Performance Analysis :-

- Performance is an indirect measurement depends on the execution time.

$$\text{Performance} \propto \frac{1}{\text{Execution time}}$$

- Execution time means the time required to complete the program.

Exec. time (CPU time) = no. of seconds / program

$$= \text{Inst}^n/\text{prog} \times \text{cycles}/\text{Inst}^n \times \text{second}/\text{cycle}$$

$\text{Inst}^n/\text{prog}$ → Inst^n count (IC)

$\text{cycles}/\text{Inst}^n$ → CPI

$\text{second}/\text{cycle}$ → CT

$$\text{CPU time} = IC * CPI * \text{cycle time}$$

In the program, different Inst^n are taking different no. of cycles to complete program execution. So,

$$\text{Program execution time} = \sum (IC_i * CPI_i)\ CT$$

where i is type of inst^n

- Speed Up (S) factor is used to measure the perf-ormance gain. i.e

$$S = \frac{\text{Performance } x}{\text{Performance } y}$$

$$S = \frac{\frac{1}{ET_x}}{\frac{1}{ET_y}}$$

$$\left(S = \frac{ET_y}{ET_x} = n \right)$$

* this means system x will runs n times faster than system y

Eg:-

sys X

[Task]

$ET_x = 5ns$

sys y

[Task]

$ET_y = 10ns$

so, $$S = \frac{ET_y}{ET_x} = \frac{10}{5}$$

$$\therefore (S = 2)$$

Thus sys x is 2 times faster than sys y

So, relation b/w system x & y is -

$$S = \frac{y}{x}$$ since $S = 2$

so, $y/x = 2 \Rightarrow \boxed{y = 2x}$

Q. Consider 2ns clock cycle processor used to execute the following code :

Inst^n	Inst^n count (IC)	CPI
LOAD	40	8
STORE	60	6
ALU	20	3
BRANCH	40	2

what is the performance of a system in terms of MIPS (Million Inst^n per second).

$$ET \text{ of prog} = \sum_i (IC_i * CP_i)\ CT$$

$$\text{ET of prog} = (40(8) + 60(6) + 20(3) + 40(2))\ 2$$

$$= (320 + 360 + 60 + 80)\ 2$$

$$= 820 \times 2 \text{ ns}$$

$$\text{Avg. Inst}^n \text{ ET} = \frac{\text{ET of prog}}{\text{Total no. of Inst}^n \text{ in prog}}$$

$$= \frac{820}{\Sigma IC} = \frac{820 \times 2 \text{ ns}}{160}$$

$$= 10.25 \text{ns}$$

1 Inst^n $\longrightarrow$ 10.25 ns

$\frac{10^9}{10.25}$ Inst^n $\longleftarrow$ 1ns

Thus (performance = 97.5 MIPS)

Q. Consider 2.5 ns clock cycle processor which consumes 9 cycles for data transfer Instⁿ, 6 cycles for ALU instⁿ and 3 cycles for Branch instⁿ. Relative frequencies of these Instⁿ are 40%, 40%, & 20% respectively. System is enhanced to make avg. CPI = 1, in this process CT is increased upto 30%. What is the performance gain.

$$ET_{old} = \Sigma (IC_i * CP_i)\ CT$$

$$= [(0.4 * 9) + (0.4 * 5) + (0.2 * 3)]\ 2.5\ ns$$

$$= \underline{16.5\ ns}$$

$$ET_{new} = \Sigma (IC_i * CP_i)\ CT$$

$$\{CPI = 1\} = [0.4 + 0.4 + 0.2]\ [2.5 + (30\% \text{ of } 2.5)]$$

$$= \underline{3.25\ ns}$$

$$S = \frac{ET_{old}}{ET_{new}} = \frac{16.5\ ns}{3.25\ ns}$$

$(S = 5.07)$ so new system runs 5.07 times faster than old system

Amdhal's Law -

- It states that improve the performance within cost & price limit while making the common case fast - (Improve the performance by enhance part of the system rather then the replacement of the system)

- It focused on the performance gain after enhance a part of the system. i.e

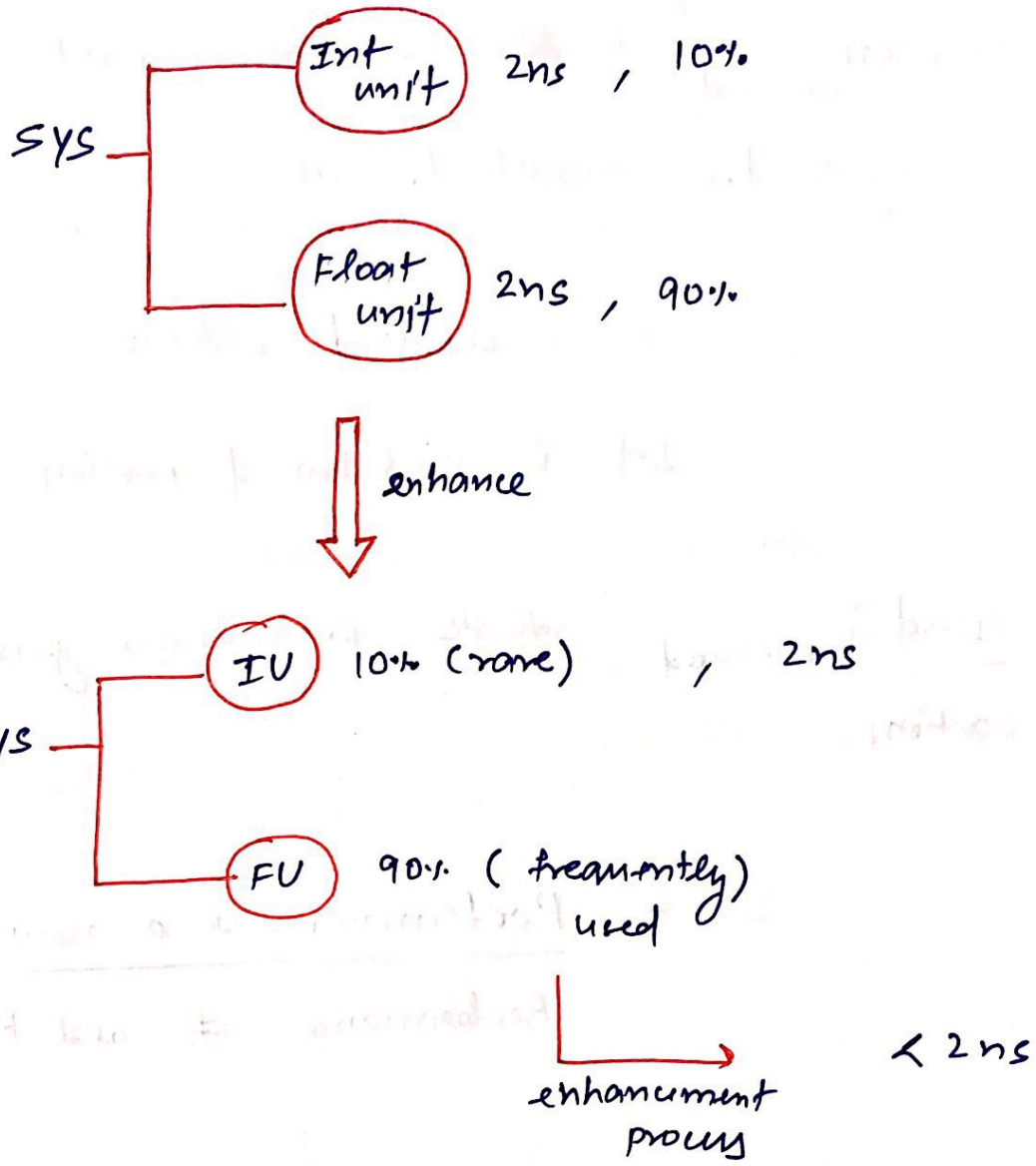
$$S_{overall} = \frac{\text{Performance of sys with enhance (new)}}{\text{Performace of sys without enhance (old)}} \quad \text{---(i)}$$

$$S_{overall} = \frac{ET_{old}}{ET_{new}}$$

- In the enhancement only the part of a system is modified therefore new system contain two portions -

$$ET_{new} = ET \text{ of unenhanced portion} + ET \text{ of an enhanced portion}$$

Eg:-



* so, always frequently used units are enhanced

- Two factors required to calculate ET_{new} :-

  ↳ $Fraction_{enhanced}$ (F)

  ↳ $Speed\ Up_{enhanced}$

- "Fraction enhanced" implies how much part of the system is enhanced or modified. i.e

F is enhanced portion

1-F is unenhanced portion

- Speed Up enhanced indicates performance gain of an enhanced portion.

$$S = \frac{\text{Performance of a new F}}{\text{Performance of old F}}$$

$$S = \frac{\text{ET of old F}}{\text{ET of new F}}$$

$$\text{ET of new F} = \frac{\text{ET of old F}}{S} \quad \text{— (ii)}$$

Substitute eq. (ii) in eq (i) :-

$$\left(\text{ET new} = \text{ET of old } (1-F) + \frac{\text{ET of old}(F)}{S}\right)$$

so,

$$\left[S_{overall} = \frac{ET_{old}}{ET \text{ of old } (1-F) + \frac{ET \text{ of old } (F)}{S}} \right]$$

- To cover the complete system let us consider the relative value i.e

$$\text{System} = 100\% \quad \text{(complete)}$$

$$S_{overall} = \frac{100\%}{(100\% - F) + F/S}$$

$$= \frac{1}{(1-F) + F/S}$$

$$= \left[\left((1-F) + F/S \right)^{-1} \right]$$

when the system contains multiple enhancement then performance gain is calculated as -

$$S_{overall} = \left[(1 - \Sigma f_i) + \Sigma \frac{F_i}{S} \right]$$

i is no. of enhancements

Q. Consider a hypothetical system used in scientific application area. Application program refers the integer and floating point units during its execution. Floating point unit is enhanced then it runs 2 times faster but only 10% of ~~integer~~ instⁿ are floating point. What is the performance gain.

$F = 10\%$ and $S = 2$

So, apply ambdal's law -

$$S_{overall} = \frac{100}{(100-F) + F/S} = \frac{100}{95} = 1.05$$

NOTE :- In the above sys enhancement, rare component is modified so there is no considerable performace gain. Correct enhancement is when we enhance frequentley used components.

$F = 90\%$ and $S = 2$ (Integer unit)

$$S_{overall} = \frac{100}{(100-90) + \frac{90}{2}} = \frac{100}{55} = 1.81$$

Q. Consider cache memory which is 9-times faster than main memory and it is referenced by the CPU 80% of time. what is the performance gain of a system using the cache memory.

$S = 9$ and $F = 80\%$

so, apply ambdal's law -

$$S_{overall} = \frac{100}{(100-F) + F/S} = \frac{100}{(100-80) + \frac{80}{9}}$$

$$= \frac{900}{180 + 80} = \frac{900}{260} = 3.46$$

Q. Consider 2ns clock-cycle processor which consumes 4 cycles per data transfer inst^n, 8 cycles ALU inst^n and 2 cycles for branch inst^n. Relative frequencies of these inst^n are 30%, 60% and 10% respectively. System is enhanced to make the ALU inst^n CPI = 1. what is the performance gain.

$$ET_{old} = \Sigma (IC_i * CPI_i) \, CT$$

$$\text{of inst}^n = (0.3(4) + 0.6(8) + 0.1(2)) \, 2ns$$

$$= 12.4ns$$

$$ET_{new} = \Sigma (IC_i * CPI_i) \, CT$$

$$\text{of inst}^n$$

$$= (0.3(4) + 0.6(1) + 0.1(2)) \, 2ns$$

$$= 4ns$$

$$\left(S = \frac{12.4}{4} = 3.1 \right)$$

Q. 3 enhancements are proposed for a new architecture with respective speedup's ($S_1 = 30$, $S_2 = 20$, $S_3 = 15$). Enhancement 1 and 2 each usable of 25% of time and enhancement 3 is usable for 45% of time. What is overall performance gain S overall.

$F_1 = 25$; $S_1 = 30$

$F_2 = 25$; $S_2 = 20$

$F_3 = 45$; $S_3 = 15$

$$S_{overall} = \frac{100}{(100 - \Sigma f_i) + \Sigma\left(\frac{f_i}{s_i}\right)}$$

$$= \frac{100}{(100 - 95) + \left(\frac{25}{30} + \frac{25}{20} + \frac{45}{15}\right)}$$

$$S_{overall} = \frac{100}{5 + 0.83 + 1.25 + 3} = 9.917$$

High-performanced CPU design:-

- High-performance CPU exhibits the concurrency.
- Concurrency means two or more Inst^n execution at a time.
- According to Flynn's classification computer architecture is of 4 kinds named as:

(i) SISD (Single Inst^n Stream & single Data Stream)

(ii) SIMD (Single Inst^n stream & multiple data stream)

(iii) MISD (Multiple Inst^n Stream & single data stream)

(iv) MIMD (Multiple Inst^n stream & Multiple data stream)

Shortcuts used :-

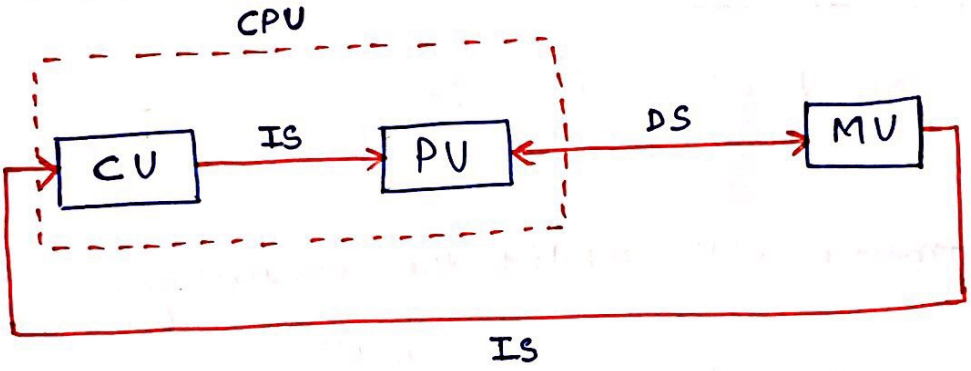
CU : Control unit
PU : Processing unit
} CPU

MU : Memory unit

Is : ~~Input~~ Instn stream

Ds : Data stream

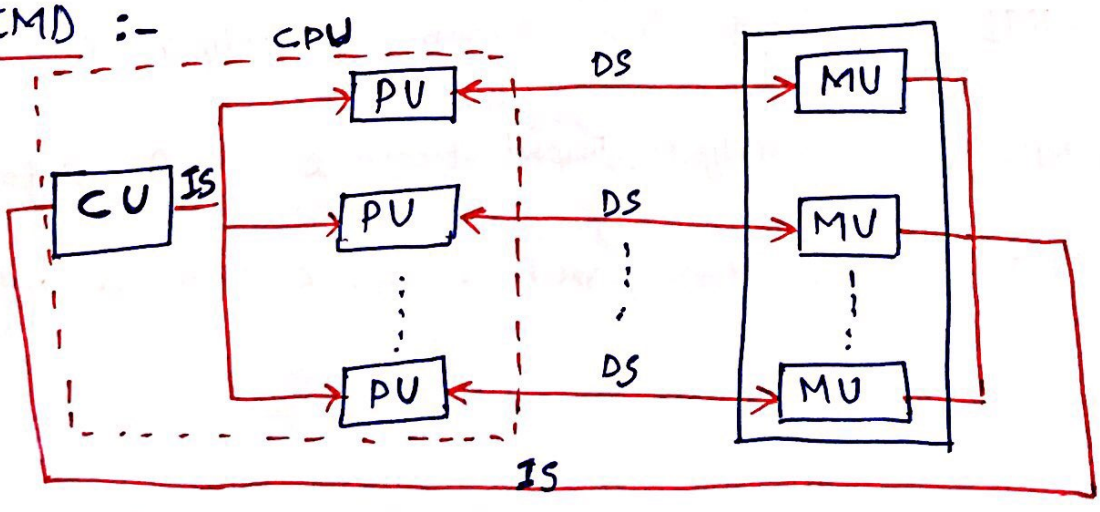
(i) SISD :-



Implemented as an uni-processor system

eg:- 8085 MP - No concurrency
8086 MP

(ii) SIMD :-

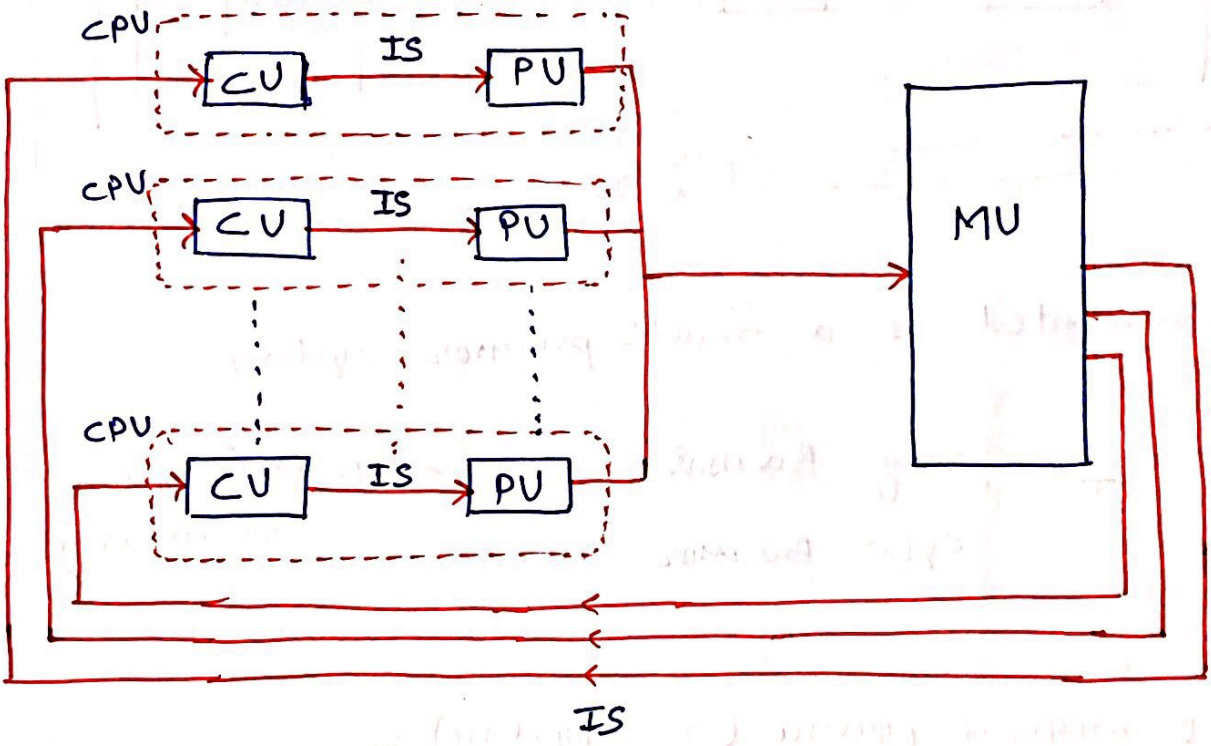


Implemented as an Array processor system (vector processor).

eg:- Storan processor - Data-level concurrency

PEPE processor

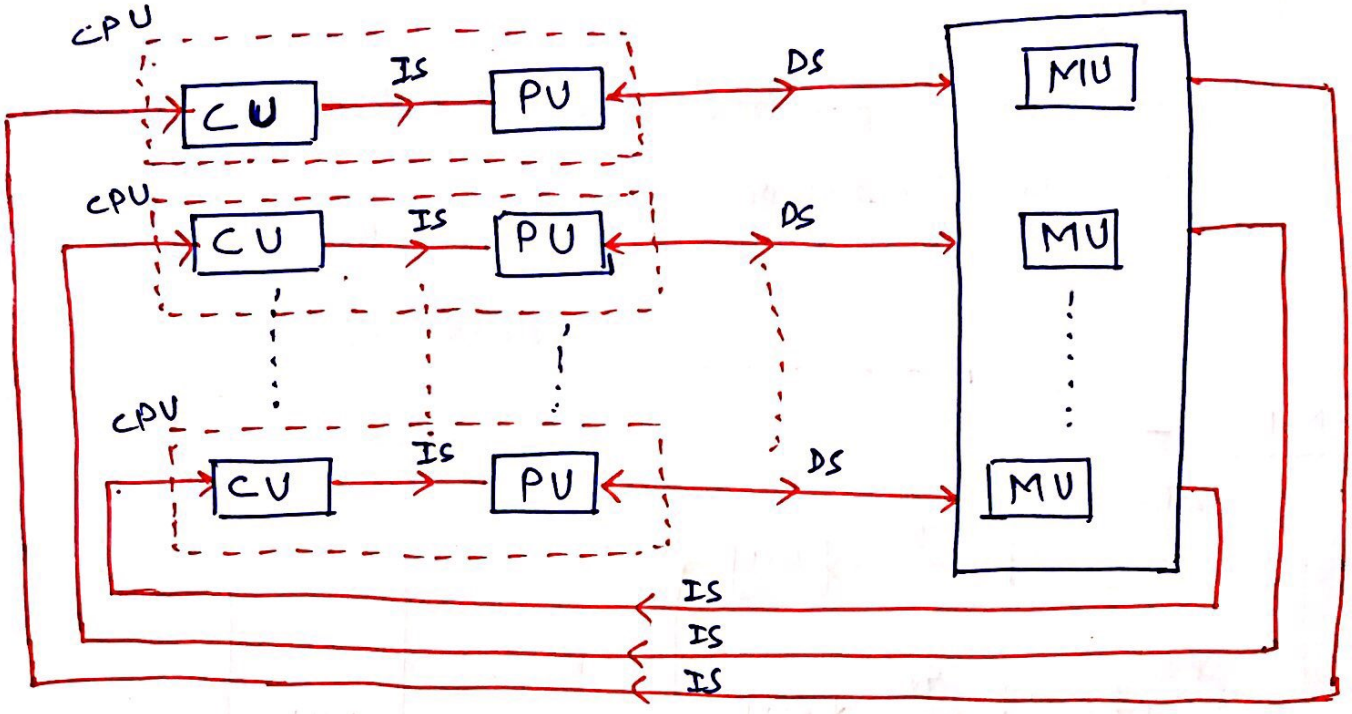
(iii) MISD :-



This architecture contains multiprocessors but only one is in the use at a time as Data stream is single, only one CPU can communicate with MV.

So, this Architecture is not yet Implemented

(iv) MIMD :-



Implemented as a multi-processor system

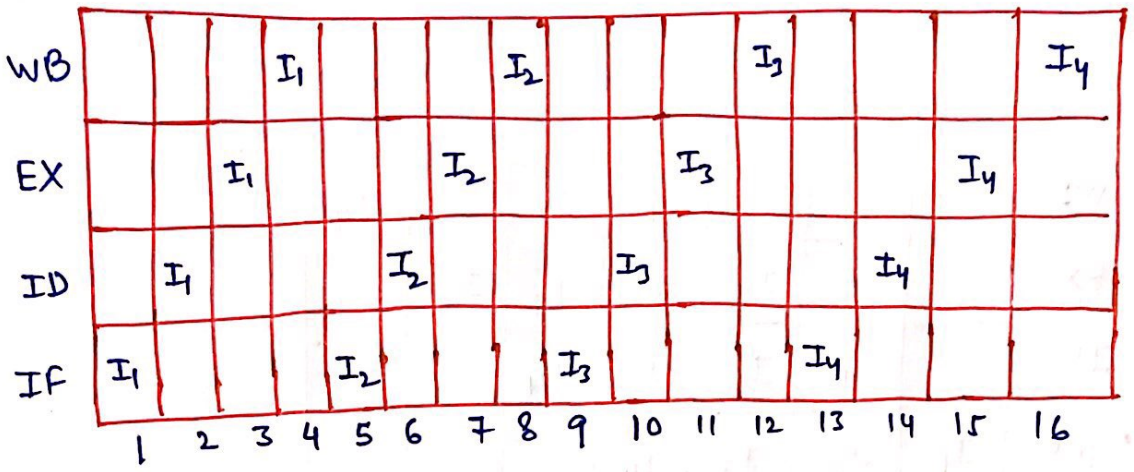
Eg:- Cray Processor - Instⁿ level concurrency
Cyber Processor

SISD without pipeline (non-pipeline) :-

Instⁿ execution states : IF ID EX WB

IF - MEM
ID - DECODER
EX - ALU
WB - REG

Execution seq of instⁿ's :-



Instⁿ	Required cycles	Counted cycles
I_1	4	4
I_2	4	4
I_3	4	4
I_4	4	4

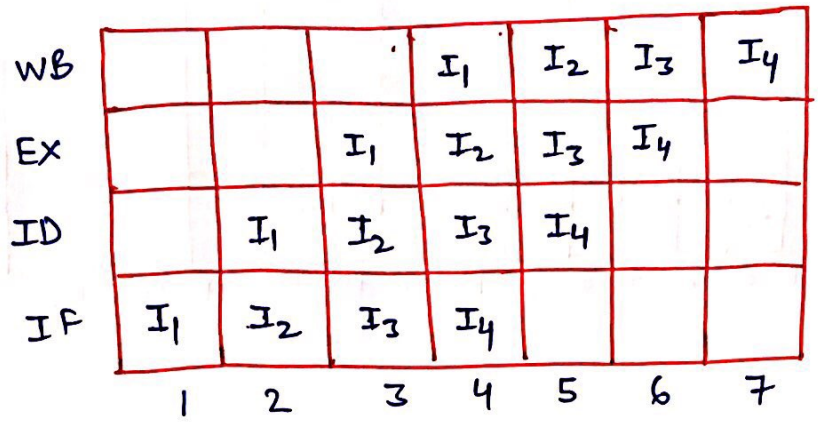
Total cycles = 16

SISD with pipeline :-

Execution stages : IF , ID , EX , WB

Pipeline Design :

Execution sequence of 4 Inst^n (I_1, I_2, I_3, I_4) –



Inst^n	Required cycles	Counted cycles
I_1	4	4
I_2	4	1
I_3	4	1
I_4	4	1

Total cycles = 7

Generalization { k – stage pipeline, n – tasks } (Total cycle = k + n – 1)

Task	Req-cycles	counted-cycles
T_1	k	k
T_2	k	1
⋮	⋮	⋮
T_n	k	1

Pipelining :-

- Function - Pipelining uses decomposition ~~process~~ techniane i.e divide the problem statement into independent sub-problems, assign them to a independent hardware. Later connect the hardware in a pipeline sequence i.e

(First H/w) $\xrightarrow{O/p \quad I/p}$ (Second H/w)

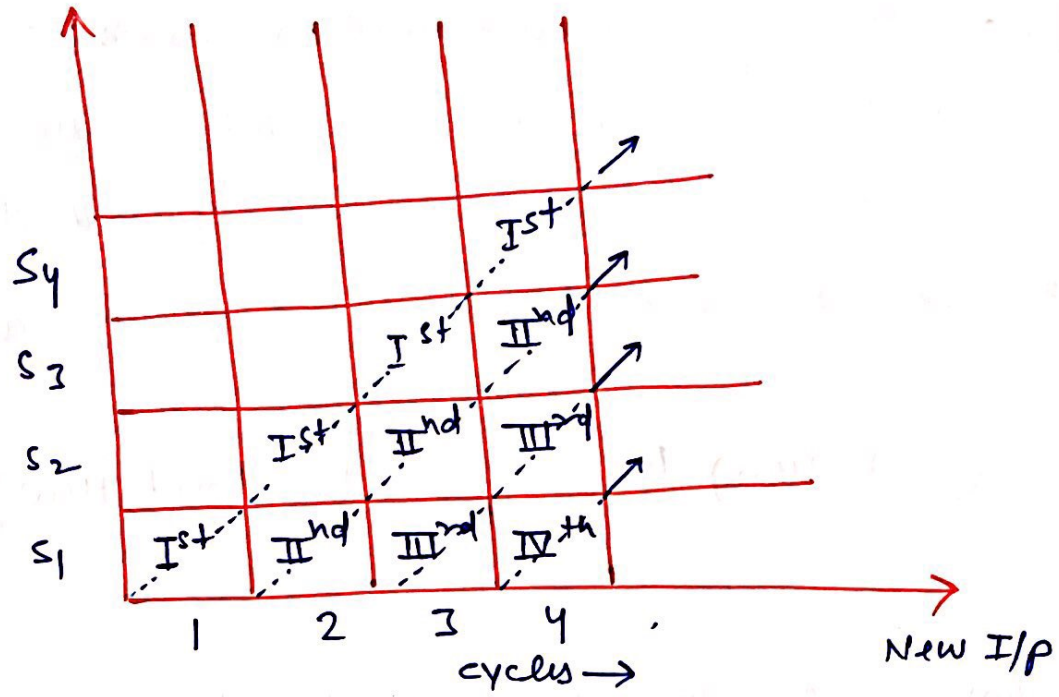
- Defination - Accepting the new I/p at one end before the previously accepted I/p is appears as an O/p at the other end.

- Defination states that, insert new I/p into a pipeline before completion of a old I/p.

- Pipline allows overlapping execution (new I/p overlapping with old I/p)

- Overlapping execution sequence in the pipeline is described using space - time diagram.

Space - Time Diagram



- Successful characteristic of a pipeline is, at every new cycle new I/p must be inserted into a pipeline

∴ CPI = 1.

NOTE:- In a non-pipline sys new I/p is inserted after complition of old I/p called as non-overlapping execution.

Design:-

- Pipleline contain two ends (I/p and o/p end) b/w these ends multiple pipes are interconnected to satisfy the functionality of a pipeline. Each piping in the

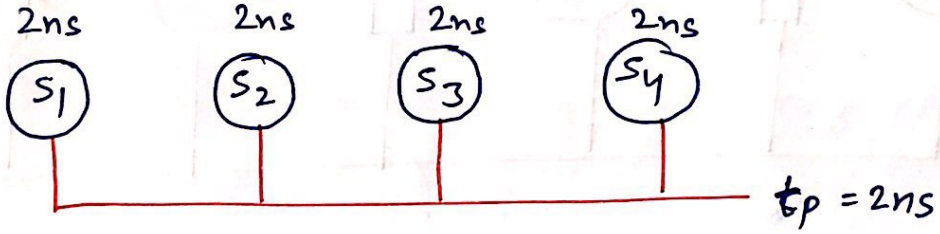
pipeline is known as stages.

- Interface registers are used b/w stages to hold the intermediate result. It is also called as Buffer, or latch or pipeline register.

- All the stages in the pipeline along with the buffer is connected to a common clock. So clock adjustment is very important in the pipeline design.

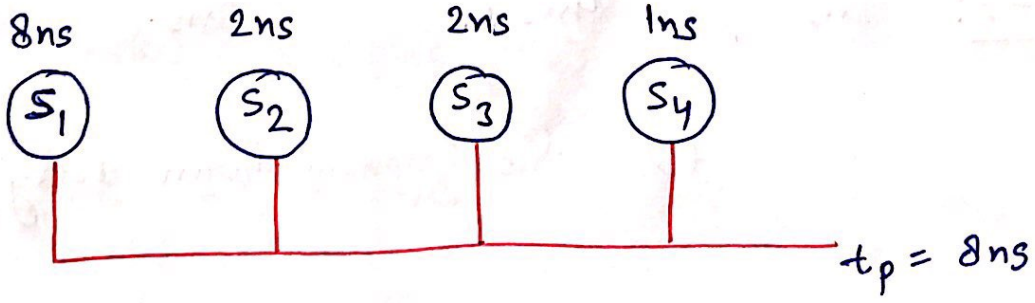
(a) in uniform - delay pipeline -

(cycle time (tp) = stage Delay)



(b) in non-uniform - delay pipeline -

(cycle time (tp) = max-stage Delay)



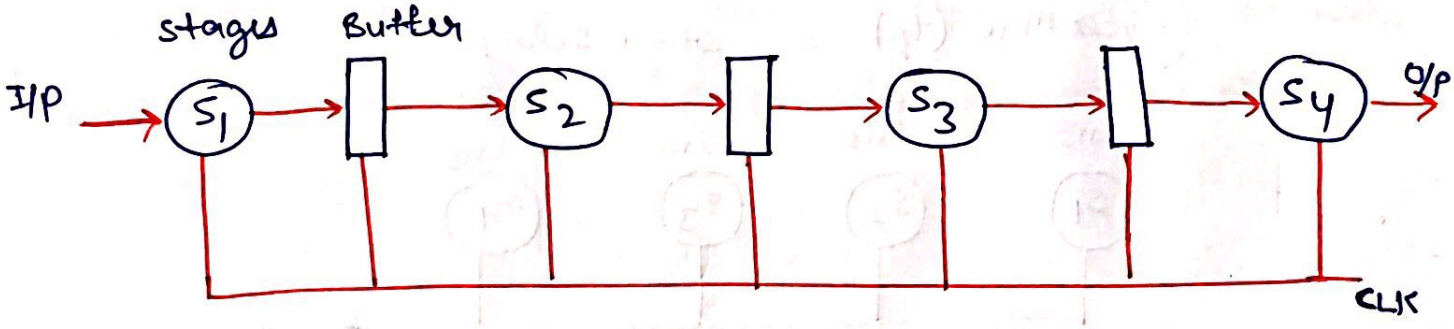
(c) if buffer - delay present in the pipeline -

$$\left(t_p = \max\left(\text{stage delay} + \text{buffer delay}\right)\right)$$

(d) If skew time / setup time overhead is present then,

$$\left(t_p = t_p + \text{overhead}\right)$$

extra - delay at run-time



CASE-1 :. 2ns 2ns 2ns 2ns

t_p = 2ns (uniform delay)

CASE - 2 : 2ns 4ns 8ns 3ns

t_p = 8ns (non-uniform delay)

CASE-3 Buffer delay = 1ns

t_p = 3ns uniform delay with buffer delay

t_p = 9ns non uniform delay with "

CASE-4 Skew time overhead is 0.2ns

t_p = 3.2ns uniform delay + Buffer delay + Skew time

t_p = 9.2ns non uniform + " + "

Performance Analysis :-

- Consider 'k-segment' pipeline with cycle time of 'tp" used to execute 'n' tasks.

- In the pipeline, very first task is executed in a non-overlapping order. so it takes k-cycles. The remaining n-1 task are executed in overlapping order i.e emerging out at the rate of 1-task / cycle so n-1 task takes n-1 cycles.

Thus total time to complete n-tasks by K-stage pipeline,

$$ET_{pipe} = (K + n - 1) \text{ cycles}$$

- Consider non-pipeline procedure used to execute 'n' tasks in which each task takes t_n time to complete ~~one task~~ therefore total time to execute n tasks

so,

$$\left(ET_{non\text{-}pipe} = n * t_n \right)$$

- Performance gain of a pipeline is -

$$S = \frac{ET_{non\text{-}pipe}}{ET_{pipe}}$$

$$\left(S = \frac{n \cdot t_n}{(k + (n-1)) t_p} \right) \quad ; \quad n \text{ is limited}$$

when no. of task (n) increases then $n >> k$

so, $k+n-1 \longrightarrow n$, under this condition

$$S = \frac{n \cdot t_n}{n \cdot t_p}$$

$$\left(S = \frac{t_n}{t_p} \right) \quad ; \quad n \text{ is infinity}$$

t_n = ~~time of~~ one-task ET in non-pipe

t_p = cycle-time

- when the pipeline stages are perfectly balanced (uniform-delay) then 1-task execution time in the non-pipeline is also equal to no. of stages in the pipeline

i.e $\begin{bmatrix} t_n = K\text{-cycles} \\ t_n = K \cdot t_p \end{bmatrix}$

Under this condition $S = \frac{K \cancel{t_p}}{\cancel{t_p}}$

$S = K$ i.e $\eta = 100\%$

K is pipeline depth.

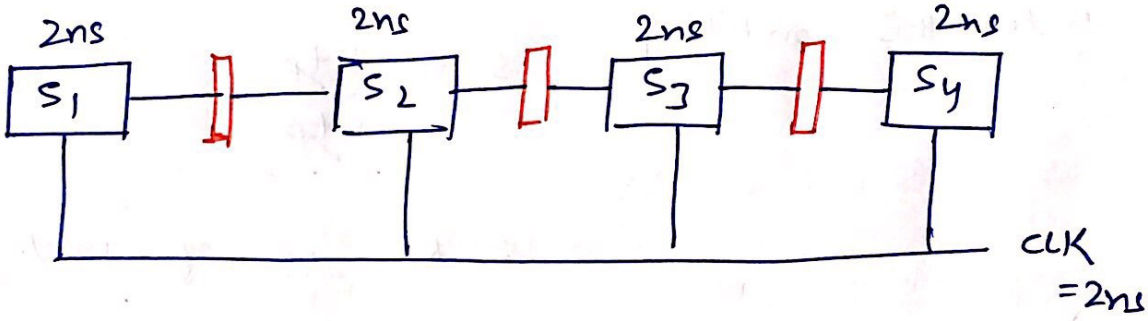
- when the system is operating with $\eta = 100\%$ then maximum speed-up is possible i.e equal to K (pipeline-depth)

$$\eta_{pipe} = \frac{S}{S_{max}} = \frac{S}{K}$$

$$\text{Throughput}_{\text{pipe}} = \frac{\text{tasks processed}}{\text{Total time to process the task}}$$

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

NOTE :- When the pipeline stages are pertectly balanced (uniform delay) then one task execution time in the pipe line is also equal to 1 task execution time in the non-pipeline.



(a) 1 - Task ET in pipeline

$k = 4$

$t_p = 2ns$

$n = 1$

$$ET_{pipe} = (k + n - 1)\,t_p$$

$$= 4 + 1 - 1 = 4t_p$$

$$= \underline{8ns}$$

(b) 1-task ET in non-pipeline (t_n)

$$t_n = S_1 + S_2 + S_3 + S_4$$

$$= 8ns$$

$$S = \frac{t_n}{t_p} \quad \text{when} \quad n \to \infty$$

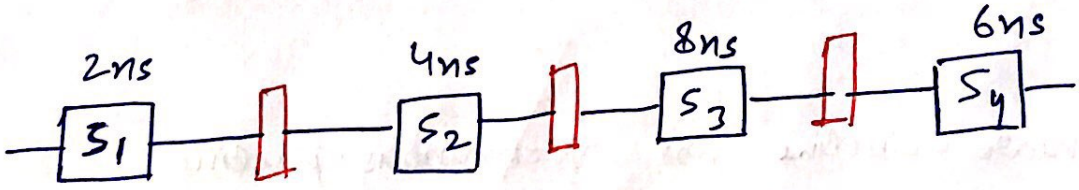
$$\eta = S/K = K/K = 1$$

$$S = \frac{8}{2} = 4 \text{ i.e } K$$

So, for $n \to \infty$ pipeline achieves 100% efficiency.

NOTE:- When pipeline stages are not balanced (non-uniform delay). Then 1-task execution time in the pipeline is greater than 1-task ET in non-pipeline.

(a) 1-tast ET in pipeline -



$K = 4$

$tp = 8ns$

$n = 1$

$$ET_{pipe} = (K + n-1)\, tp$$

$$= (4)\, tp$$

$$= 32ns$$

(b) 1 task ET in non-pipe (t_n)

$$t_n = s_1 + s_2 + s_3 + s_4$$

$$= 20ns$$

$$S = \frac{t_n}{t_p} \; ; \; n \to \infty \qquad s = \frac{20}{8} \qquad S = 2.5 \neq K$$

$$\eta = \frac{S}{K} = \frac{2.5}{4} = 62.5\%$$

NOTE :- For a non-~~linear~~ uniform delay pipe η can't be 100%.

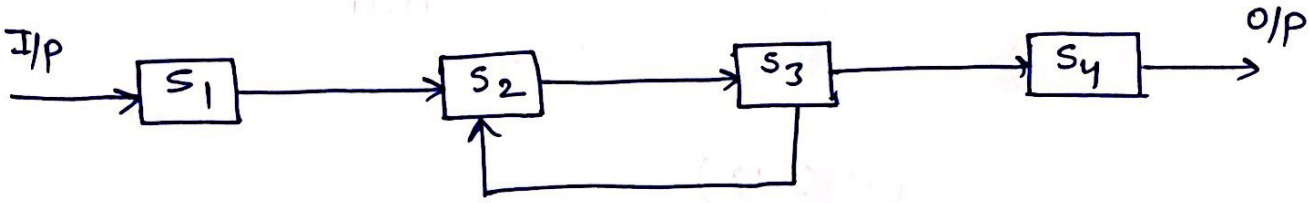
Types of Pipe-line :-

(i) Linear-pipe-line - This pipeline contains only forward connection. So latency is always 1. Latency means clock-cycle difference b/w two successive initiations in the pipeline.

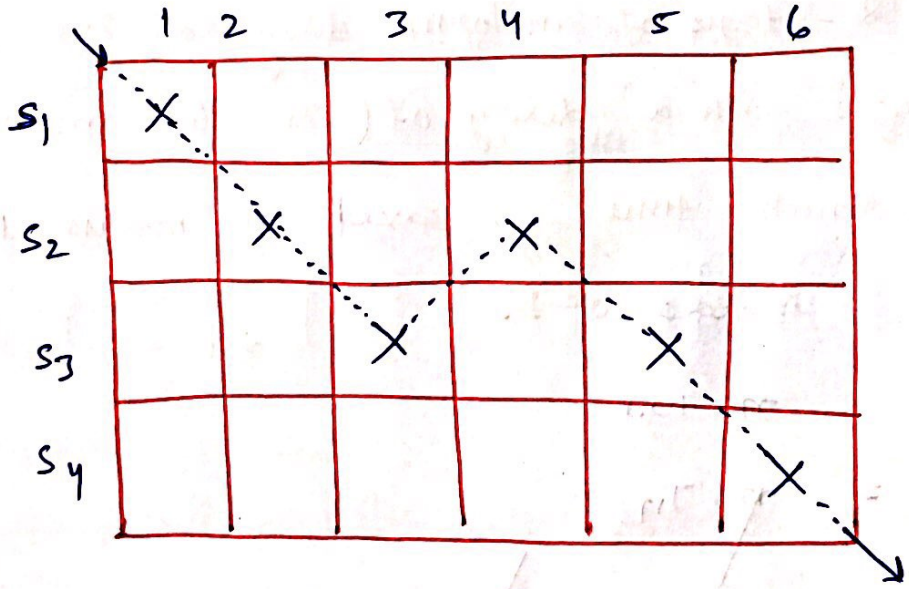
Linear-pipeline are synchronous pipeline.

(ii) **Non-Linear pipeline :-** This pipeline contains forward and backward connection so reservation table is required to process the I/p. Therefore Latency depends on the reservation table.

They are asynchronous pipeline.



Reservation - Table



$(S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4)$

Q. Consider ist^n pipeline with a speedup of 8 operating with 72% efficiency. what could be the no. of stages in the pipeline.

$$\eta = \frac{S}{K} \quad ; \qquad S = 8 \quad \text{and} \quad \eta = 0.72$$

$$\text{so,} \quad K = \frac{S}{\eta} = \frac{8}{0.72} = 11.11$$

$$(K \cong 12)$$

Q. Consider two pipelines A and B where A is having 8-stages of uniform delay of 2ns. B having 6-stages with a delays of (2ns, 6ns, 8ns, 4ns, 3ns, 1ns). How much time is saved to prous 100 task using A instead of B.

$$n = 100$$

$$ET_B = n \cdot Tn$$

$$= n(2+6+8+4+3+1)$$

$$= 100(24)$$

$$= 2400 \text{ ns}$$

$$ET_A = (K+n-1)\,T_p$$

$$= (8+100-1)\,2ns$$

$$= 107(2)\ ns$$

$$= \underline{214ns}$$

$$ET_B = (K+n-1)\,T_p$$

$$= (6+100-1)\,8$$

$$= 105(8)\ ns$$

$$= \underline{840\ ns}$$

$$\text{Time saved} = ET_B - ET_A$$

$$= 840 - 214$$

$$= \underline{(626ns)}$$

Q. Consider 4- stage pipeline with a respective delays of (20ns, 40ns, 30ns, 10ns). What is performance gain and efficiency.

$n \rightarrow \infty$, $K = 4$

$t_p = 40ns$, $t_n = 100ns$

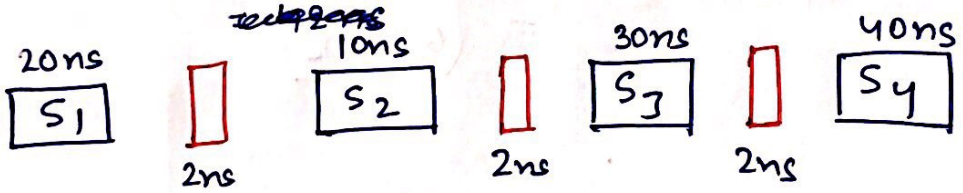
Speed performance gain $S = \frac{t_n}{t_p} = \frac{100}{40}$

$$S = 2.5$$

$$\eta \text{ (efficiency)} = \frac{S}{K} = \frac{2.5}{4} = 62.5\%$$

Q. Consider 4-stage pipeline with a respective delays of (20ns, 10ns, 30ns, 40ns). Interface register used b/w stages having delay of 2ns. What is performance gain of pipeline when a large no. of task are executed in pipeline).

$\eta \to \infty$, $t_p = 40$ ns , $t_{IR} = 2$ns , $t_n = 100$ns



t_p = max (stage delay + Buffer delay)

= max (22, 12, 32, 40)

no, Interface register after last stage

$(t_n = 20 + 10 + 30 + 40)$ Interface register are not present in non-pipeline.

$$S = \frac{t_n}{t_p} = \frac{100}{40} = 2.5$$

Q. Consider 3.2 ns clock cycle processor which consumes 6-cycles per data-transfer inst^n. 8-cycles per ALU inst^n and 4-cycles for branch inst^n. Relative frequencies of these inst^n (50%, 25%, 25%). System is enhanced with a pipeline. In this implementation 0.6 ns overhead is present. What is the performance gain.

$$ET_{non\text{-}pipe} = \Sigma (IC_i * CPI_i) * CT$$

$$= [0.5(6) + 0.25(8) + 0.25(4)] * 3.2 ns$$

$$= 19.2 ns$$

$$ET_{pipe} = \Sigma (IC_i * CPI_i) * CT$$

$$= (0.5 + 0.25 + 0.25)(3.2 + 0.6)$$

$$= 3.8 ns$$

$$\text{Performance gain} = \frac{ET_{non\text{-}pipe}}{ET_{pipe}} = \frac{19.2}{3.8}$$

$$[S = 5.05]$$

Q. Consider 4-stage pipeline where different instⁿ are spending different cycles, at different stages.

	S_1	S_2	S_3	S_4
I_1	1	3	2	1
I_2	1	1	3	2
I_3	1	3	2	1
I_4	1	1	2	1

(a) How many cycles required to complete the code execution?

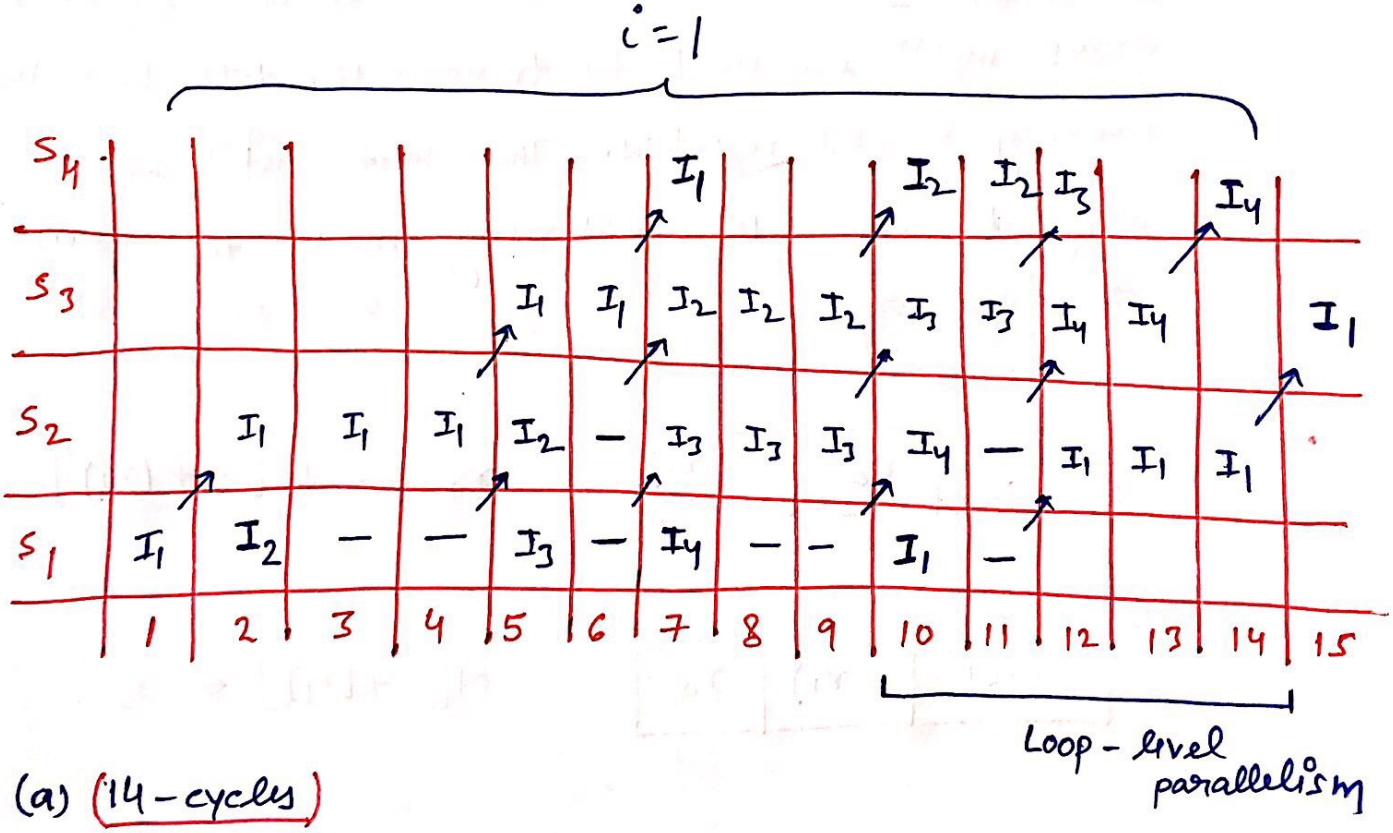
(b) Speed up factor?

(c) Following loop is executed in the pipeline -

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

The o/p of I_2 for (i=2) will be available after ____ cycles.

Since this pipeline doesn't have uniform or non-uniform delay cases so we have to procede using reservation table –



(a) (14-cycles)

(b) $S = \frac{ET_{non-pipe}}{ET_{pipe}} = \frac{(I_1+I_2+I_3+I_4)}{14} = \frac{7+7+7+5}{14}$

$(S = 1.85)$

(c) 21 - cycle

RISC pipeline :-

- To analyse the implementation issues of a pipeline, let us consider RISC pipeline as a refrence modal.

- RISC CPU supports inst^n pipeline, used to execute the inst^n.

- RISC CPU inst^n set is -

(i) Data transfer inst^n : In the RISC CPU, LOAD & STORE inst^n are used to transfer the data b/w the memory & CPU registers. These ~~sinst~~ inst^n are designed with index addressing mode to access the memory.

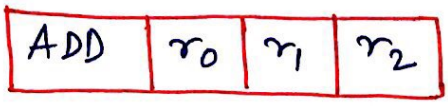
LOAD	r_0	$3(r_1)$

$r_0 \leftarrow M[3 + [r_1]]$

STORE	$3(r_1)$	r_0

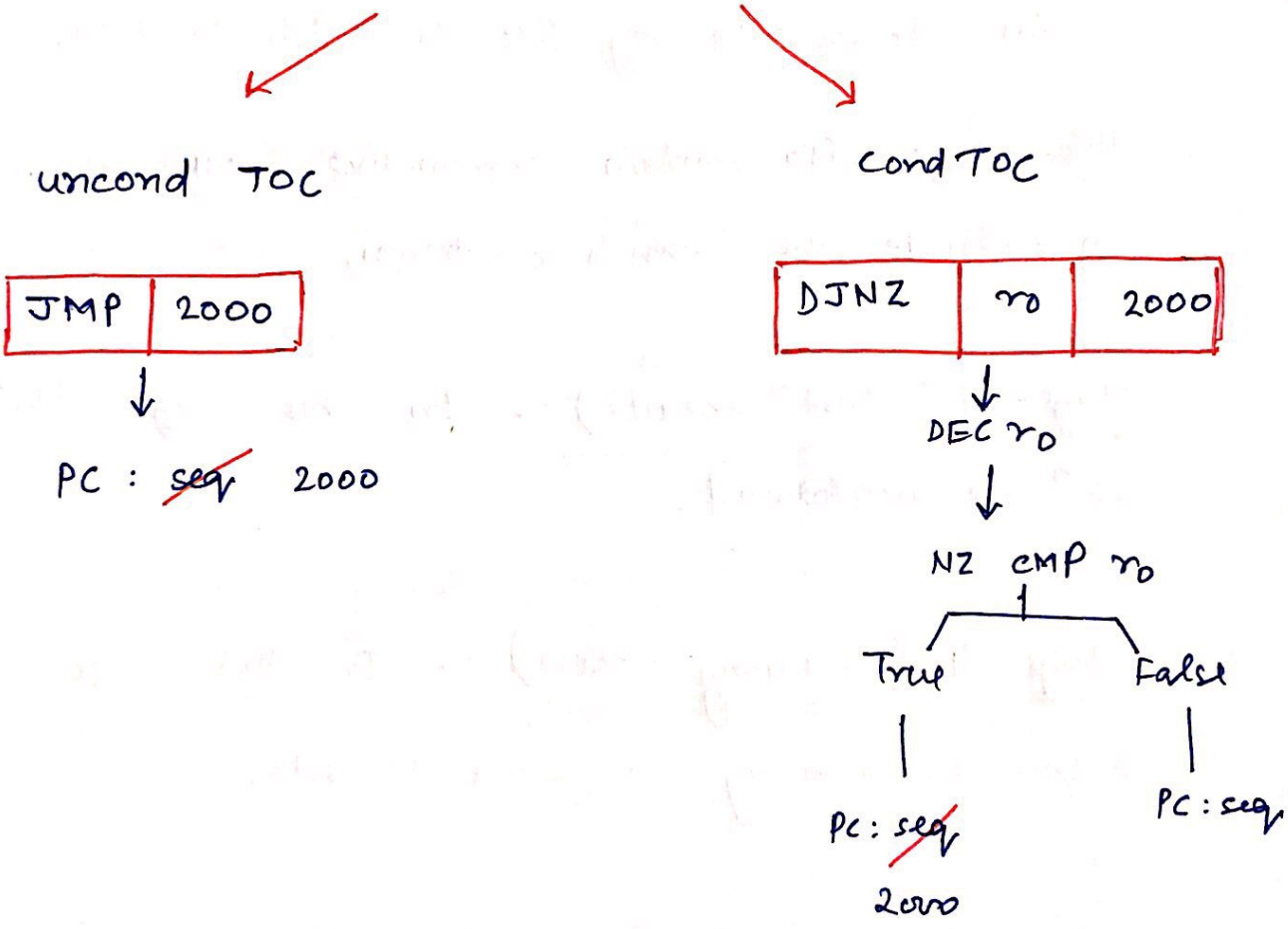
$M[3 + [r_1]] \leftarrow r_0$

(ii) Data manipulation inst^n : In the RISC CPU, ALU op^n are performed only on a register data so it only follow register addressing modes.



$r_0 \leftarrow r_1 + r_2$

(iii) Transfer of control inst^n : They contain two types unconditional & conditional Jump inst^n.



- To Execute above inst^n, 5-stage pipeline is used.

- Different stages present in the pipeline are as follows :

Stage - 1 (Inst^n fetch) :- In this phase CPU phase the inst^n from the memory.

Stage - 2 (Inst^n decode) :- In this stage 2 op^n are performed.

(i) Decode the inst^n

(ii) Access the reg file to fetch the data.

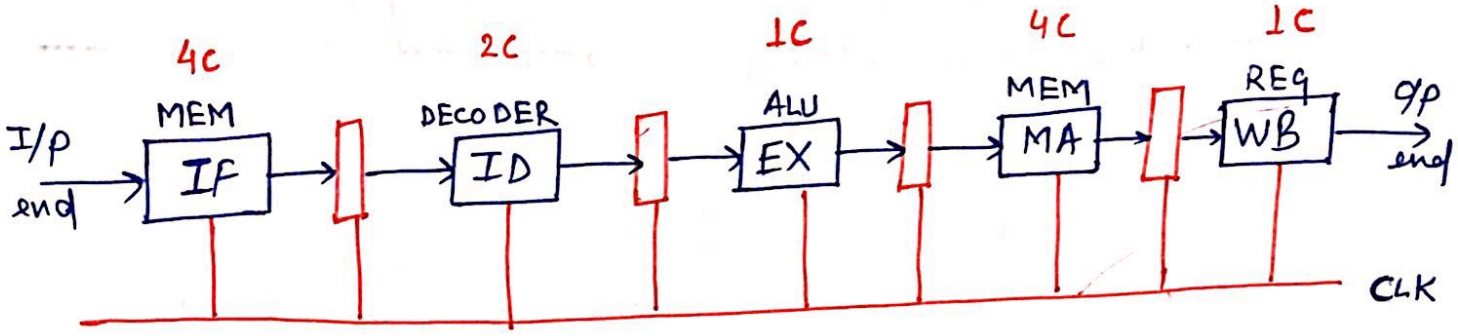
This stage also contain comparitive (CMP) circuit to evaluate the branch condition.

Stage - 3 (Inst^n execute) :- In this stage ALU op^n are performed.

Stage - 4 (Memory access) :- In this stage CPU refers the memory to access the data.

Stage - 5 (Write Back) :- In this stage register WRITE op^n is performed.

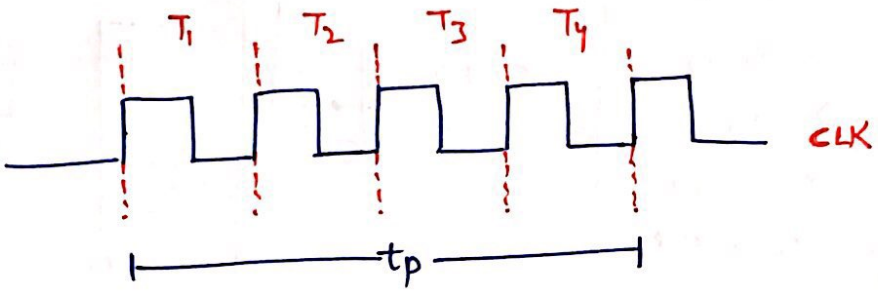
- Above stage inst^n are assigned to an independent H/W to form the 5-stage pipeline.



$$t_p = \max(S_1, S_2, S_3, S_4, S_5)$$

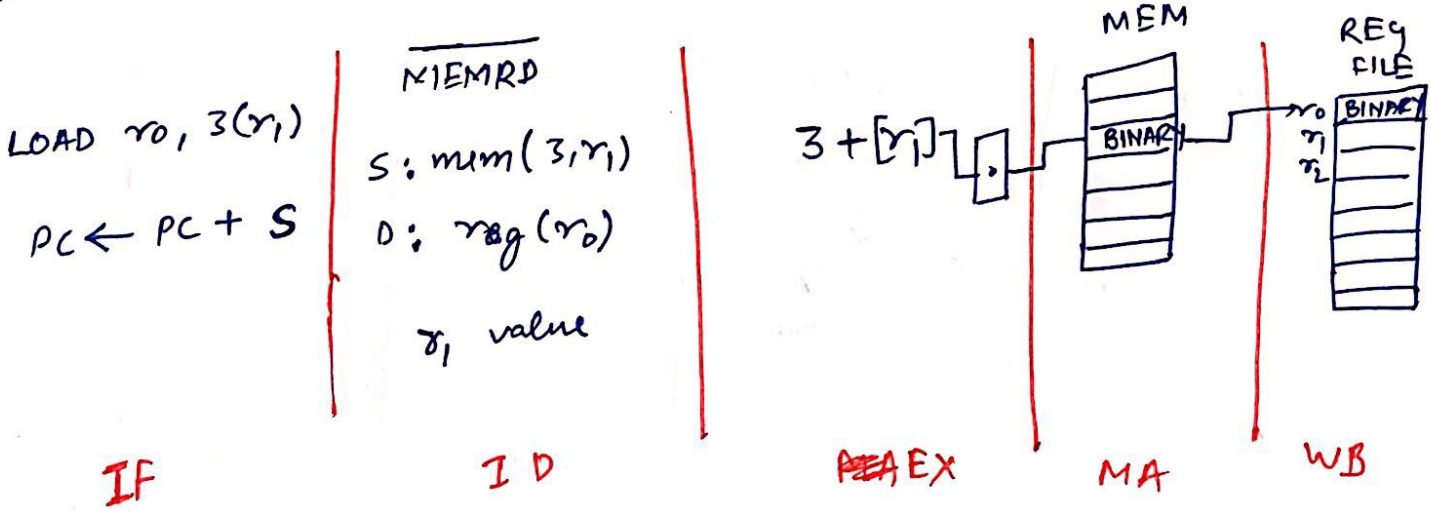
$$= \max(4, 2, 1, 4, 1)$$

$$= 4 \text{ cycles}$$

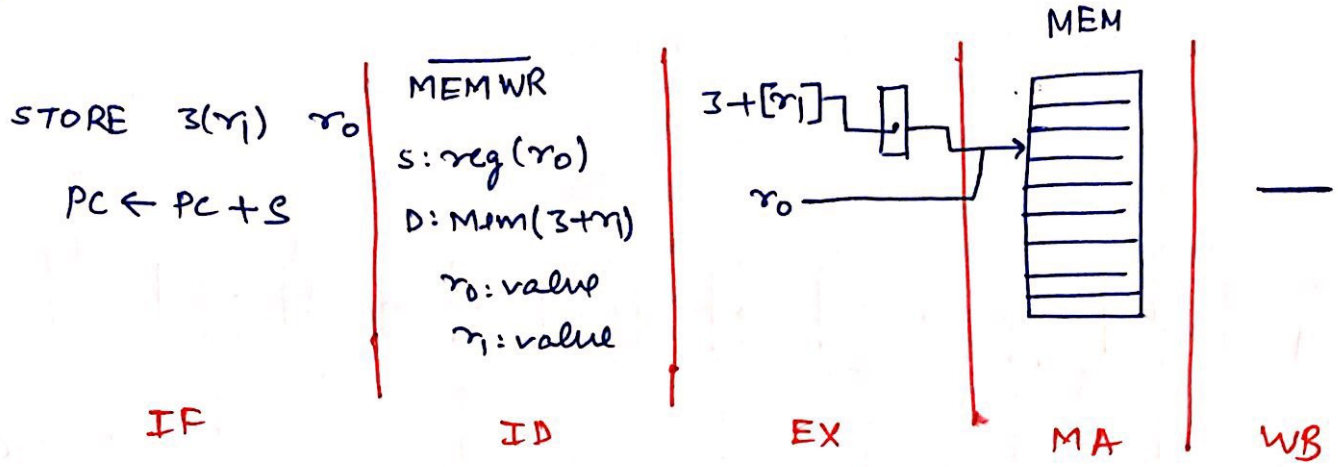


- Data ~~manipulation~~ transfer inst^n implementation :-

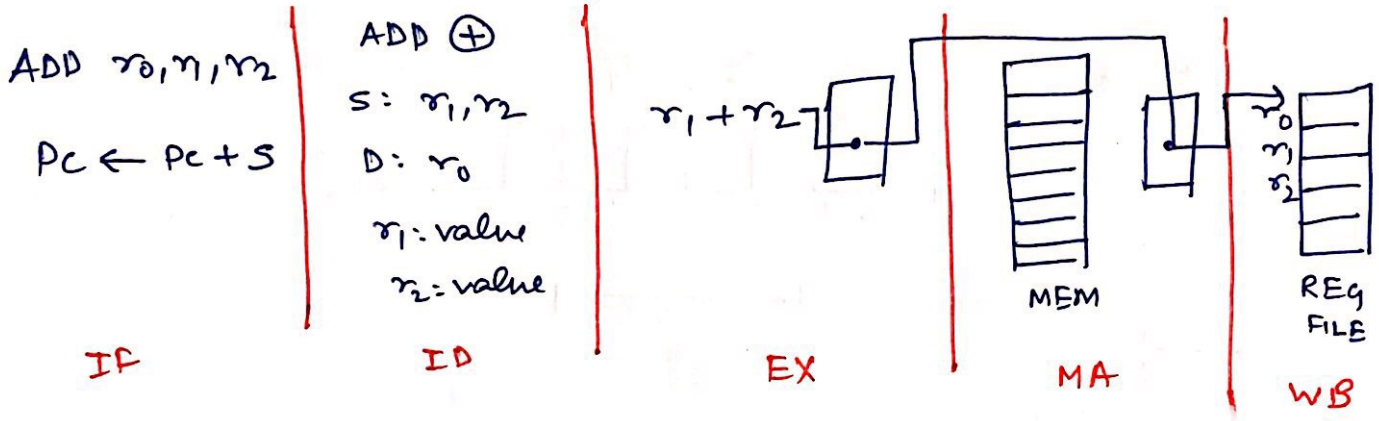
(i) LOAD inst^n



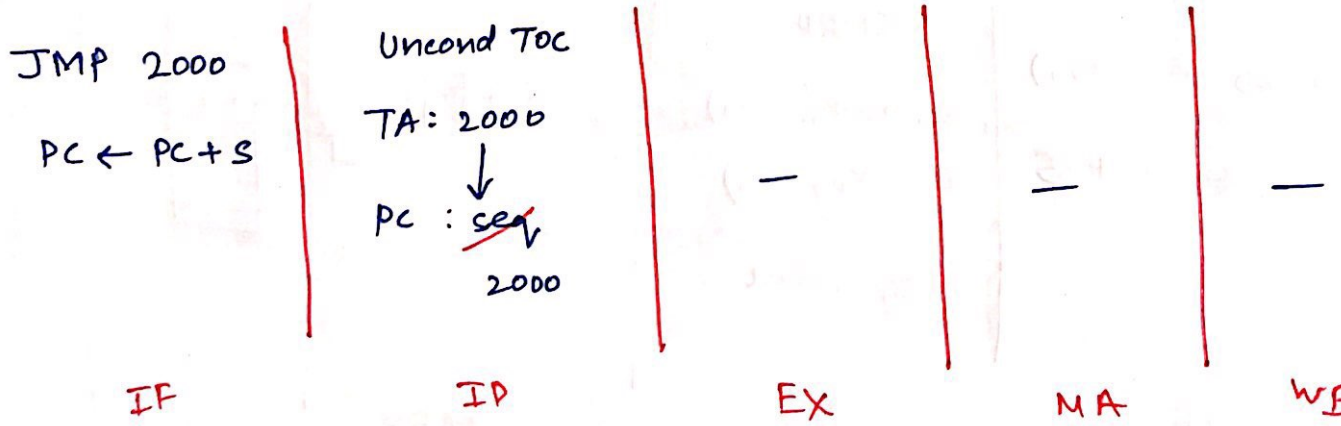
(ii) STORE inst^n

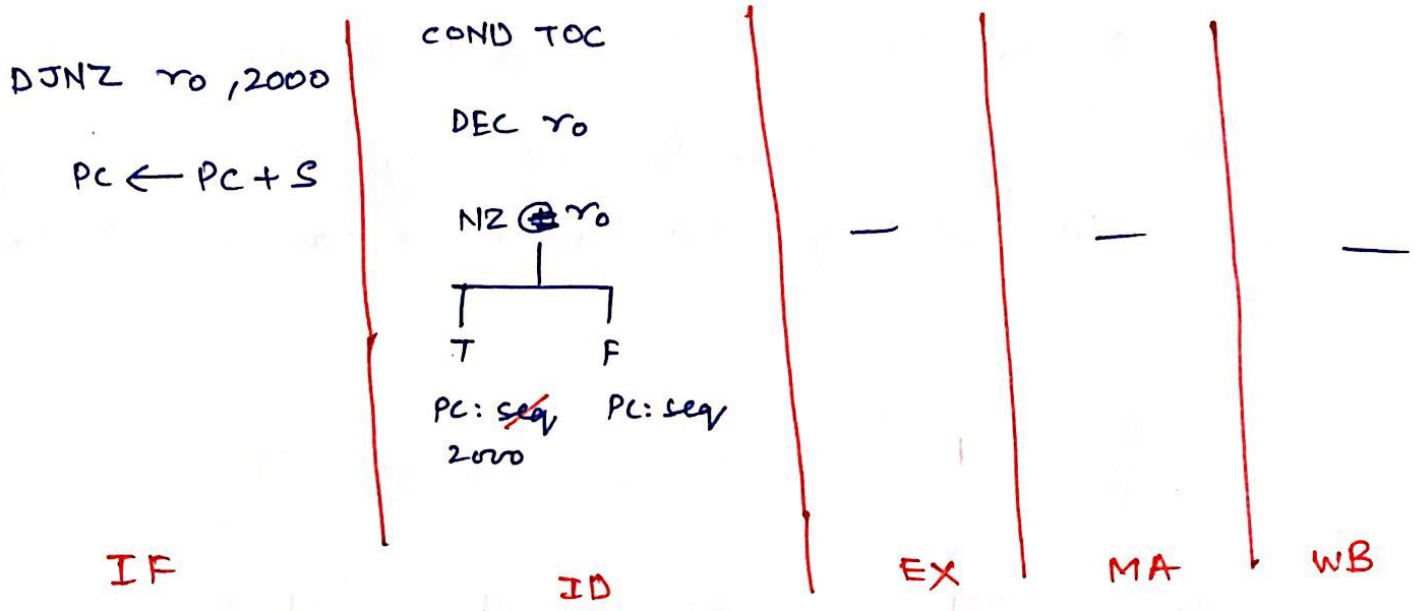


- Data manipulation inst^n implimentation :-



- TOC inst^n implimentation :-

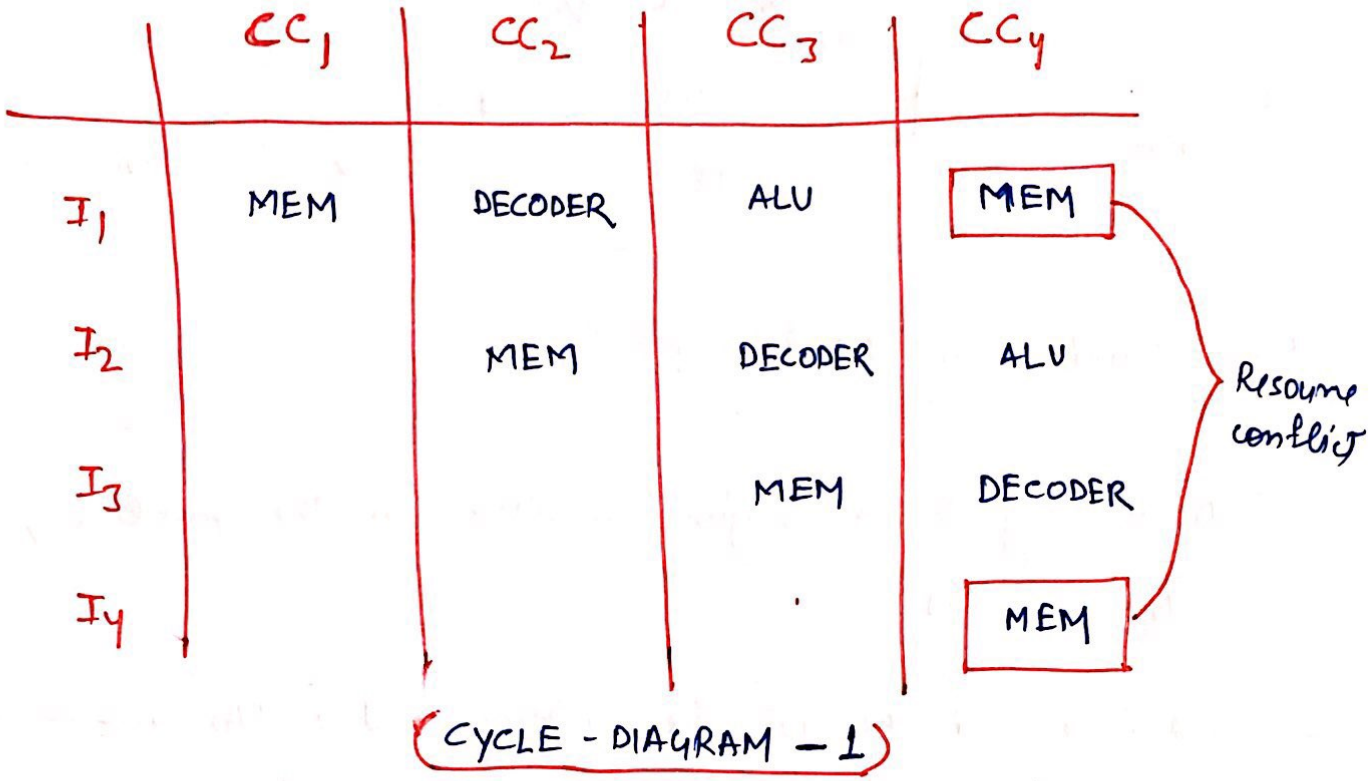




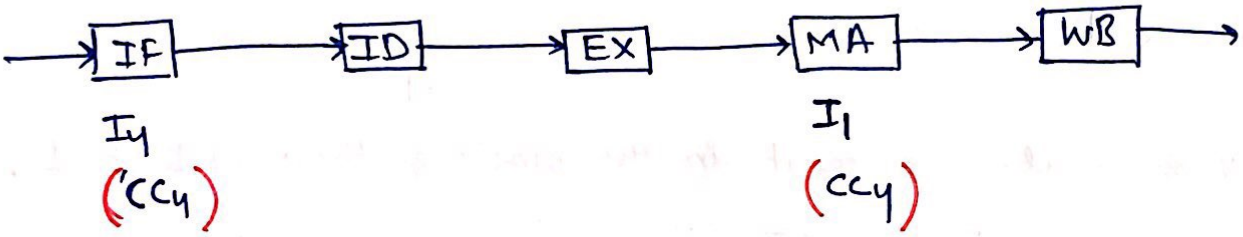
Dependencies in Pipeline :-

- Dependency is a major problem in the pipeline, causes extra-cycles.
- a cycle in the pipeline without new I/p initiation is called as extra-cycle, Also named as stall or hazard.
- When stall is present in the pipeline then CPI ≠ 1.
- 3 types of dependencies are present in the pipeline names as:
  - (i) structural dependency
  - (ii) Data - dependency
  - (iii) Control - dependency

(i) Structural dependency :- It occured in the pipeline due to resource conflict. Resource may be a memory or register or any other functional unit (ALU).



CYCLE - DIAGRAM - 1



In the above exec. I_1 & I_y both instn are accessing the same resource at same ~~cycle~~ time (i.e CC_y). This situation in the pipeline is ~~used~~ known as resource conflict.

- Conflict is an unsuccessful op^n so we need to keep the inst^n I_4 into waiting state until resource become available. This waiting creates stalls in the pipeline.

	CC_1	CC_2	CC_3	CC_4	CC_5	CC_6	CC_7
I_1	MEM	DECODER	ALU	MEM	WB (reg)		
I_2		MEM	DECODER	ALU	MEM	WB	
I_3			MEM	DECODER	ALU	MEM	WB
I_4							MEM

(CYCLE-DIAGRAM-2)

Structural stall or Hazard

(7-cycles − 4 I/p's = 3 stalls)

- To minimize the structural hazards, H/W mechanism is used i.e renaming.

- In cycle-diagram 1 I_1 refer the memory in 4th stage to access the data, whereas I_4 refers the memory to access the inst^n simuntaneously.

- When the inst^n & data both are present in the same memory then the above ~~suit~~ situation creates conflicts.

- Renaming mechanism states that organize memory into two independent modules to store the data and inst^n seperately known as ~~eo~~. data memory (DM) and code memory (CM).

- Now I_1 refer DM and I_4 refer CM at CC_4, and no conflict happens.

	CC_1	CC_2	CC_3	CC_4	CC_5	CC_6	CC_7
I_1	CM	ID	ALU	DM	WB		
I_2		CM	ID	ALU	DM	WB	
I_3			CM	ID	ALU	DM	WB
I_4				CM	ID	ALU	DM
I_5					CM	ID	ALU
I_6						CM	ID
I_7							CM

(CYCLE – DIAGRAM – 3)

(7-cycles − 7 I/p's = 0 stalls)

(ii) Data dependencies :-

- Consider the program segment where Inst^n 'J' follows Inst^n 'I' in a program order.

Prog:-

I:	Inst^n
J:	Inst^n
⋮	⋮

Data-dependency condition will be occurred when Inst^n 'J' tries to read the data before Inst^n 'I' write it.

(READ — before — WRITE)

Eg:- ADD r_0, r_1, r_2 ; $r_0 \leftarrow r_1 + r_2$ (r_0 is Dest)

MUL r_3, r_0, r_2 ; $r_3 \leftarrow r_0 * r_2$ (r_0 is Source)

NOTE: When the above inst^n are executed in the non-pipeline system then data dependency condition doesn't occurred because I_2 inst^n executes after ~~comp~~ execution I_1. i.e (READ - After - WRITE)

- When the above inst^n are executed in the pipeline system then data-dependency condition will be occurred i.e I_2 starts executing along with the I_1. So I_2 reads the r_0 data register before I_1 writes it. Therefore I_2 incorrectly access the old value from the register r_0 (data loss).