gets stored in the register which is the cause of inconsistency.

	1	2	3	4	5	6	7	8
IF	i	j						
ID		ADD	SUB			$R_1 R_6$		
OF			$R_2 R_3$	R_2+R_3			$R_1 R_6$	
EX					$R_1 \leftarrow$ Result			$R_4 \leftarrow$ Result
WB								

Sol 2

Sol 2 Operand forwarding Technique.

idea :- Forward the result of execution unit of first instruction to the operand fetch unit of the second instruction, from where it can be read by the required instruction as their operand.



Benefit → Delay of only 1 clk cycle incurred in this case in comparison to 2 in Sol 1.

Ex – i : ADD R_1, R_2, R_3 J : SUB R_4, R_1, R_6

W_i (R_1) — Register in which the results are written; R_i (R_2, R_3); W_j (R_4); R_j (R_1, R_6)

$$W_i = \{R_1\}, \quad R_i = \{R_2, R_3\}$$

$$R_j = \{R_1, R_6\}, \quad W_j = \{R_4\}$$

Condition – If $i < j$ & $W_i \cap R_j \neq \phi$, then WR Hazard may occurs

Let's check for given example –

$W_i \cap R_j = \{R_1\}$ & $i < j$ (as per given order)

So, WR Hazard occur

WR Hazard – occur when an instruction refers to a result that has not yet been calculated or retrieved.

10)

Data Hazard :- When Instructions that exhibit data dependence modify data in different stages of pipeline

Write after Read Hazard (RW / Anti-dependency Hazard)

Example —

This can also occurs with both an arithmetic opn

i : ADD $R_1 R_2 R_3$; // $R_1 \leftarrow R_2 + R_3$

j : OR $R_2 R_3 R_4$; // $R_2 \leftarrow R_3$ OR R_4

Logical Instruction execution is faster than arithmetic. Hence, it may happen that the jth Instruction gets executed before the ith Instruction.

→ If in case this happens, then the ith instruction would add into R_2 a modified value of R2 & thus it would produce incorrect result. Thus it is called RW Hazard.

Condition of RW Hazard :-

If $i < j$ and $R_i \cap W_j \neq \phi$; then RW Hazard may occurs.

Solution :- Delay the Pipe → ~~Discussed in an elaborate manner in the Pipeline Stall, Pipeline~~ is also called pipeline stall, pipeline break, pipeline bubbling.

$R_i = \{R_2, R_3\}$, $W_j = \{R_2\}$

$R_i \cap W_j = \{R_2\} \neq \phi$

Control Hazard (Branch Hazard)

What is it?

on many instruction pipeline micro architectures, the process will not know the outcome of the branch when it needs to insert a new instruction into the pipeline (generally the fetch stage)

pipeline cannot operate normally due to control flow (non-sequential)

Example

```
S1
S2  if(cond^n)        # if the cond^n is false
S3                    Jump to S6
S4
S5
```

wastage of pipeline as this was never required.

	1	2	3	4	5	6	7	8
IF	S_1	S_2	S_3	S_4	S_5			
ID		S_1	if(cond)	S_3				
EX			S_1	T or F ≥ P			S_6	
WB				S_1				S_6

o/p of S_1 — 3 clocks cycles wasted as we did not get any o/p here — o/p of S_6

Drawbacks of this Hazard Problem caused by this Hazard:-

It reduces the speedup of the pipeline.

Structural Hazard :-

→ It occurs due to a resource conflict in the pipeline

→ A resource conflict is a situation when more than one instruction tries to access the same resource in the same cycle. [In the same clock Pulse]

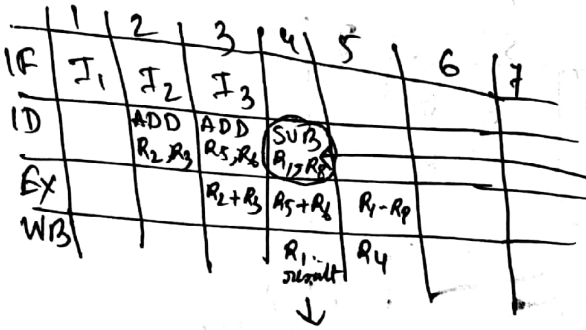
→ The resource can be register, memory or ALU.

Example-

I_1 : ADD $R_1 R_2 R_3$; // $R_1 \leftarrow R_2 + R_3$

I_2 : ADD R_4, R_5, R_6 ; // $R_4 \leftarrow R_5 + R_6$

I_3 : SUB R_7, R_1, R_8 ; // $R_7 \leftarrow R_1 - R_8$



In consistency as here needs previous value of R_1

R_1 used by two different instruction for different processor in the same clock cycle

Solution :

Delay the pipe (Delay I_3's ID phase till I_1 Complete)

↓

Also called pipeline Bubbling, Pipeline Stall, Pipeline Break.

Pipeline Hazards :- The hazard occurs due to unavailability of required data, instruction or resource during execution of an instruction in pipelining architecture.

I_1	F	D	E	WB		
I_2		F	D	E	WB	
I_3			F	D	STALL	E WB
I_4				STALL	F	D E STALL WB

STALL - It is the delay of execution process. This helps in not occurring pipeline hazards.

support + But due to STALL this instruction execution process takes more time for execution which needs to the decrease in the performance.

Hazard: Circumstances that would cause incorrect execution if the next instruction was launched.

→ Structural hazard :- It is basically arise bcz it don't have enough resources in a pipeline ~~system~~ implementation.

Load	1st clock cycle 1	2	3	4	5	6	7	8
~~Load instruction~~	1 IF	ID	EX	~~MEM~~ WB	WB			
	2	IF	ID	EX	MEM	WB		
	3		IF	ID	EX	MEM	WB	
	4			Stall	Bubble			

When two or more different instruction want to use same cycle e.g. MEM were the same memory part as IF as shown in this.

Dealing with structural hazard:

Stall
→ low cost, simple
→ increases CPI
→ Use for rate case

* Pipeline hardware resource
→ use for multi-cycle resources
→ good performance
→ sometimes complex e.g. RAM

Replicate resource
- good performance
- increases cost (+ may be interconnect delay)
- useful for cheap or divisible resource

Write after Write Hazard (WAW) :- output Dependency
↳ occurs when transaction tries to write an operand out of orders.

e.g./

i : $R_2 \leftarrow R_4 + R_7$

j : $R_2 \leftarrow R_1 + R_3$

→ If instruction j updates the value of R_2 before i, the final value written into register R_2 would be that of instruction i but what was required was of instruction j. Hence a WAW Hazard is said to occur.

Solution Delay the writeback of j until i finishes executing.

Condition of WW →

If $i < j$ & $W_i \cap W_j \neq \phi$ (intersection), WW Hazard may occur.

Checking for the current example

$$W_i = \{R_2\}$$

$$W_j = \{R_2\}$$

$$W_i \cap W_j = \{R_2\} \neq \phi$$

Hence WW Hazard may occurs.

Pipeline

- To improve the performance of a CPU we have two options:
  1. Improve the hardware by introducing faster circuits.
  2. Arrange the hardware such that more than one operation can be performed at the same time.
- Since there is a limit on the speed of hardware and the cost of faster circuits is quite high, we have to adopt the 2nd option

Arithmetic pipeline

$X = 0.9505 \times 10^3 = A \times 10^a$

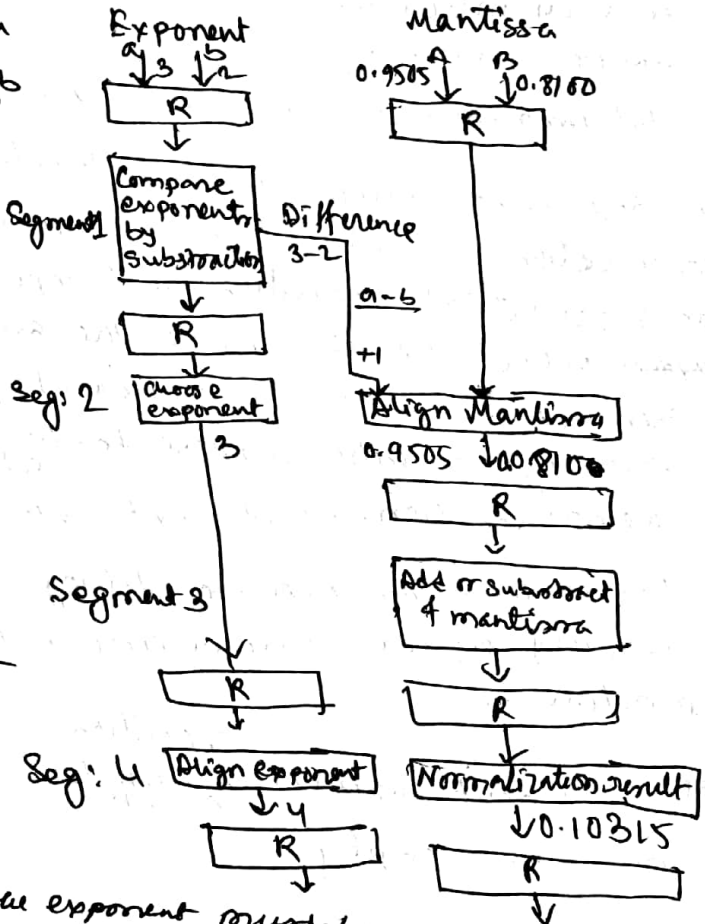
$Y = 0.8100 \times 10^2 = B \times 10^b$

$= 0.0810 \times 10^3$

$Z = X + Y$

$= 1.0315 \times 10^3$

$= 0.10315 \times 10^4$



Pipeline for floating point addition / Subtraction

In this arithmetic pipeline used to do some arithmetic op^n like FP additions. We considering that these type of pipeline 4 stage or there stage 1 – 4.
let us the no.

In case of floating point add/sub the exponent must be same for the both the no. then the mantissa of the significants can get, before the significants add/sub we calculate exponent. explain problem.

CPO-2 CA-P5

Instruction pipelining

Step → / Instruction ↓	1	2	3	4	5	6	7	8	9	10	11	12	13
1	IF	ID	OF	Ex									
2		IF	ID	OF	Ex								
Branch 3			IF	ID	OF	Ex							
4				IF	-	-	IF	ID	OF	Ex			
5					-	-	-	IF	ID	OF	Ex		
6					•	-	-	-	IF	ID	OF	Ex	
					·		-	-	-	IF	ID	OF	Ex

Timing of instruction pipeline

We know that a program consist of multiple instruction in instruction pipeline architecture each an every instruction will get executed on different stages or phases. So,

1st instruction IF then goto ID in 2nd cycle & soon.

In 3rd instruction, this instruction is having the category Branch - jump instruction conditional & unconditional. What will happen linear or sequential executions are hamper as that the next instruction a result in the program update which is just update the 3rd instruction, is not executable.

Then bcz due to the presence of jump or goto statement instruct that program control suppose to the control to target the different get transfer memory location. Then next instruction fetch to unknown

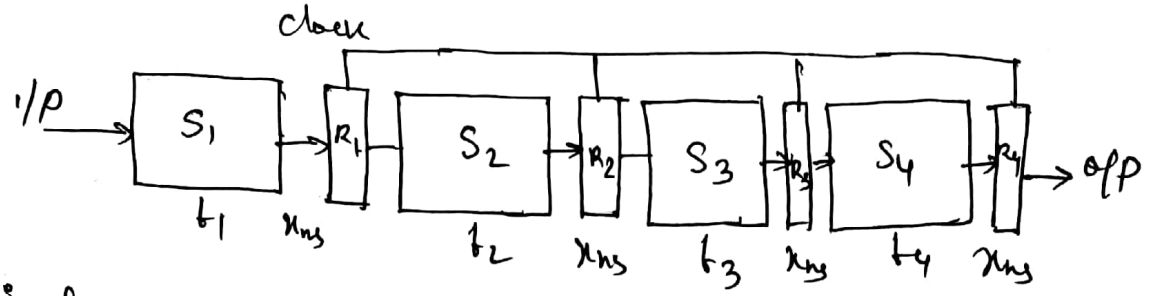
This is know as pipeline stalling.

Then why pipeline efficiency decrease. And the the detoriation of the performance the pipeline is

Stall - The next instruction can be cleaned up means the pipeline is empty. will execution

CA-P5

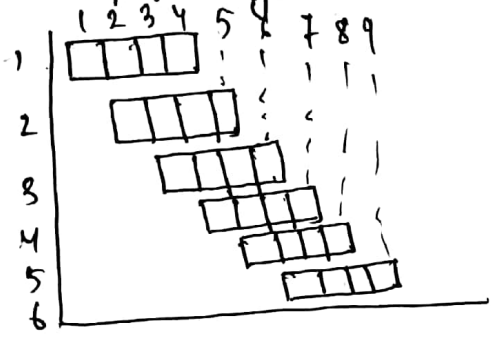
Pipeline Model (four stage)



In pipeline system each segment consists of an i/p register followed by a combinational circuit. The register is used to hold data and Combinational circuits perform operation on it. The o/p of combinational circuit is applied to the i/p register of next segment.

Example

1. Suppose an instruction is divided into 4 segment and a program has 6 instruction. Find the total time required to complete all operations in pipelining as well as non-pipeling.



Sol.

$t_{np} = N \times S \times T$

$= 6 \times 4 \times T = 24T$ sec

$t_p = 1 \times S T + (N-1) T$ (cycle time)

$= 4T + 5T = 9T$

t_p = Time taken for 1st task + Time taken for remaining tasks

2. If a processor takes 20 ns to execute a sub operation, assume that a pipeline has 4 segments & execute 100 no. of instructions in a sequence. Find the following.

(a) Time taken to execute all the instruction in
a) non pipeline (b) pipeline

c) What is the overall speed of pipeling over non-pipeling.

$T = 20$ns
$S = 4$
$N = 100$

a) $t_{np} = 100 \times 4 \times 20$ ns $= 8000$ ns

b) $t_p = 4 \times 20 + 99 \times 20$
$= 103 \times 20$ ns $= 2060$ ns

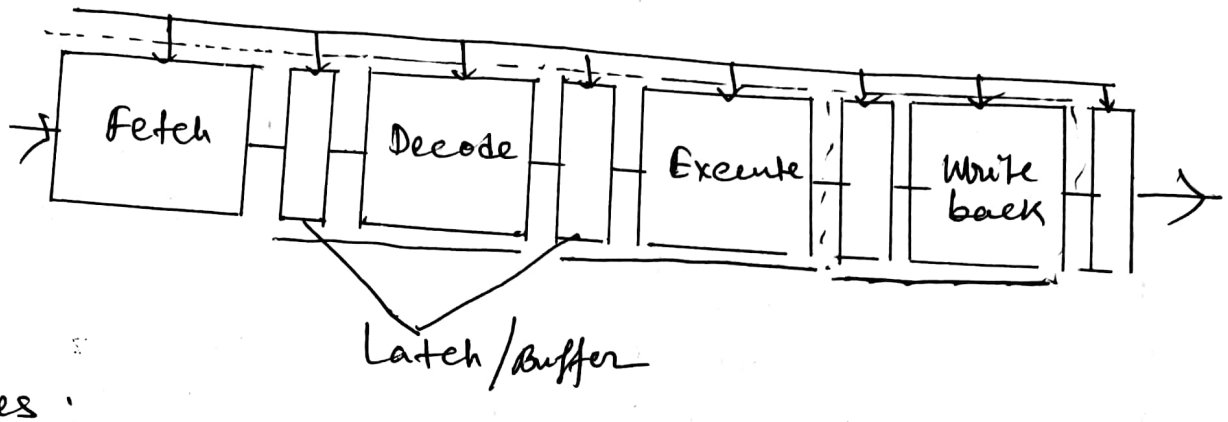
c) Overall speed of P over np $= 8000 \div 2060 = 3.88$

$\therefore$ pipeling is 3.88 times faster than non-pipeline.

CA-PG

~~Pipeline for~~

~~Four~~ 4 stage pipeline for a RISC

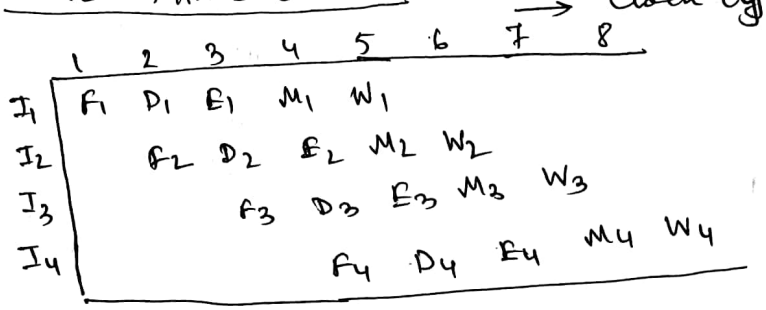


Stages :

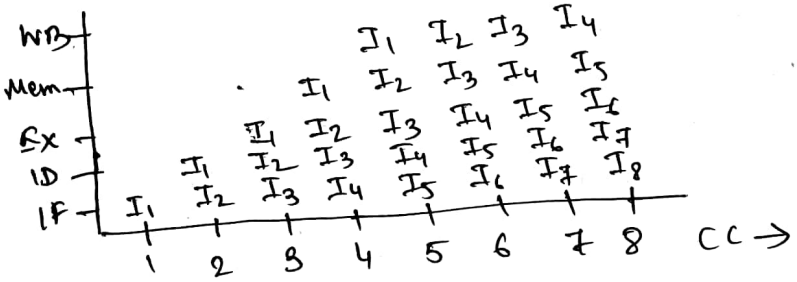
1. Instruction Fetch (IF) :-
   - → Fetching instruction
   - — Incrementing Program Counter
2. Instruction Decode (ID) :-
   - — Interpretation of opcode
   - — Decode and search where source registers are present
   - — Read the register contents ⇒ check for branch
3. Execution (Ex) :-
   - — Effective address computation.
   - — performs the operation specified by ALU.
4. Memory operation (Mem)
   - — Read
   - — Write
5) Write back (WB)
   - — Wrote the result into register file.

CA-P7

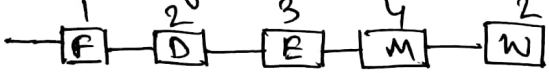
TIME SPHASE DIAGRAM



Or



Q. How many clock cycle required to finished.



For 1st instruction = 12 cc required (1+2+3+4+2)

For 2nd instruction = 16 cc required

Pipeline takes the slowest stage time :—

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
I_1	F_1	F_1	D_1	E_1	E_1	E_1	M_1	M_1	M_1	M_1	W_1	W_1								
I_2		F_2	–	D_2	D_2	–	E_2	E_2	E_2	–	M_2	M_2	m_2	m_2	W_2	W_2				
I_3			F_3	–	–	D_3	D_3	–	–	E_3	E_3	E_3	–	–	M_3	M_3	M_3	M_3	W_3	W_3
I_4																				

Ex-P2

4- Instructions are executed in a 4 stage pipelining. Assume that each instruction requires different stage delay (interms of no. of clock cycles) as mention in the table below.

	IF	ID	EX	WB	
I_1	1	3	2	1	→ 7
I_2	2	2	3	1	→ 8
I_3	1	1	1	1	→ 4
I_4	2	1	1	2	→ 6

How many clock cycle will be required to complete these 4 instructions in the given pipeline system.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
I_1	IF	D_1	D_1	D_1	E_1	E_1	W_1						
I_2		F_2	F_2	D_2	D_2	—	E_2	E_2	E_2	W_2			
I_3				F_3	—	D_3	—	—	—	E_3	W_3		
I_4					F_4	F_4	D_4	—	—	—	E_4	W_4	W_4

Answer → 13 clock cycle.

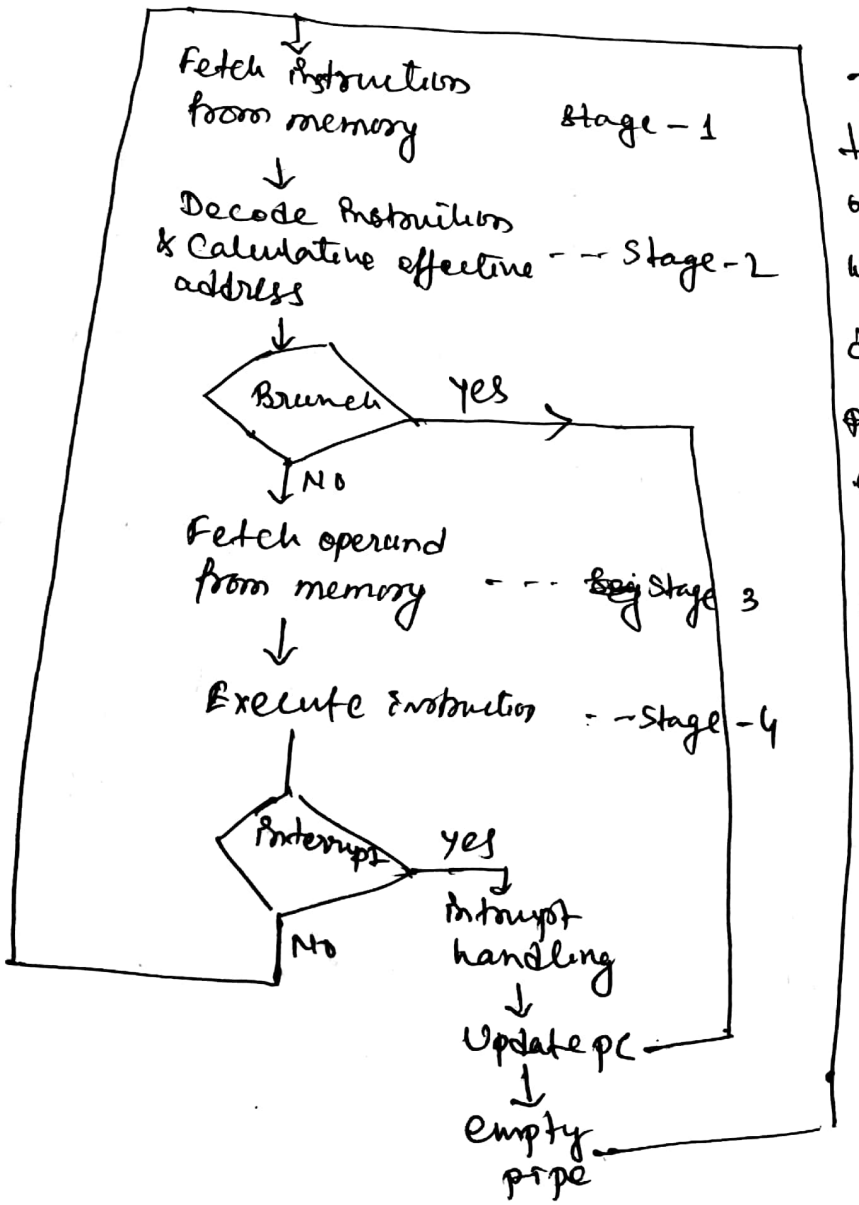
Classification of Pipeline :-

Instruction Pipelining — Execution of stream of in instructions is pipelined by overlapping the execution of current instruction with fetch, decode, and operand fetch of subsequent instructions - These are also called instruction look-ahead pipeline. It is inter-instruction pipeline.

It is executed by follows steps.

~~IF, ID,~~ - Fetch the opcode
- Decode
- Fetch the operand
- Execution of the above instr.
- Store / write back results

CA-PQ



019 - A2

The time that each step takes to fulfill its fn depends on the instruction and the way it is executed. The design of an instruction pipline would be most efficient of the instruction cycle is devided into segment to equal duration.

CA-P10

Arithmetic Pipeline — The ALU of Computer is segmented in various data formats for pipeline operations. They are used for floating point operators, multiplication of fixed point no. etc.

example :- The input to the floating point Adder pipeline

$$X = A * 2^a \qquad Y = B * 2^b$$

Here A & B are mantissas (significant digits of floating point no.) while a & b are exponent.

The floating point addition & substraction is done in Parts.

1. Compare the exponent
2. Align the mantissas
3. Add or Subtract mantisses
4. Produce the result.

$N = \pm M \times r^{\pm E}$

r = representation of base Number = N

$X = 0.9784 \times 10^3$

$Y = 0.8621 \times 10^2$

Step1 :- Comparision of exponent

step2 : If it is unequal, then allign the mantissa with respect to exponent $Y = 0.0862 \times 10^3$

Step 3 :- Add / subtract mantissas

$Z = (0.97840 + 0.08621) \times 10^3$

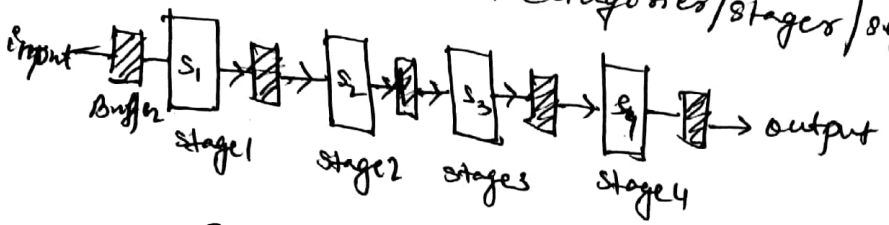
$= 1.06461 \times 10^3$

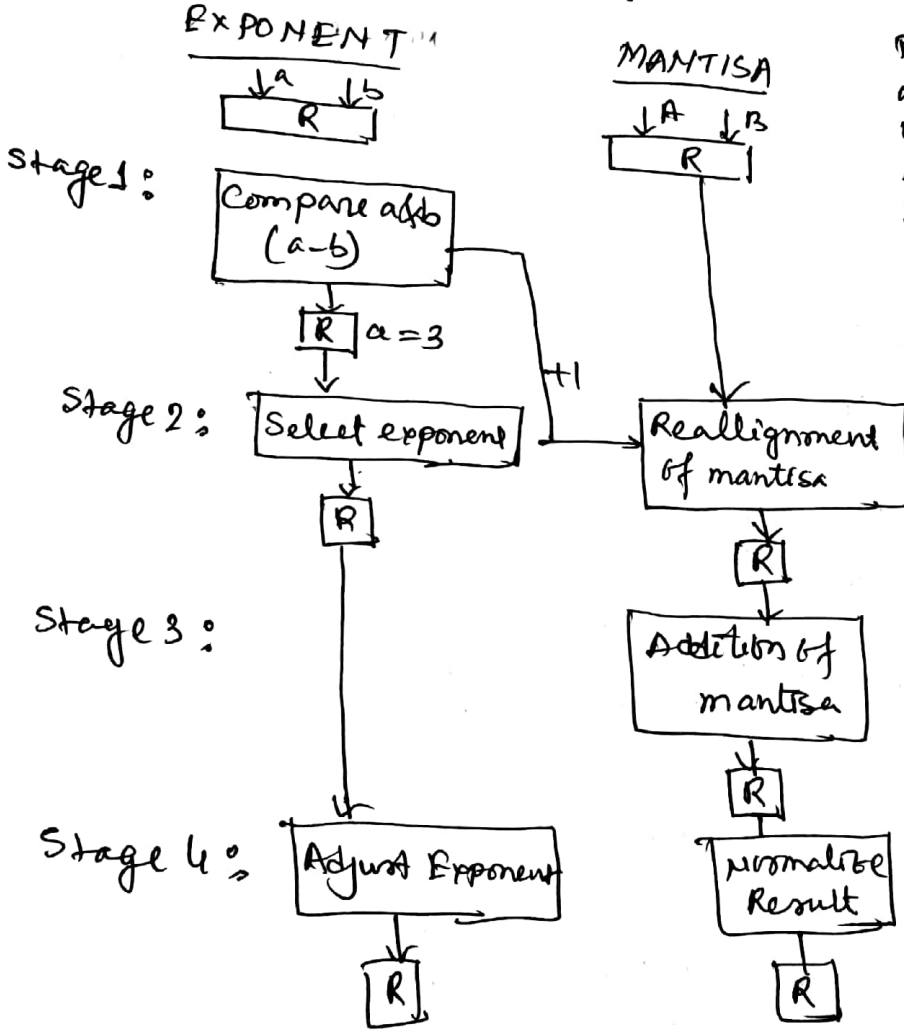
Step 4 :- Normalize the mantisa if necessary.

CA-P11

Architecture of Arithmetic Pipelining :-

→ It is divided into 4 Catagories/Stages/segments





In this arithmetic pipeline used to do some arithmetic ops like FP addition. We considering that there type of pipeline 4 stages or phase

Let us the no.

In case of floating point add/sub the exponent must be same for both the no. then the mantisa of the significant can get before the significant ion add/sub. I should equalize exponent.

Instruction execution in pipeline are :-

- Speed up ratio (S)
- Frequency (f)
- Efficiency (E)
- Throughput (H)

Processor Pipeline :-

Some data streams is processed by a cascade of processors. Each processor does some specific task.

$$\text{Speedup ratio} = \frac{\text{Execution Time in a Non-pipelined Arc}}{\text{Execution Time in a pipeline Architecture}}$$

Let.

T_{np} = Execution Time of a single instruction in a non-pipeline architecture.

T_p = Execution time in a pipeline architecture

N = No of instructions to be executed

K = No. of steps / stages.

So. Speedup ratio $= \frac{N \cdot T_{np}}{K \cdot T_p + (N-1)T_p} = \frac{N \cdot T_{np}}{(N+K-1)T_p}$

When $N >> K$, we can ignore K $= \frac{N\, T_{np}}{N\, T_p}$

$\therefore$ Speedup Ratio $= \frac{T_{np}}{T_p}$

In ideal case pipeline $T_{np} = K\, T_p$

Speedup ratio $= = K \frac{T_p}{T_p}$

$\therefore$ Speedup ratio $\propto K$

CA- P14

Calculate cycle Time

In pipelined architecture —

- There is global clock that synchronizes the working of all stages.
- At the begining of each clock cycle, each stage reads the data from its register and process it.
- Frequency of clock is set such that all the stages are synchronized.
- Cycle time is the value of one clock cycle.

There are two possible cases.

Case 1 All the stages offer same delay

Cycle time = Delay offered by one stage including the delay due to its register.

Case 2 All the stages do not offer same delay :-

Cycle time = Maximum delay offered by any stage including the delay due to its register

Q. A 4 stage pipeline has the stages delay of 150 ns, 120ns 160ns and 140 ns respectively. Register that are used b/w the stages have a delay of 5 ns each. Assuming constant clocking rate, what is the total time taken by pipeline to process 1000 data items.

$$T_p = \max(120, 150, 160, 140) + \text{Register stage delay}$$

$$= 160 + 5 = 165 \text{ ns}$$

$$\text{Total time} = 165\,(n + k - 1) = 165\,(1000 + 3) = 165.5 \text{ ns}$$

CA-P13

- Calculating Frequency of clock :—

$$\text{Frequency of the clock } (f) = \frac{1}{\text{Cycle time}}$$

$$= \frac{1}{Tp}$$

- Efficiency (E)

$$\text{Efficiency } (\eta) = \frac{\text{Speed up}}{\text{No. of stages in Pipelined Architecture.}}$$

$$= \frac{S_K}{K}$$

$$\therefore S_K = \frac{n \times Tnp}{(K+n-1)Tp}$$

$$= \frac{n \times K \times Tp}{(n+k-1)\,Tp} = \frac{nK}{n+k-1}$$

$$\therefore E_K = \frac{S_K}{K} = \frac{n}{K+n-1}$$

- Throughput :— No. of instruction executed per unit time

$$H = \frac{n}{(K+n-1)Tp} = \frac{\text{No. of Instruction executed}}{\text{Total time taken.}}$$

$$= \frac{nf}{(K+n-1)}$$

CA-P15

5 functional units operate using pipeline with Clock cycle delay 10ns, 8 ns, 10ns, 10ns, 7ns. Assum pipeline have or 1ns overhead. Find the speed up factors.

T_p = max (10, 8, 10, 10, 7) ns + 1ns

= 11 ns.

For non pipeline system, T_{np} = 10+8+10+10+7 = 45ns

$$\text{Speed up factor} = \frac{45}{11} = \simeq 4$$

Q The stage delays is a 4 stage pipeline are 800, 500, 400, 300 pico seconds. The first stage is replaced with a f^b equivalent design involving two stages with respective 600 and 350 picoseconds. The throughput increase of the pipeline is ____ %.

Execution Time in 4 stage Pipeline

T_p = max (800, 500, 400, 300) + 0 = 800 picosecond.

Throughput (4 stage) 1/800 =

Execution Time in 2 Stage

Cycle time (tp) = max (600, 350) + 0

= 600 ps

Throughput 2 stage pipeline

Throughput = Number 1 instruction / 600 ps

Throughput increase

= {Final throughput − Initial throughput) / Initial throughput} × 100

= {(1/600 − 1/800) / (1/800)} × 100

= {(800/600) − 1} × 100 = (1.33 − 1) × 100

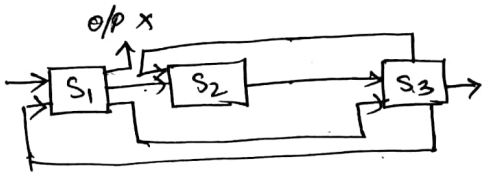
= 33.33%.

CPD-1

Pipeline Optimization

1. Linear pipeline are static pipeline bcz they are used to perform fixed function. The output of the pipeline is produced from the last stage. Static pipeline is specified by single Reservation table. The reservation Table is trivial in the sense that data flows in linear streamline. All initiations to a static pipeline use the same reservation table.

   Non-linear pipeline are dynamic pipeline bcz they can be reconfigured to perform variable fns at different times. Functions partitioning is relatively difficult because the pipeline stages are interconnected with loops (feedforward & feedback) in addition to streamline connections. The output of the pipeline is not necessarily produced from the last stage. Dynamic pipeline may allow different initiations to follow a mix of reservation table.



A Three stage pipeline.

A reservation Table is a way of representing the task flow pattern of a pipelined system. A reservation table has several rows and columns. Each row of the reservation table represents one resource of the pipeline and each column represent one time-slice of the pipeline.

Time →	1	2	3	4	5	6	7	8
Stage S_1	X					X		X
S_2		X		X				
S_3			X		X		X	

Diagram → RT Table → Collision vector → forbidden latency
                                      → permissible "
State diagram
(i) Simple cycle (ii) Greedy cycle (iii) MAL (minimum avg latency)

CPO-2

latency — The number of time units (clock cycles) b2n two initiations of a pipeline is the latency b2n them. Latency values must be non negative integers.

Any attempt by two or more initiations to use the same pipeline stage at the same time will cause a collision. A collision implies resource conflicts b2n two initiations in the pipeline. Latencies that cause collisions are called forbidden latencies.

To detect a forbidden latency, one needs simply to check the distance b2n any two clock marks in the same row of the reservation Table.

e-g The distance b2n the 1st mark and the second mark in row S_1 in fig. is 5 is a forbidden latency.
Similarly latencies 2, 4, 5, 7 are all seen to be forbidden from inspecting the same reservation table.

Forbidden latency = {(6-1), (8-1), (8-6)}; (4-2), (5-3); (7-3), (7-5)}

= {5, 7, 2, 2, 2, 4, 2}

= {2, 4, 5, 7}.

Collision Free Scheduling —

The main objective is to obtain the shortest avg. latency b2n initiations without causing collisions.

Then we study —

collision vector — is a method of analyzing how often we can initiate a new operation into the pipeline and maintain synchronous flow without collisions.

For reservation table with n columns, the maximum forbidden latency $m \leq n-1$.

The permissible latency p should be as small as possible.

The choice is made in the rang $1 \leq p \leq m-1$

CPO-3

A permissible latency of $p=1$ corresponds to the ideal case.

The combined set of permissible and forbidden latencies can be easily displayed by a collision vector, which is an m-bit binary vector. $C = (C_m C_{m-1} \cdots C_2, C_1)$

The value of $C_i = 1$ if latency i causes a collision and $C_i = 0$ if latency i is permissible.

Note that it is always true the $C_m = 1$, corresponding to the maximum forbidden latency.

permissible latencies = {1, 3, 6, X}

Collision vector = $\{C_7 C_6 C_5 C_4 C_3 C_2 C_1\}$

1 0 1 1 0 1 0

The collision Vector $C_x = \{1011010\}$ for f^n X

$C_y = \{1010\}$ for f^n Y

CPO CPO-4

Simple cycle (No one visited in twice

Not a simple cycle

List of all the simple cycle

cycle	Avg. latency
(1,8)	4.5
(3,8)	5.5
(5,8)	7

Greedy cycle – Greedy cycle are those cycle in which avg. latency ≤ no. of bits in the initial collision vector.