PIPELINE -

⇒ The speed of execution of program is influenced by many factors. One way to improve performance is to use faster circuit technology to build the processor and the main memory.

⇒ Another possibility, is to arrange the hardware so that more than one operation can be performed at the same time.

⇒ Pipelining is a particularly effective way of organizing concurrent activity in a computer system. Pipelining is commonly known as an assembly-line operation.

⇒ A pipeline can be visualized as a collection of processing segments through which binary information flows.

⇒ There are two areas of computer design where the pipeline organization is applicable.

1. An Arithmetic Pipeline:- It divides an arithmetic operation into sub-operations for execution in the pipeline segments.

2. An Instruction Pipeline :- It operates on a stream of instructions by overlapping the fetch, decode and execute phases of the Instruction cycle.

Function of the Pipeline - one pipe output is connected as input to the another pipe.

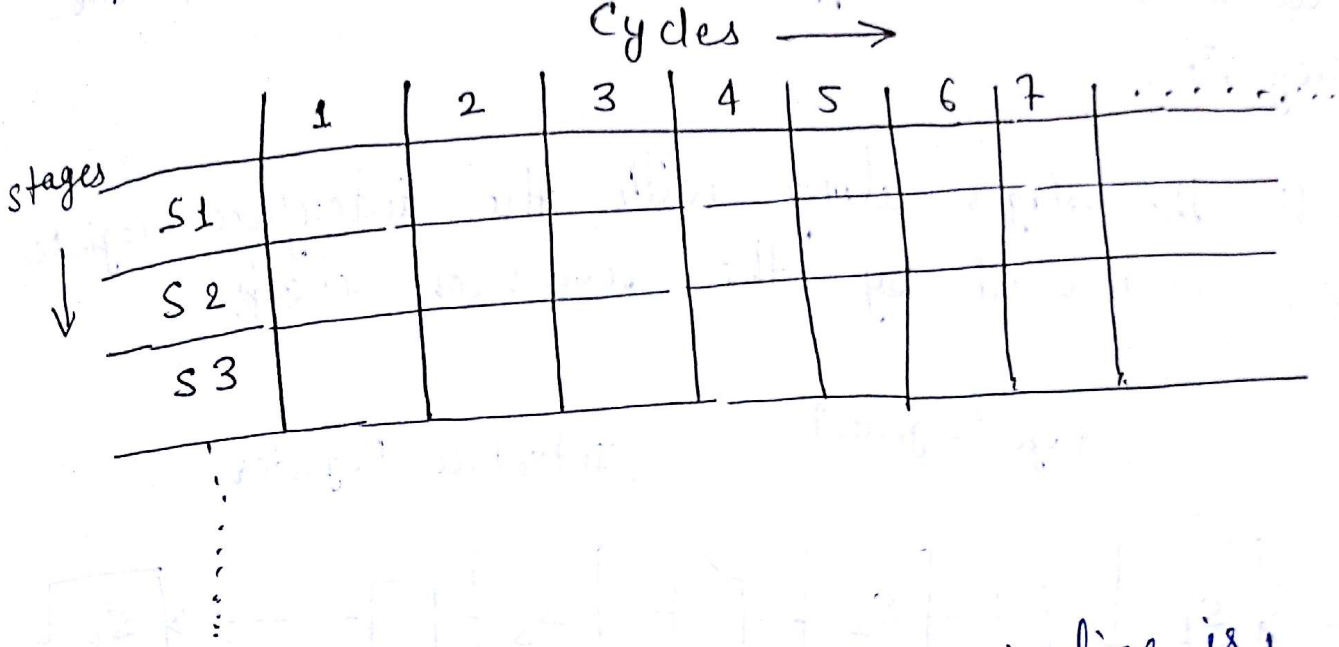
Definition of Pipeline -

(1). Accepting new inputs at one end before previously accepted input appears as an output at the other end.

⇒ The definition states that insert the new inputs before completion of the old inputs, That means the new inputs are executed along with the old inputs. Therefore, pipelining allows the overlapping execution.

⇒ In the non-pipelining process new inputs are inserted after completion of the old inputs. Non-pipelining allows the non-overlapping execution.

⇒ The overlapping execution sequence in the pipeline is represented by using the space-time diagram.



⇒ The succesful characteristic of pipeline is, in every new cycle new input must be inserted into the pipeline.

i.e CPI (Cycle Per Instruction) = 1

Designing of the Pipeline

⇒ Pipe has two ends i.e. input and output end.

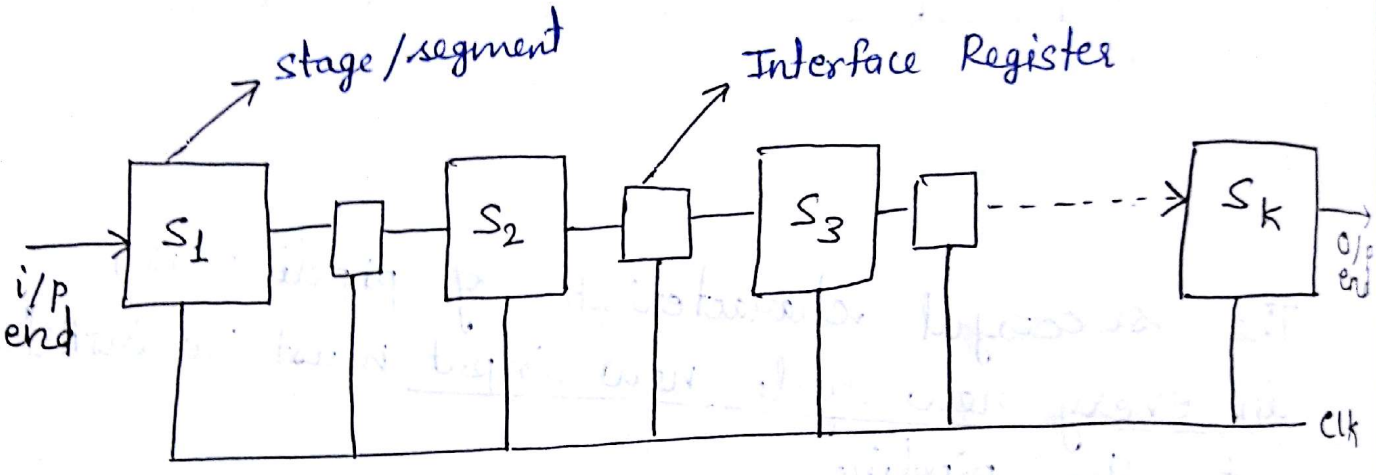
⇒ Between the input and output ends multiple pipes are interconnected to satisfy the objective of the pipeline.

⇒ Each pipe is called as stage or segment.

⇒ Interface registers are used between the stages

to hold the intermediate result. It is also called as the buffer or latch or pipeline register.

⇒ All the stages along with the interface registers are controlled by the common clock.



⇒ Consider 4 segment pipeline used to execute the four tasks and show the execution sequence of 4 tasks in the pipeline.

Stages { S1, S2, S3, S4 }

i/p { T1, T2, T3, T4 }

Stages/Cycles	1	2	3	4	5	6	7	8	9
S_1	T_1	T_2	T_3	T_4					
S_2		T_1	T_2	T_3	T_4				
S_3			T_1	T_2	T_3	T_4			
S_4				T_1	T_2	T_3	T_4		

Overlapping Execution sequence

* Overlapping takes 7 cycles for 4 tasks.

Stages / Cycles	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S1	T_1				T_2				T_3				T_4			
S2		T_1				T_2				T_3				T_4		
S3			T_1				T_2				T_3				T_4	
S4				T_1				T_2				T_3				T_4

Non- Overlapping Execution Sequence

* Non- overlapping takes 16 cycles for 4 tasks.

Performance Evaluation of the Pipeline Processor:-

1). Consider 'K' segment pipeline with the clock cycle time tp used to execute the n tasks.

2). The very first task in the pipeline is executed in the non- overlapping order. Therefore it takes 'k' cycles to complete the execution.

3). The remaining (n-1) tasks emerge from the pipe at the rate of 1 cycle / task. Therefore (n-1) tasks will take (n-1) cycles to complete the execution. Therefore the total time required to

execute 'n' tasks in the 'k' segment pipeline is.

$$ET_{pipe} = (k+n-1) \text{ cycles}$$
$$= (k+n-1) * t_p$$

4) Consider non-pipeline processor used to execute the 'n' tasks. In which each task requires t_n time to complete the execution. Therefore, the total time required to complete the n-task in the non-pipeline processor is.

$$ET_{non-pipe} = n * t_n$$

5). The performance gain of the pipeline processor over the non-pipeline processor is

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

$$= \frac{1 / ET_{pipe}}{1 / ET_{non-pipe}}$$

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

$$S = \frac{n * t_n}{(k+n-1) * t_p}$$

(6). When the n value increases then $(K+n-1) \approx n$, then

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

$$\boxed{S = \frac{t_n}{t_p}}$$

t_p = clock cycle time of pipelined system

t_n = time taken by each task to complete in non-pipelined system.

(7). When all the tasks are taking the same no. of cycles then tasks non-overlapping execution time is also equal to the no. of stages in the pipeline i.e.

$$t_n = k \text{ cycles}$$

$$\boxed{t_n = k * t_p}$$

then

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

$$= \frac{k * t_p}{t_p}$$

$= k =$ no. of stages in the pipeline also called as the pipeline depth.

(8). When the processor is operating with 100% efficiency then max speed up is equal to pipeline depth.

$$100\%\ \eta \quad - \quad S_{max}$$
$$?\ \eta \quad - \quad S$$
$$\eta_{pipe} = \frac{S}{S_{max}} = \frac{S}{K}$$

Efficiency $\eta_{pipe} = \frac{S}{K}$

(9). The throughput of the pipeline is the no. of tasks per total time required to process the task, i.e.

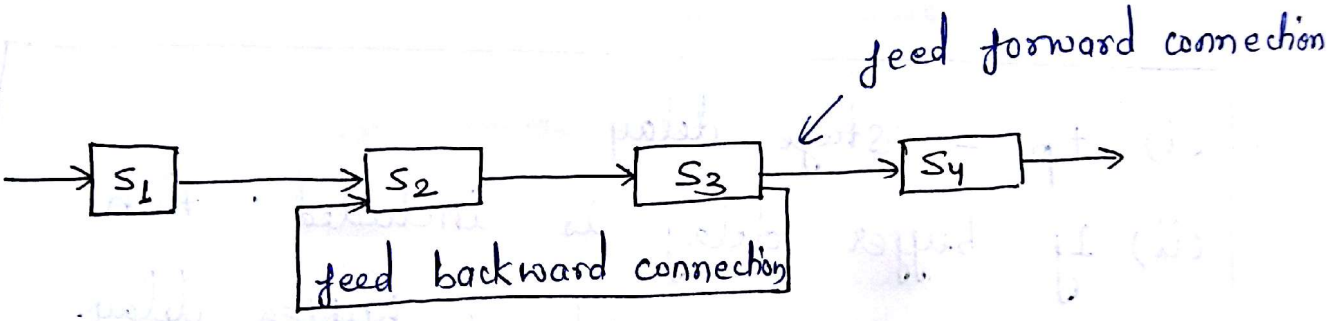
$$TP_{pipe} = \frac{\text{No. of tasks processed}}{\text{Total time required to process the task}}$$

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

Types of the pipeline -

(1). Linear Pipeline - In this pipeline only one specific functionality is performed. Linear pipeline consists only the feed forward connections.

(2). Non-Linear Pipeline - This pipeline perforns the multiple functionalities. It consists feed forward and feed - backward connections.



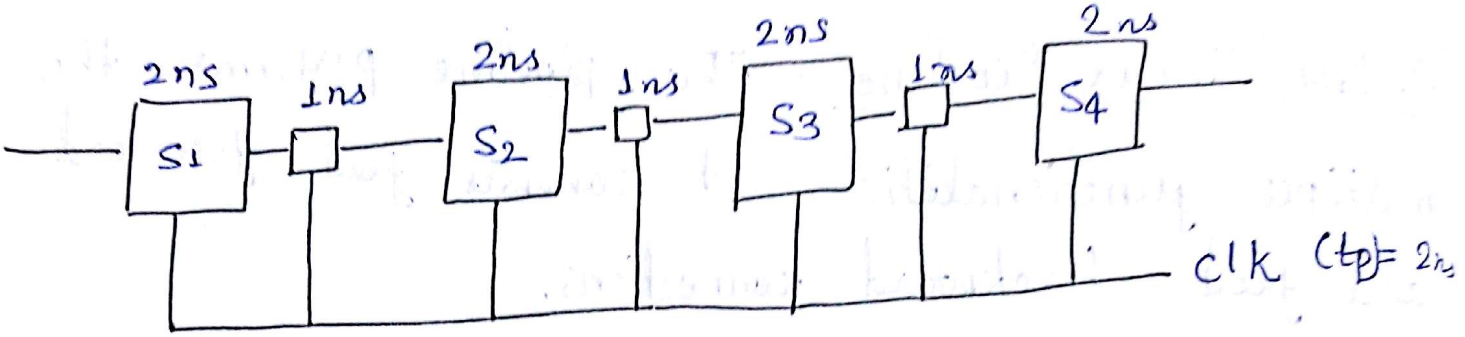
In the non-linear pipeline the execution sequence is depending on the reservation table.

(3). Synchronous pipeline - In this pipeline the operations are initiated based on the clock.

(4). Asynchronous pipeline - In this pipeline the operations are initiated based on the hand-shaking signals.

NOTE :-

(1). Uniform Delay pipeline - In this pipeline all the stages are taking the same amount of delay to complete the operation.



(i). t_p = stage delay

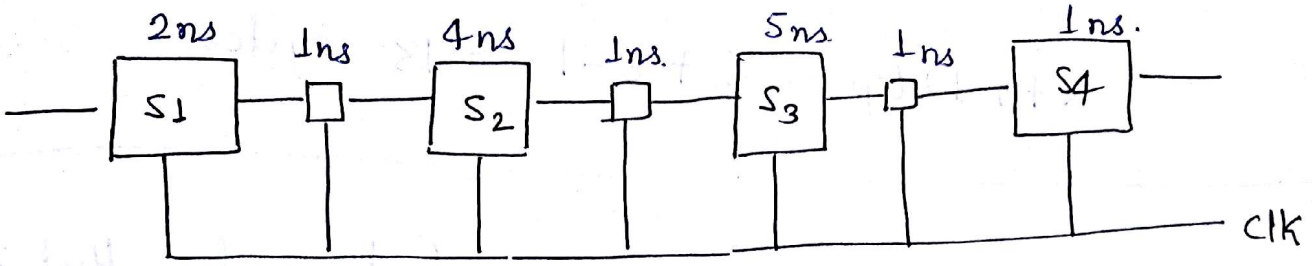
(ii) If buffer delay is included ; then

t_p = stage delay + buffer delay.

(iii). If skewtime / set-up time overhead is included, then

t_p = t_p + overhead delay.

(2). Non- Uniform Delay Pipeline - In this pipeline different stage delays are maintained to complete the operation.



(i). t_p = max (stage delay)

(ii). If buffer delay is included

t_p = max (stage delay) + Buffer delay

(iii). variable Buffer delay present, then

t_p = max (stage delay + Buffer delay)

Qu.2) Draw a space-time diagram for a six-segment pipeline showing the time it takes to process eight tasks.

Solu. -

Segment / cycle	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8					
2		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8				
3			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8			
4				T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8		
5					T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	
6						T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8

$(K+n-1)t_p = 6+8-1 = 13$ cycles.

Qu 3.3) Determine the number of clock cycles that it takes to process 200 tasks in a six-segment pipeline.

Solu.

$K = 6$ segments
$n = 200$ tasks

$(K+n-1) = 6+200-1 = 205$ cycles.

Qu. 9.4). A non-pipeline system takes 50 ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Determine the speed up ratio of the pipeline for 100 tasks. What is the maximum speedup that can be achieved?

Solu.

$t_n = 50$ ns
$K = 6$
$t_p = 10$ ns
$n = 100$

$$S = \frac{n t_n}{(K+n-1) t_p}$$

$$S = \frac{100 \times 50}{(6+100-1) \times 10} = 4.76$$

$$S_{max} = \frac{t_n}{t_p}$$

$$S_{max} = \frac{50}{10} = 5$$

Qu. Consider 2 pipelines A & B, where pipeline A is having 8 stages of uniform delay of 3 ns. Pipeline B is having 5 stages with respective stage delays of 2ns, 3ns, 5ns, 6ns & 1 ns. How much time is saved using the pipeline A instead of pipeline B when 100 tasks are pipelined.

Solu.

pipe A: $K = 8$

$n = 100$

tp = stage delay

$= 3$ ns.

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

$$= (8+100-1) * 3$$

$$= 321 \text{ ns.}$$

Pipe B = k = 5

n = 100

tp = max (stage delay)

= 6 ns.

ET_B = (k + n - 1) tp

= (5 + 100 - 1) * 6

= 624 ns.

Time Save = $ET_B - ET_A$

= 624 - 321

= 303 ns.

Qu. 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 need 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?

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

Solu. Pipelined processor has four stages IF, ID, EX, WB.

Clock Cycles	Instruction
1	ADD
1	SUB
3	MUL

Consider the following diagram

Clock Cycles →	1	2	3	4	5	6	7	8
$R_2 \leftarrow R_1 + R_0$	IF	ID	EX	WB				
$R_4 \leftarrow R_3 * R_2$		IF	ID	EX	EX	EX	WB	
$R_6 \leftarrow R_5 - R_4$			IF	ID			EX	WB

So total required clock cycle is 8.