COMPUTER SCIENCE

Computer Organization and Architecture

Instruction Pipelining

Lecture_01

Vijay Agarwal sir

01 Pipelining Concepts

CPU Time Calculation

Cycle

Cycle Time

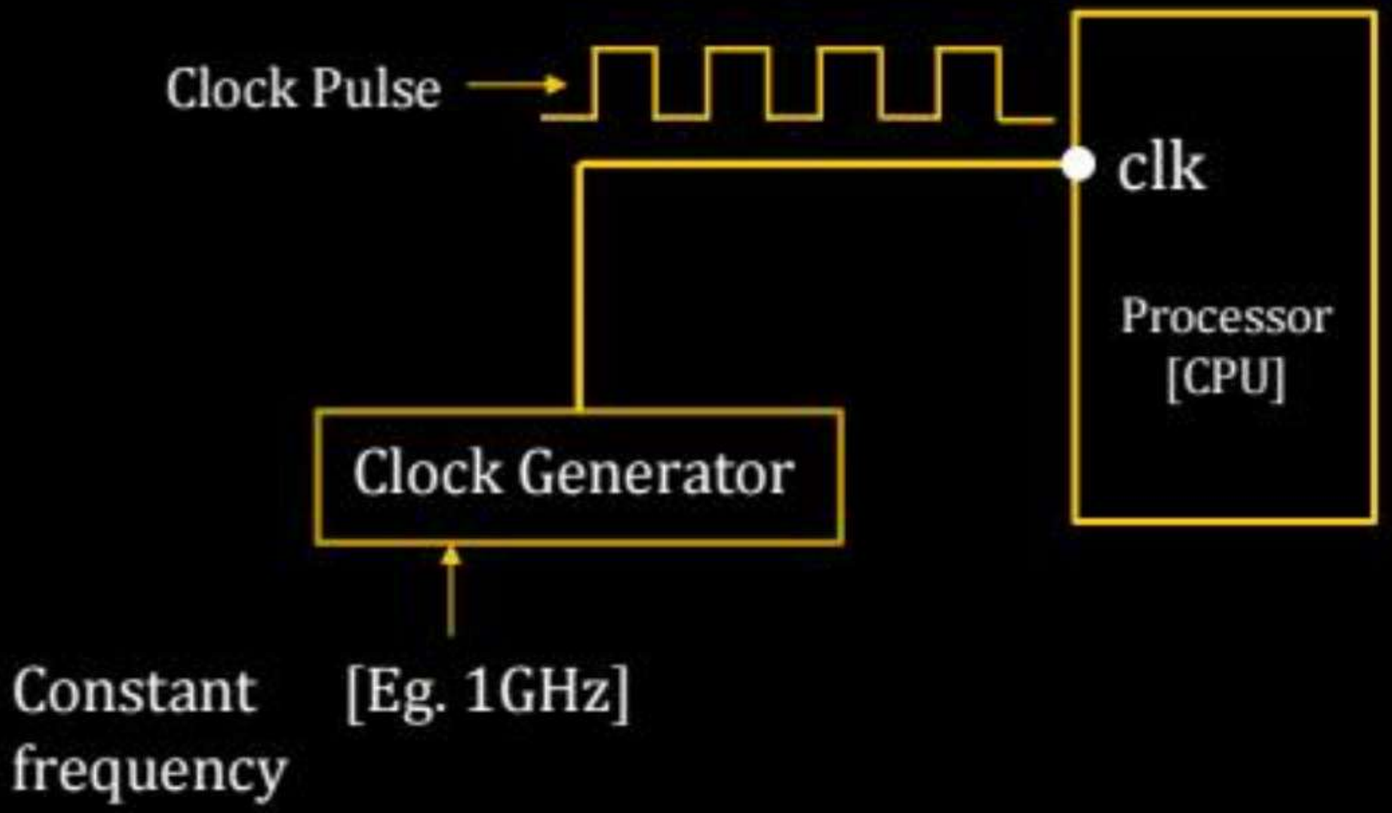
CPI

MIPS

CPU Time Calculations

- CPU Time Calculations Program Execution Time.
- Program Execution time is calculated based on the clock.
- Processor contain clock pins & these clock pin is externally connected with the clock generator.
- (or)
- So in the computer system all the operation are controlled by the clock, so CPU contain clock pins which is externally connected with clock generator.

- Clock Generator is operating with a constant frequency to generate the clock pulse [clock signal].
- These clock signals are carried into the CPU through (with the help of) Clock pin. So CPU operation are controlled by the clock signal.



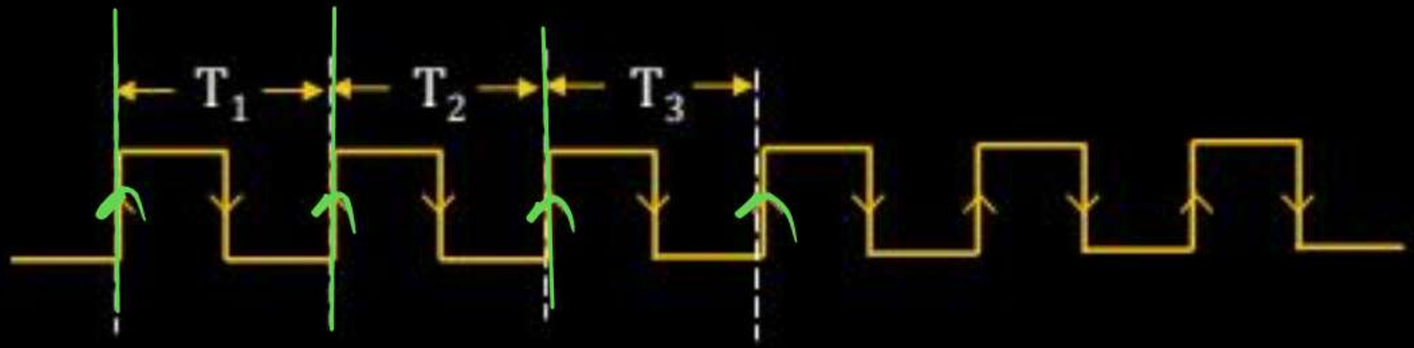
- Program E.T (Execution Time) is calculated based on 2 factor

  1) Cycle

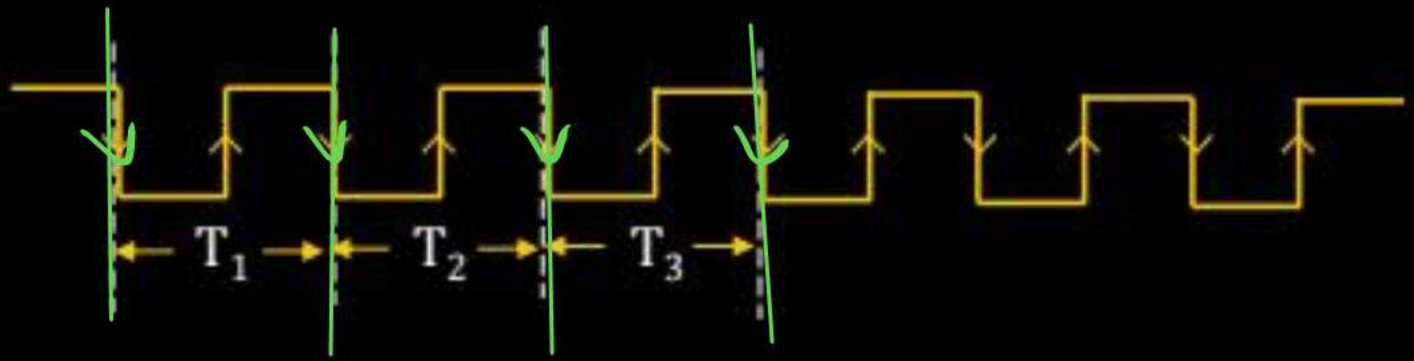
  2) Cycle Time

1) Cycle: Cycle is defined as clock pulse transition either from rising edge to rising edge (or) falling edge to falling edge.

Rising edge



Falling edge



2) Cycle Time: The time required to transfer the pulse either from rising edge to rising edge or falling edge to falling edge is called as cycle time.

Cycle time depends on the clock frequency.

$$\text{Cycle time} \propto \frac{1}{\text{Clock frequency}}$$

$$\text{Cycle time} \propto \frac{1}{\text{Clock frequency}}$$

Example: 1 GHz clock is used.

$$\text{Cycle time} = \frac{1}{1\text{GHZ}}\ \text{sec}$$

$$= \frac{1}{10^9}\ \text{sec}$$

$$= 10^{-9}\ \text{sec}$$

$$= 1\text{ns (nanosecond)}$$

CPU Time Calculation / Program ET

CPU Time means program Execution Time.

Program Execution time = # Seconds / Program

= # Instruction/ program × #Cycle / Instruction × #Second / cycle

- # Instruction/ program → Instruction Count
- #Cycle / Instruction → Cycle per Instruction (CPI)
- #Second / cycle → Cycle time

Prog. ET / CPU time = IC × CPI × cycle time

- Program is a combination of Data transfer, Data manipulation, & Transfer of control (TOC) Instruction. Different Instruction takes (consume) different cycle to complete the execution so,

i = Instruction type

$$\text{Prog. E.T} / \text{CPU Time} = [\Sigma(IC_i \times CPIi)] \text{ Cycle time}$$

i: Type of instruction

$$\left[\Sigma\left(IC_i \times CPI_i\right)\right] \times \text{Cycle time}$$

Let CPI_i be the number of cycles required for instruction type i, and I_i be the number of executed instructions of type i for a given program. Then we can calculate an overall(Average) CPI as follows:

$$\text{CPI}_{(\text{Avg})} = \frac{\sum_{i=1}^{n}(CPIi \times Ii)}{I_c}$$

(eg) $\frac{\quad}{950} = 8.36 \text{ Cycle}$.

The processor time T needed to execute a given program can be expressed as $T = Ic \times CPI \times \tau$ (Where, $\tau = 1/f$, & I_c instruction count)

$950 \times 8.36 \times 0.66 \text{ ns}$ (in that eg.)

A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS) referred to as the MIPS rate. We can express the MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{Ic}{T \times 10^6} = \frac{f}{CPI \times 10^6} \text{ MIPS}$$

Q.

Common Data for Question 1, 2 & 3
Consider a 1.5 GHZ clock frequency processor used to execute the following program segment.

Instruction type	Instruction count	CPI
LOAD	300	11
STORE	200	9
ARITHMATIC	250	7
SHIFT	150	6
BRANCH	50	4

Q. 1 What is Average Instruction Execution of the program?

Q. 2 What is the MIPS Rate of a program?

Q. 3 What is the total Program ET?

Solution for Question 1

$$\text{Average Instruction ET} = \frac{\text{\# of cycle/prog}}{\text{\#of Instruction /prog}}$$

$$\Rightarrow \frac{(300\times11)+(200\times9)+(250\times7)+(150\times6)+(50\times4)}{950}$$

$$\Rightarrow \frac{3300+1800+1750+900+200}{950} = 8.36 \text{ Cycle}$$

$$\text{Cycle Time} = \frac{1}{\text{clock frequency}} = \frac{1}{1.5 \text{ Ghz}} \Rightarrow \frac{1}{1.5} \text{ nsec} \Rightarrow 0.66 \text{ nsec}$$

$$\text{Average Instruction ET} = 8.36 \times 0.66 \text{ nsec}$$

$$\text{Average Instruction ET} = 5.51 \text{ nsec}$$

Solution for Question 2

MIPS

Average Instruction ET = 5.51×10^{-9} sec

In 1 second → # Instruction

$= \frac{1}{5.5 \times 10^{-9}}$ Instruction / second

$= \frac{1}{5.51} \times 10^{9}$ Instruction / second

$= 0.1814 \times 10^{9}$ Instruction / second

$= 181.4 \times 10^{6}$ Instruction / second

= 181.4 MIPS

$\frac{F}{CPI_{Avg} \times 10^{6}}$ MIPS

$= \frac{1.5 \times 10^{9}}{8.36 \times 10^{6}}$ MIPS

Solution for Question 3

Total Program ET = # Instruction / Program × Average Instruction ET

$$= 950 \times 5.51 \text{ nsec}$$

$$= 5234.5 \text{ nsec}$$

Program ET = 5.234 nsec

Q. Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instruction. The instructions mix and the CPI for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	CPI	Instruction Mix (%)
Arithmetic and logic	1	60%
load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

The average CPI when the program is executed on a uniprocessor with the above trace results is CPI = $0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1)$ = 2.24. The corresponding MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$

$1 \times .60 + 2 \times 0.18 + 4 \times 0.12 + 8 \times .10$

$CPI_{Avg} = 2.24$ cycle

$$MIPS = \frac{F}{CPI \times 10^6} \Rightarrow \frac{400 \times 10^6}{2.24 \times 10^6} \approx 178$$

OR

CPI = 2.24 cycle

Cycle time = $\frac{1}{400\,MHz}$ sec

1 Instn ET = ——

In 1 sec ——

10^6 IPS

Pipelining Strategy

To apply this concept
To instruction
Execution we must
Recognize that an
Instruction has a
Number of stages

Similar to the use of
An assembly line in a
Manufacturing plant

New inputs are
Accepted at one end
Before previously
Accepted inputs
Appear as output at
The other end

PIPELINE

- Pipelining is a mechanism which is used to improve the performance of the system in which task (Instruction) are executed in overlapping manner.
- Pipelining is a decomposition technique that means the problem is divided into sub problem & Assign the sub problem to the pipes then operate the pipe under the same clock.

PIPELINE

- Pipelining means Accepting New Input at one end before the Previously accepted i/p appear as an output at other end.
- That means New Input are executed with OLD Input in a Overlapping Manner (in a pipelining)

PIPELINE

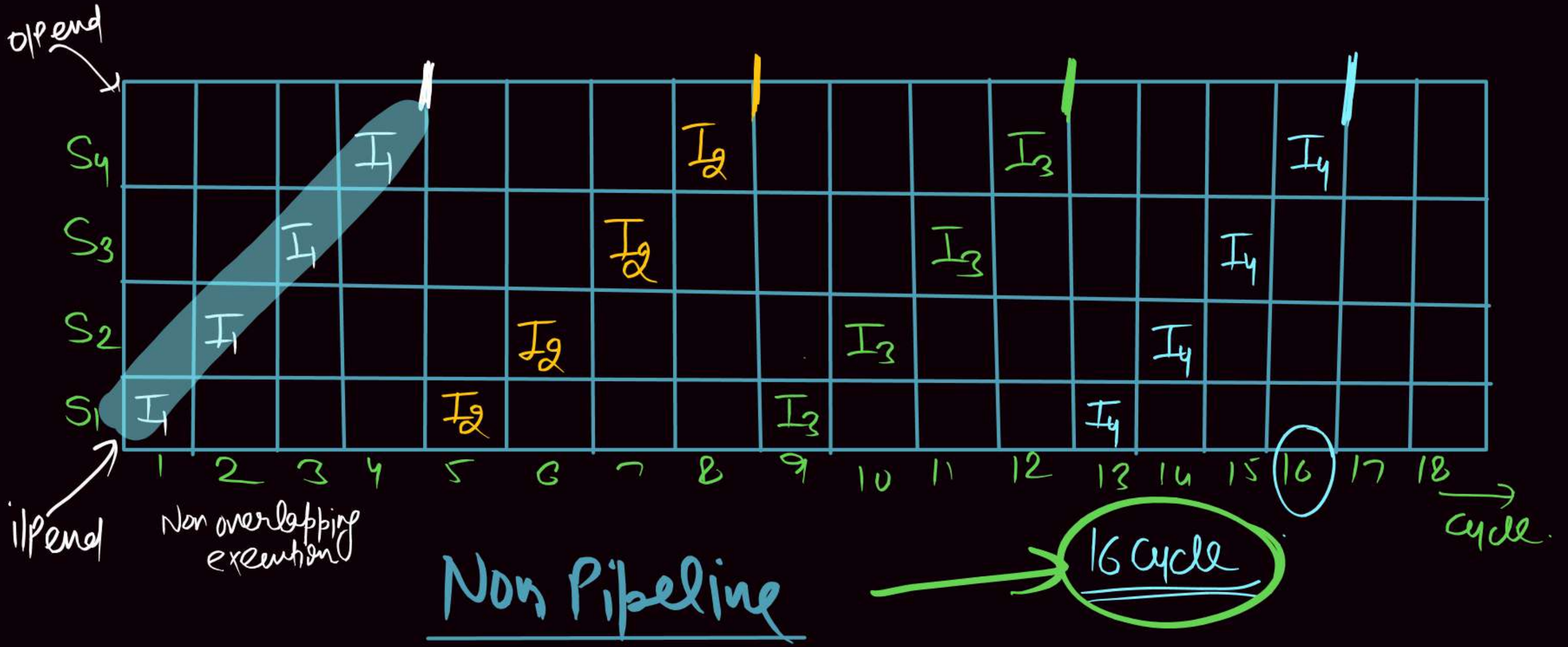
Non Pipeline : Accepting New Input Only After Completion of OLD Input

(OR)

Accepting New Input After Previously Accepted Input appear as as output at Other end.

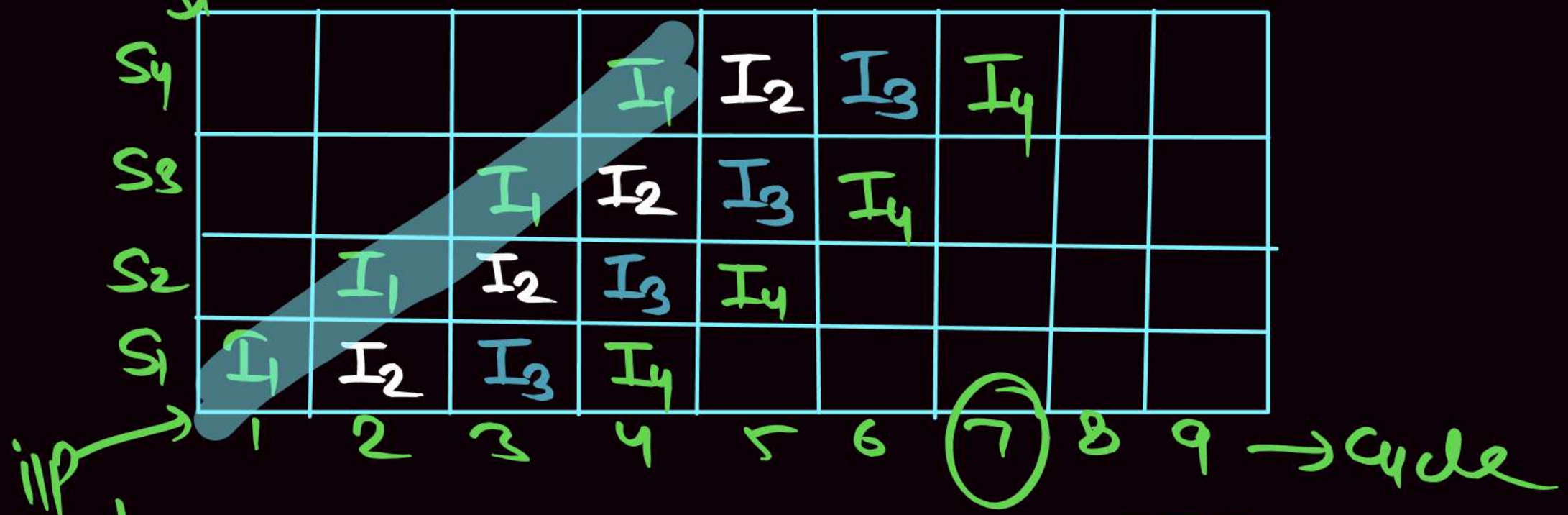
In Non Pipelining Non overlapping execution.

I_1, I_2, I_3, I_4


o/p end
S4
S3
S2
S1
I1
I2
I3
I4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
cycle.
i/p end
Non overlapping execution
Non Pipeline
16 cycle

Olpend

4 Instn (I_1, I_2, I_3, I_4).

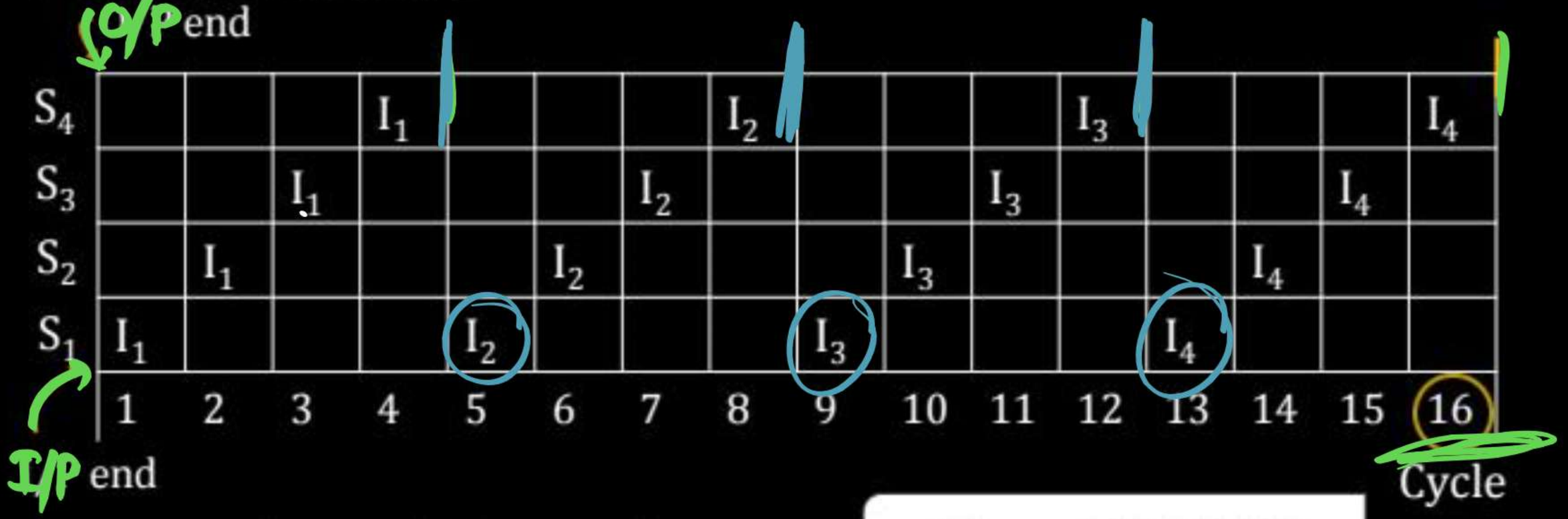


PIPELINE → 7 cycle

∴

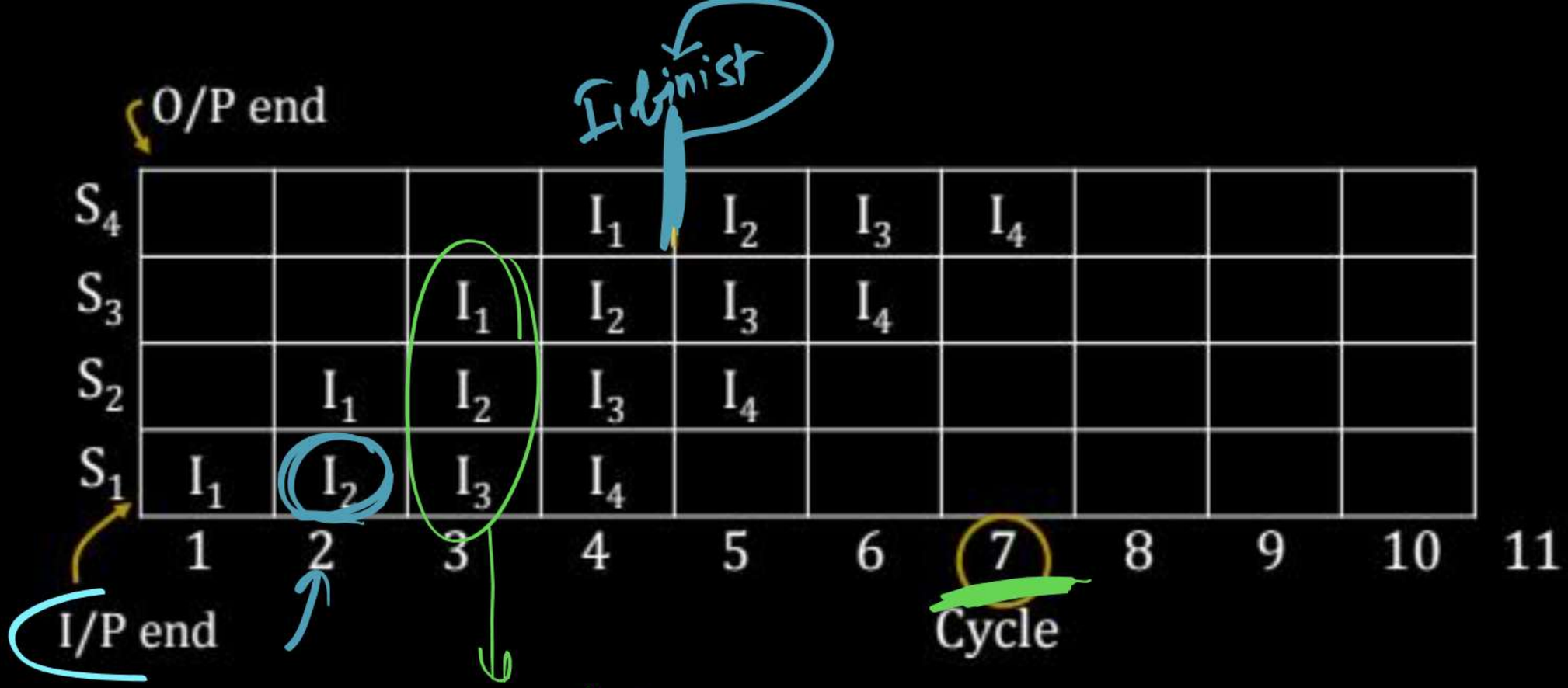
Let us consider 4 segment pipeline used to execute 4 instruction the execution sequence as:

Segment/stages = [S_1 S_2 S_3 S_4]
Instruction: [I_1 I_2 I_3 I_4]



n = 4, t_n = 4, Non pipeline

Non-PIPELINE



PIPELINE

k = 4

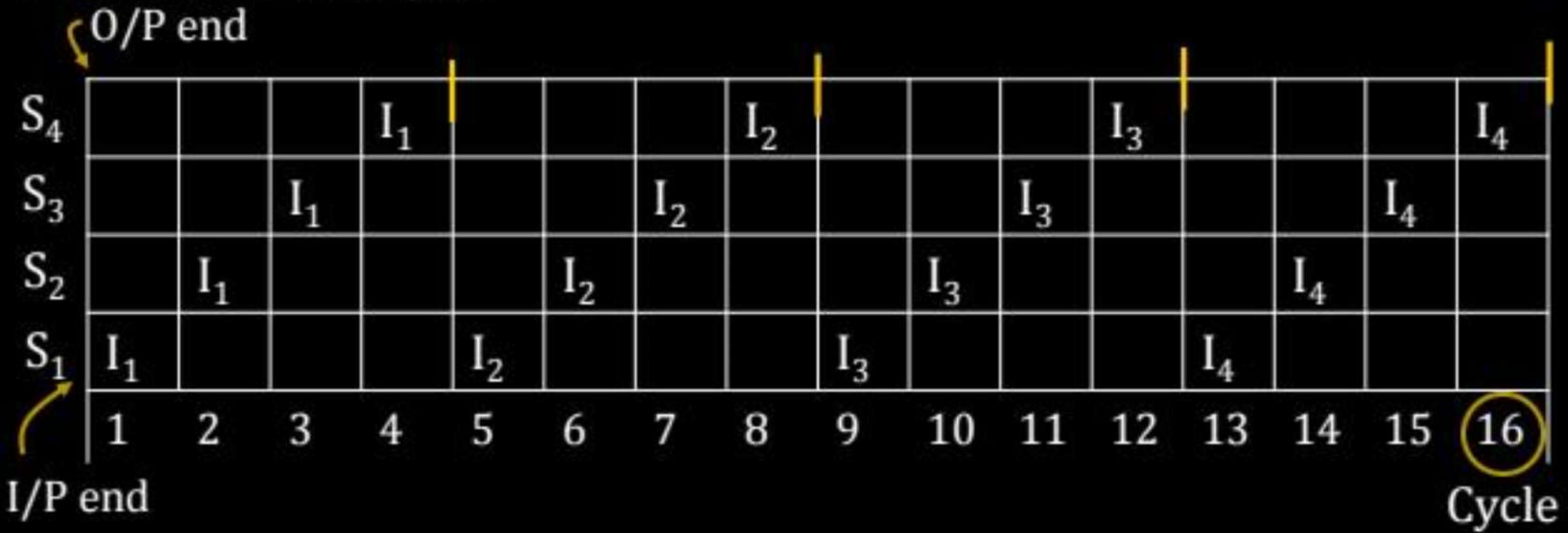
n = 4

PIPELINE

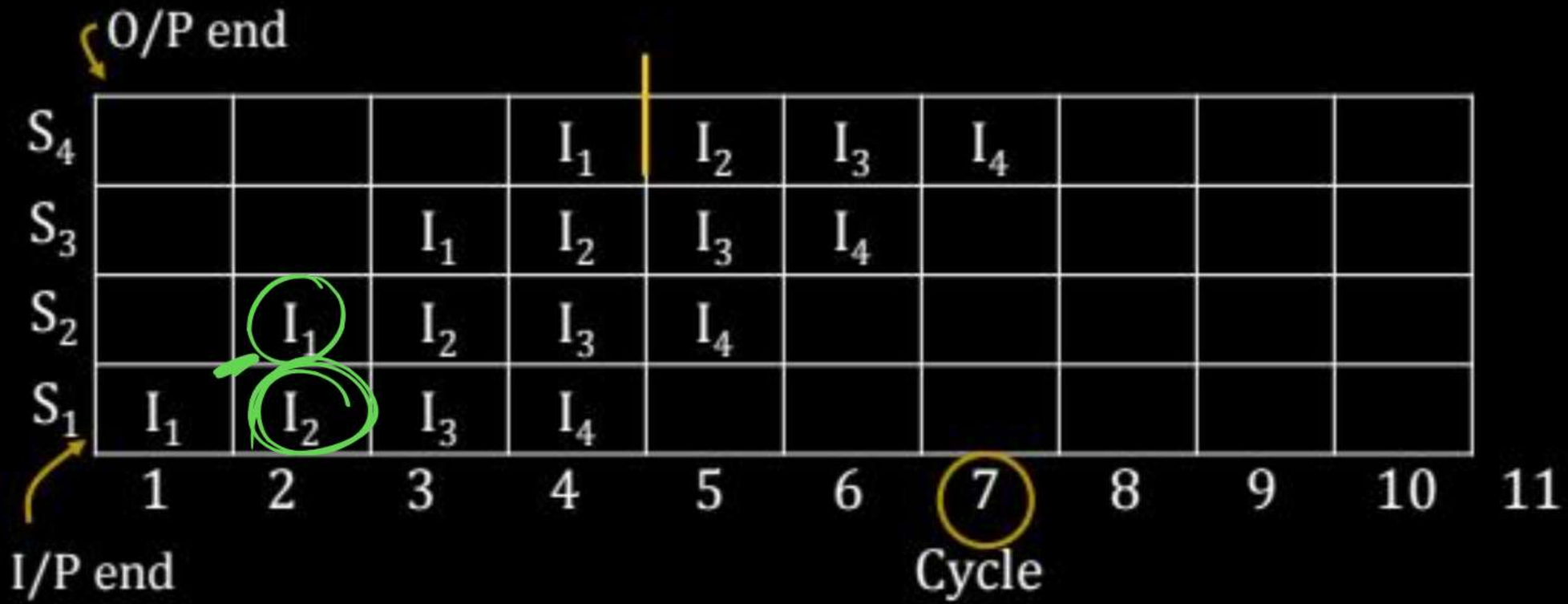
Let us consider 4 segment pipeline used to execute 4 instruction the execution sequence as: PIPELINING

Segment/stages = $[S_1\ S_2\ S_3\ S_4]$

Instruction: $[I_1\ I_2\ I_3\ I_4]$



n = 4, t_n = 4, Non pipeline



PIPELINE

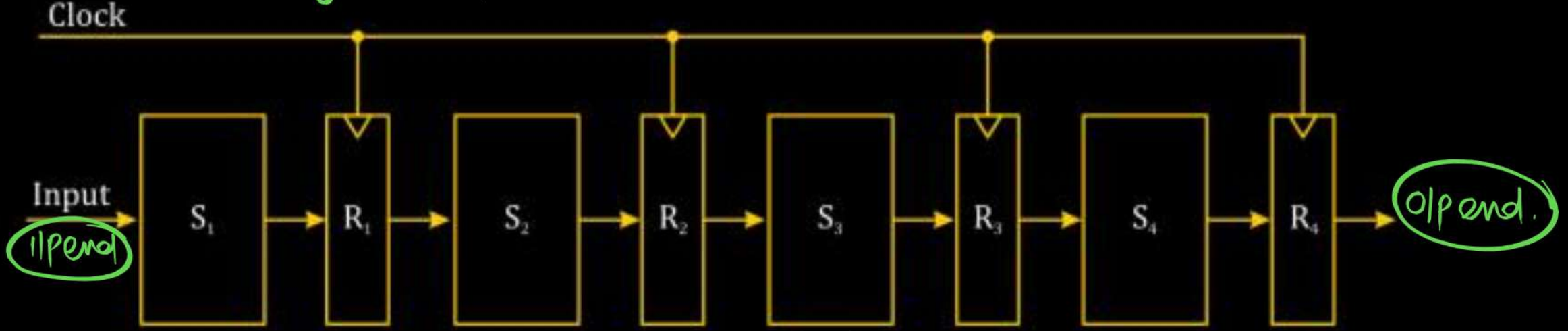
k = 4

n = 4

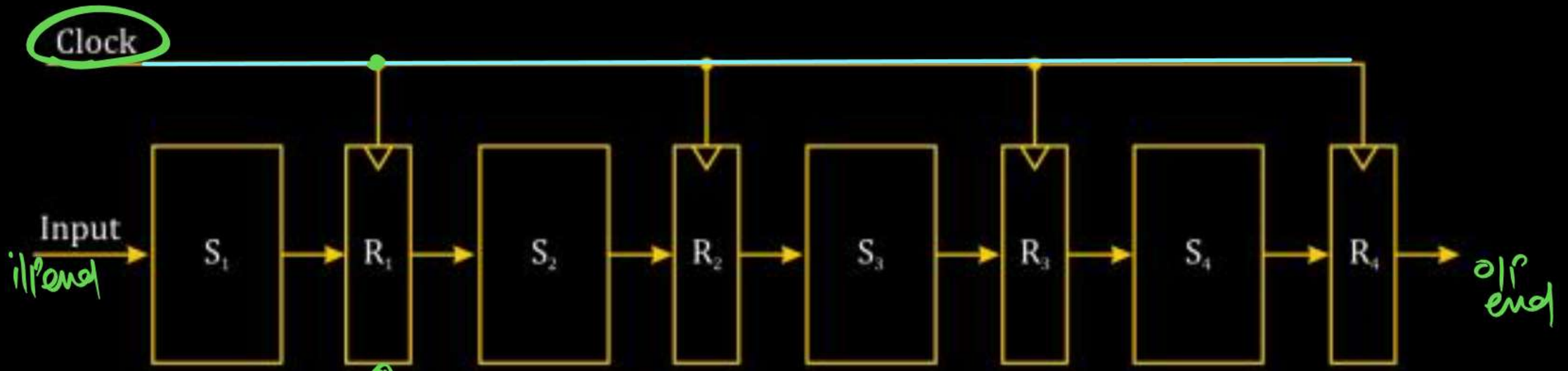
PIPELINE Design

Pipe has 2 end → Input end
→ Output end

Between the these 2 end Multible pipes are interconnected to functioning of Pipeline.



Four-segment pipeline.



Four-segment pipeline.

Stage / Segment

Buffer / Interface Register

PIPELINE Design

- These Pipes are called Stage | Segment.
- Between the Stages 'Buffer' are used to store the Intermediate Result.
- This Buffer is also called as Pipeline Register (or) Interface Register (or) Latch (or) Buffer.
- All the Stage along with the Buffer are controlled by Common clock.

Let us consider 4 segment pipeline used to execute 4 instruction the execution sequence as: PIPELINING

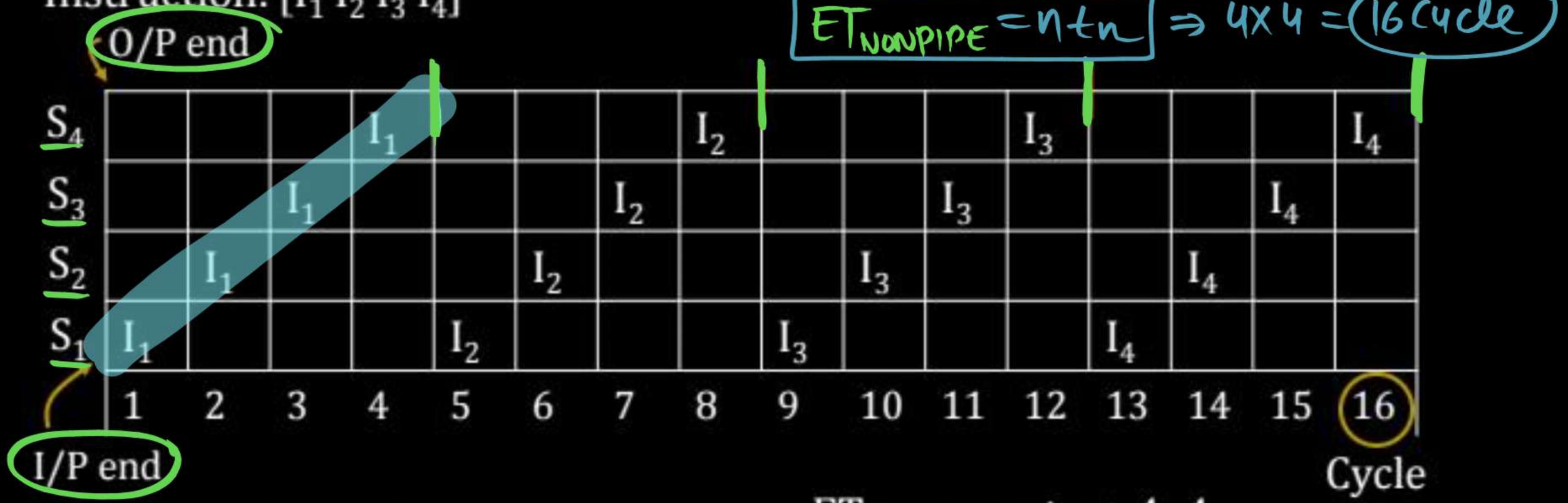
Segment/stages = [S_1 S_2 S_3 S_4]

Instruction: [I_1 I_2 I_3 I_4]

$t_n = 4$ cycle

$n = 4$

$ET_{NONPIPE} = n t_n \Rightarrow 4 \times 4 = 16$ cycle



$n = 4$, $t_n = 4$, Non pipeline

$ET_{NONPIPE} = nt_n \Rightarrow 4 \times 4$

$ET_{NONPIPE} = 16$ cycle

$$ET_{PIPELINE} = [k + (n-1)]\, tp$$

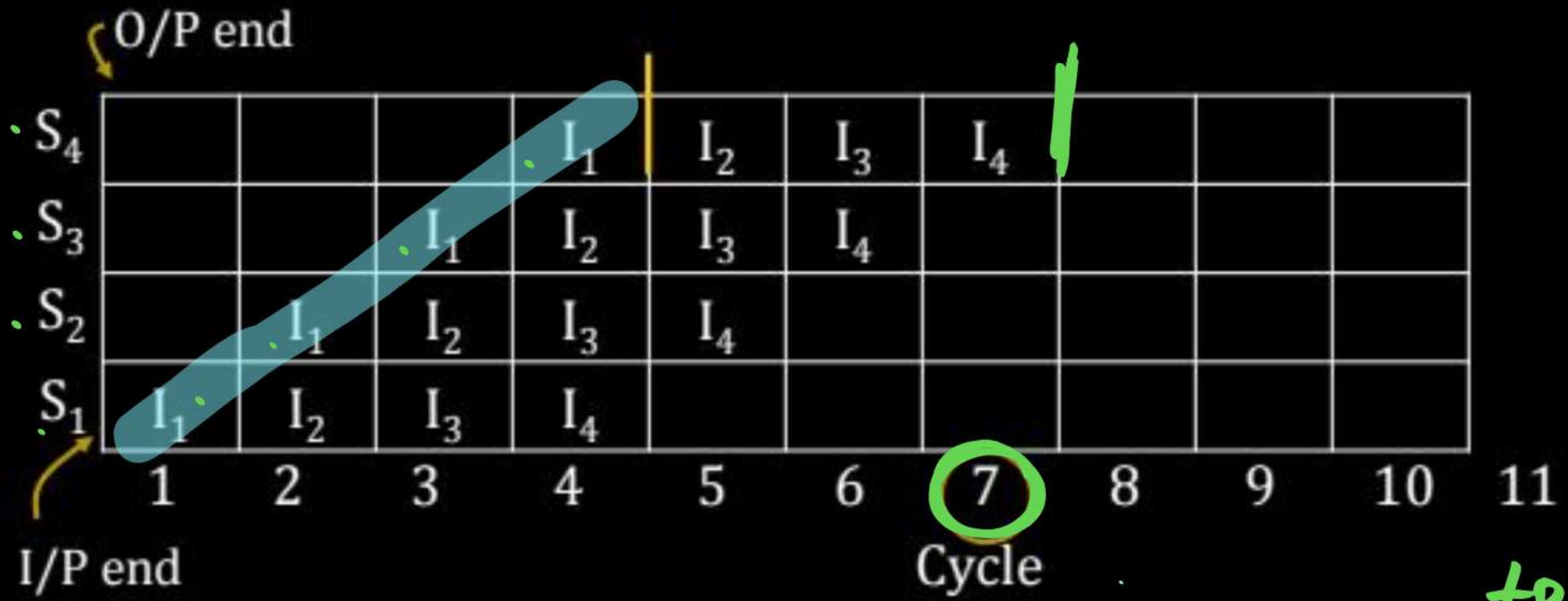
$$= [4 + (4-1)] \times 1$$

$$ET_{PIPE} = 7 \text{ cycle}$$

$k = 4$

$n = 4$

$tp = 1 \text{ cycle}$



PIPELINE
k = 4
n = 4

$$ET_{PIPE} = (k + (n-1))tp$$
$$= (4 + (4 - 1) \times 1 \text{ cycle})$$
$$ET_{PIPE} = 7 \text{ cycle}$$

$tp = 1$
$k = 4$
$k * tp + (n-1)tp$

$$ET_{PIPE} = [k + (n-1)]\,tp$$

PIPELINE

Execution time in Pipeline:

$$ET_{PIPE} = k * tp + (n-1)tp$$

$$ET_{PIPE} = [k+(n-1)] \text{ cycle}$$

$$\boxed{ET_{PIPE} = [k+(n-1)]tp}$$

k: #Stages [Segment]

n: #Task (or) #Instruction

tp: cycle time in Pipeline [each stage Delay in Pipeline]

Execution Time in Non Pipeline.

In Non Pipeline 1 Instructions takes t_n time.

Time Taken to execute n Instn in Non Pipeline.

$$ET_{NONPIPELINE} = n\,t_n$$

n: # Instn (or) # Task

t_n: each Instn ET in Non Pipeline.

PIPELINE Performance Evaluation

Now consider the case where a k-segment [stages] pipeline with a clock cycle time t_p, is used to execute n tasks.

The first task T_1 requires a time equal to kt_p, to complete its operation since there are k segments in the pipe.

The remaining n – 1 tasks emerge from the pipe at the rate of one task per clock cycle and they will be completed after a time equal to $(n - 1)t_p$.

Therefore, to complete n tasks using a k-segment pipeline requires $k + (n - 1)$ clock cycles.

$k = 4$

$n = 6$

$t_p = 1$ cycle

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

$$= (4 + (6-1)) \times 1 \text{ cycle}$$

$$ET_{PIPE} = 9 \text{ cycle}$$ Ans

Non Pipeline

t_n : 4 cycle

$$ET_{NONPIPE} = n\, t_n$$

$$= 6 * 4$$

$$= 24 \text{ cycle}$$

Space-time diagram for pipeline.

Segment:

	1	2	3	4	5	6	7	8	9
1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

Clock Cycles

For example, the diagram of Fig. 9-4 shows four segments and six tasks. The time required to complete all the operations is 4 + (6 – 1) = 9 clock cycles, as indicated in the diagram.

So, to complete n tasks using a *k*-segment pipeline is:

ET pipe= $k + (n - 1)$ clock cycles

$ET_{PIPE} = [k + (n-1)]t_p$

k : # Stages (or) # Segment

n : # Instruction (or) # Task

t_p : Each Stage Delay in Pipeline

So, to complete n tasks in Non- pipeline is:

ET Non-pipeline= $n * tn$

n: # Instruction (or) # Task

tn: Each Instn (one Instn) ET in Non Pipeline.

The Performance Gain of Pipeline over Non Pipeline

Performan Gain [Speed up Factor] $= \frac{\text{Performance of PIPE}}{\text{Performance of Non Pipe}}$

$$= \frac{1/ET_{PIPE}}{1/ET_{NONPIPE}}$$

Performance $\propto \frac{1}{ET}$

RAM : 10 Hours

SHYAM : 5 Hours

SHYAM Performance Better.

$$S = \frac{ET_{NONPIPE}}{ET_{PIPE}}$$

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

(i) $k=4$
$n=4$

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

$$= \frac{4\,t_n}{7\,t_p}$$

$$= \frac{4}{7}\,\frac{t_n}{t_p}$$

$$= 0.56\,\frac{t_n}{t_p}$$

$k=4$
$n=100$

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

$$S = \frac{100\,t_n}{103\,t_p}$$

$k=4$
$n=1000$

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

$$= \frac{1000\,t_n}{1003\,t_p}$$

$k=4$
$n=10000$

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

$$= \frac{10000\,t_n}{10003\,t_p}$$

$[k+(n-1)] \simeq n$

When very large No. Instn Executed

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

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

As the number of tasks increases, n becomes much larger than $k - 1$, and $k + n - 1$ approaches the value of n. Under this condition, the speedup becomes

When very large of Instruction executed → $S = \frac{t_n}{t_p}$

(or)

n value is Not given

(or)

In Ideal Case.

When very large number of Task are executed then

$(k+(n-1) \approx n$

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

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

SUMMARY Till Now.

$$ET_{PIPE} = [k + (n-1)]\, tp$$

$$ET_{NONPIPE} = n\, tn$$

$$\text{SPEEDUP FACTOR } [S] = \frac{ET_{NP}}{ET_P}$$

$$S = \frac{n\, tn}{[k + (n-1)]\, tp}$$

$$\eta = \frac{S}{k}$$

η: efficiency

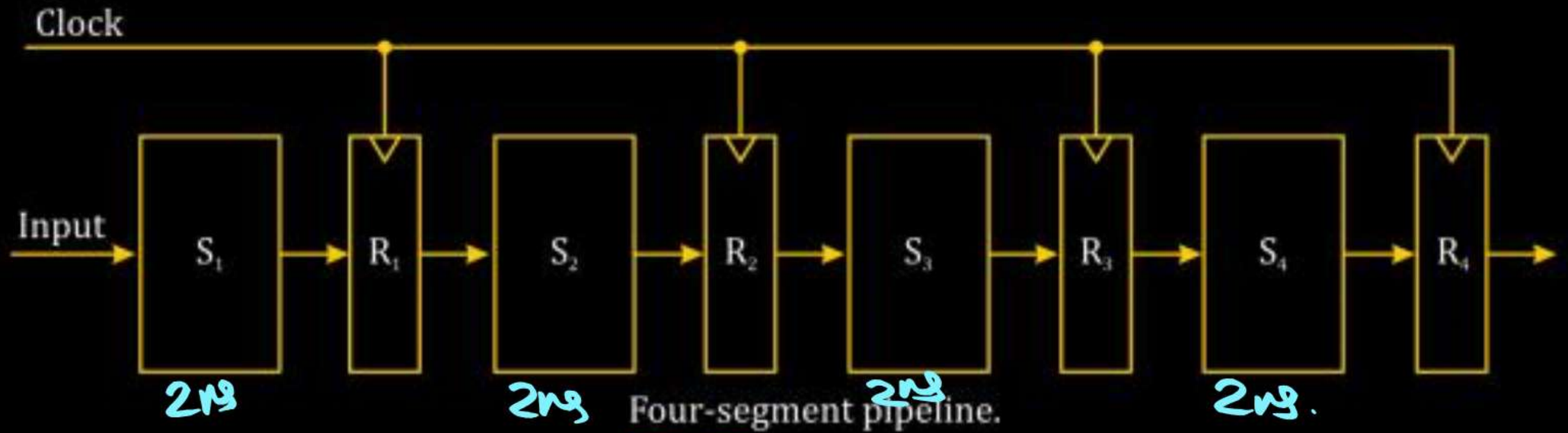
S: speedup Factor, k: # Stage.

When n value is Not given (or) In ideal Case (or) when very Large No. of Instn Executed then

$$S = \frac{tn}{tp}$$

eg ET_{PIPE} for 1 Instn

$ET_{NONPIPE}$ for 1 Instn



Four-segment pipeline.

$$ET_{PIPE} = [k + (n-1)]\,t_p$$

$$= [4 + (1-1)] \times 2\,ns$$

$$\boxed{ET_{PIPE} = 8\,ns}$$ Ans

$$\boxed{ET_{NONPIPE} = n\,t_n}$$

$$t_n = 2 + 2 + 2 + 2$$

$$t_n = 8\,nsec.$$

$$ET_{NP} = 1 \times 8\,ns$$

$$\boxed{ET_{NP} = 8\,ns}$$ Ans

Ideal case

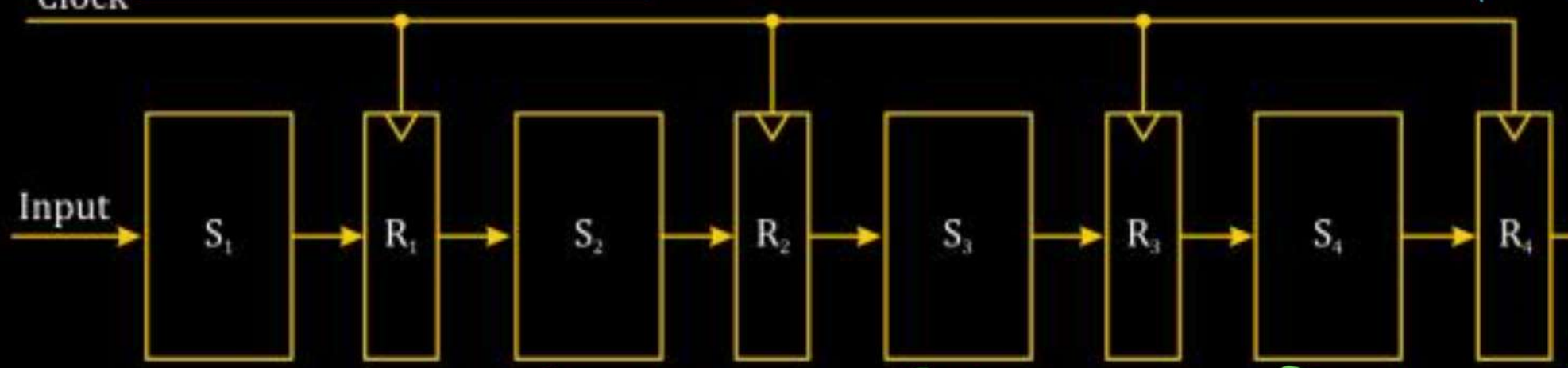
When in Pipeline all Stages are Perfectly balanced then 1 Task ET in Non PIPELINE

$$ET_{NP} = k * tp.$$

So

$k = 4$ Stage.

$tp = 2$ cycle (or) 1 cycle.



Four-segment pipeline

2 cycle (or) 1 cycle

2 cycle (or) 1 cycle

2 cycle (or) 1 cycle

2 cycle (or) 1 cycle

Each Stages taking same cycle.

tn

$k = 4$ (# Stages)

$t_p = 2$ cycle

$t_n = 2 + 2 + 2 + 2$

$= 4 * 2$ (k, t_p)

$= 8$

$t_n = k * t_p$

this But Only in one Condition when Pipeline are Perfectly balance.

$k = 4$ Stage

$t_p = 1$ Cycle

$t_n = 1 + 1 + 1 + 1$

$= 4 * 1$ (k, t_p)

$t_n = k * t_p$

$$S = \frac{ET_{NP}}{ET_P}$$

In Ideal Case

$$S = \frac{tn}{tp}$$

(when n is Not given (or) in Ideal Case)

$$S = \frac{k * tp}{tp}$$

In Ideal Case

When Perfectly Balance

$$S = k$$

In OS

Throughput : # Process Completed Per time Unit

In CN

throughput = eff x B.W.

Throughput : Rate of output.

Throughput: Number of Task are completed per unit of Time.

for n Instn
$ET_{PIPE} = [k+(n-1)]tp$ time

ie In $[k+(n-1)]tp$ Time $\Rightarrow$ n Instruction (or) Task executed

So in 1 unit [per unit] of time $= \frac{n}{[k+(n-1)]tp}$

$$\text{Throughput}\ [T_p] = \frac{n}{[k+(n-1)]tp}$$

In Pipeline

When Number of Instn [Task] are Not given (or) In Ideal Case.

Number of Task are Completed Per (1 unit) unit of time.

$$\text{Throughput} = \frac{1}{t_p}$$

Again Summary till Now

$$ET_{PIPE} = [k + (n-1)]\,tp$$

$$ET_{NonPIPE} = n\,tn$$

$$\text{SPEED Factor } [S] = \frac{n\,tn}{[k + (n-1)]\,tp}$$

When n is very large (or) n value Not given (or) Ideal Case

$$S = \frac{tn}{tp}$$

$$\text{Throughput} = \frac{n}{[k + (n-1)]\,tp}$$

When n value Not given or in Ideal Case

$$\text{Throughput} = \frac{1}{tp}$$

$$\eta_{\text{efficiency}} = \frac{S}{K}$$

η = efficiency

S: Speed up Factor (or) Performance GAIN

k: # Stages (or) # segment

n: # of Instructions (or) # of Tasks.

tn: Each (one) Instruction ET in Non Pipeline.

tp: Each Stage Delay in Pipeline.

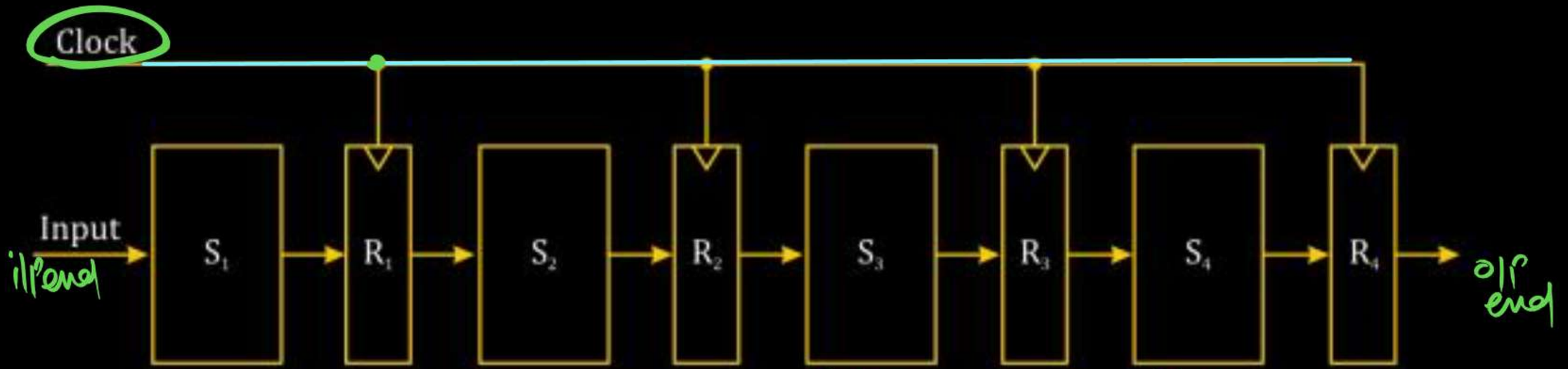
Type of PIPELINE

① UNIFORM Delay : Linear Pipeline

② Non Uniform Delay : Non Linear Pipeline

UNIFORM Delay Pipeline :

In Uniform Delay each Stages taking the Same Amount of Complete the assigned Task to Perform task.



Four-segment pipeline.

UNIFORM Delay

(n=1) 1 Task ET in Pipeline:

$$ET_{PIPE} = [k + (n-1)]\, t_p$$

$k = 4, \quad n = 1$

t_p: [Stage Delay] $= 2$ ns

$$ET_{PIPE} = [k + (n-1)]\, t_p$$

$$= (4 + (1-1)) \times 2\text{ns}$$

$$ET_{PIPE} = 8\text{ns.}$$ Ans

1 Task ET in Non Pipeline

$$ET_{NONPIPE} = n\, t_n$$

$n = 1$

$t_n = 2 + 2 + 2 + 2$

$t_n = 8$ ns

$ET_{NP} = n * t_n$

$= 1 \times 8$

$$ET_{NONPIPE} = 8\text{ nsec}$$

1 Task ET in PIPELINE ≡ 1 Task ET in Non PIPE.

In Pipeline.

If Bubble Delay is given in the Question

$$t_p = \text{Stage Delay} + \text{Bubble Delay}$$

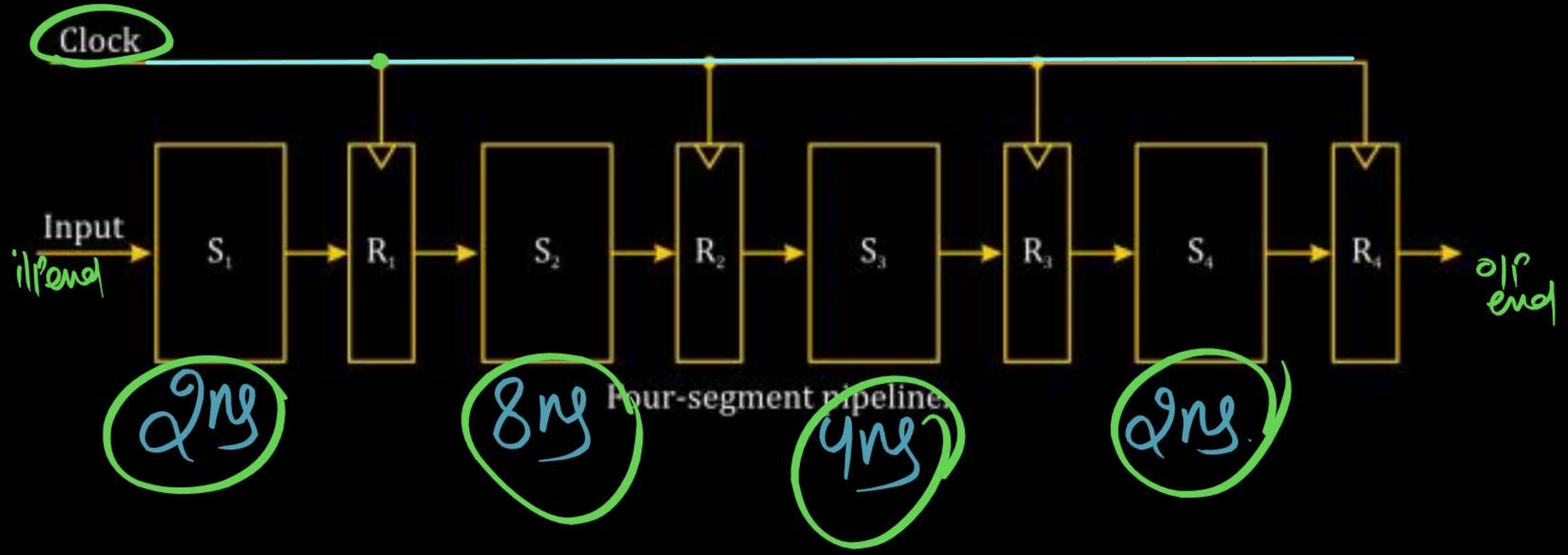
Assume Bubble Delay = 1 ns

$t_p = 2 + 1$

$$t_p = 3\text{nsec}$$

Non UNIFORM Delay PIPELINE :

In Non Uniform Delay Pipeline Each Stages taking Different amount of time to complete the assigned operation.



Four-segment pipeline

Non uniform Delay PIPELINE

n=1, k=4

1 Task ET in PIPELINE

$$ET_{PIPE} = [k+(n-1)]\,tp$$

$tp = \max[\text{Stage Delay}]$

$= \max[2, 8, 4, 2]$

$tp = 8\,nsec$

$ET_{PIPE} = [k+(n-1)]\,tp$

$= [4+(1-1)] \times 8ns$

for 1 Task

$ET_{PIPE} = 32\,nsec$

1 Task ET in Non Pipeline

n=1

$$ET_{NONPIPE} = n\,tn$$

$tn = 2+8+4+2$

$tn = 16\,nsec$

$ET_{NP} = n\,tn$

$= 1 * 16$

for 1 Task

$ET_{NONPIPE} = 16\,nsec.$

If Rubber Delay is given in the Question

$$tp = \max[(\text{Stage Delay} + \text{Rubber Delay})]$$

Non Uniform Delay

If $n = 1000$ [for 1000 Instruction].

$$ET_{PIPE} = [k + (n-1)]\,tp$$

$k = 4,\ n = 1000$

tp : max (Stage Delay) = max (2, 8, 4, 2)

$tp = 8$ nsec

$$ET_{PIPE} = [k + (n-1)]\,tp$$
$$= [4 + (1000 - 1)] \times 8$$
$$= 1003 \times 8$$

$$ET_{PIPE} = 8024 \text{ nsec}$$

For 1000 Instn in Non pipeline

$$ET_{NP} = n * tn$$

$tn = 2 + 8 + 4 + 2 = tn = 16$

$ET_{NP} = 1000 \times 16$

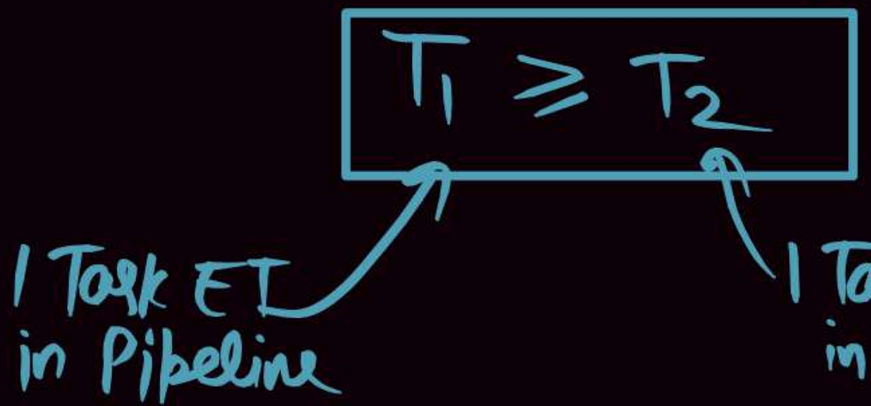
$$ET_{NON\ PIPE} = 16000 \text{ nsec}.$$

Important Points

① When Pipeline are Perfectly balanced [Uniform Delay] then 1 Task [Instn] ET in Pipeline is equal to 1 Task ET in Non Pipeline

② When Pipeline are Not Perfectly balanced [Non Uniform Delay] then One Task ET in Pipeline is greater than 1 Task ET in Non Pipeline.

only for 1 Task



③ But when the Number of Instruction (Task) increased then PIPELINE Performance is Best in Non Uniform & Uniform Delay Pipeline.

④ Buffer Delay is included only in Pipeline. Because in Non Pipeline we are Not Storing the Intermediate Result. So Only in PIPELINE Buffer Delay include if given in Question.

For Uniform Delay → $t_p = \text{Stage Delay} + \text{Buffer Delay}$

For Non Uniform Delay → $t_p = \text{Maximum} \left[(\text{Stage Delay} + \text{Buffer Delay}) \right]$

$\therefore$

Q. 1 Consider on instruction pipeline which has speed up factor 20 while operate with 80% efficiency. What could be the number of stages in the pipeline?

Q. 2

4 segment pipeline have the respective stage delay of 10ns, 15ns, 20ns and 30ns. What is the efficiency of the pipeline when very large number of task are executed?

Q. 3 A 4 segment instruction pipeline has the respective stage delay of 8ns, 11ns 20ns, 2ns respectively. The interface register are used between the stages have a delay of 2ns. What is the approx. speed up factor when very large number of task are pipelined?

Q. 4 Consider 4 stage pipeline with the respective stage delay of 20ns, 80ns, 50ns and 10ns. In the enhancement process of the pipeline largest stage is split into 2 equal stage delay.

(i) What is speed up between new and old design?

(ii) What is the clock frequency of new pipeline?

Q. 1 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 **[GATE-2004: 2 Marks]**

A 120.4 microseconds

B 160.5 microseconds

C 165.5 microseconds

D 590.0 microseconds

Q. 2 We have two designs D1 and D2 for a synchronous pipeline processor. D1 has 5 pipeline stages with execution times of 3 nses, 2 usec, 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? **[GATE-2005: 2 Marks]**

A 214 nseconds

B 202 nseconds

C 86 nseconds

D -200 nsecond

Q. 3 A nonpipelined 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 **[GATE-2008: 2 Marks]**

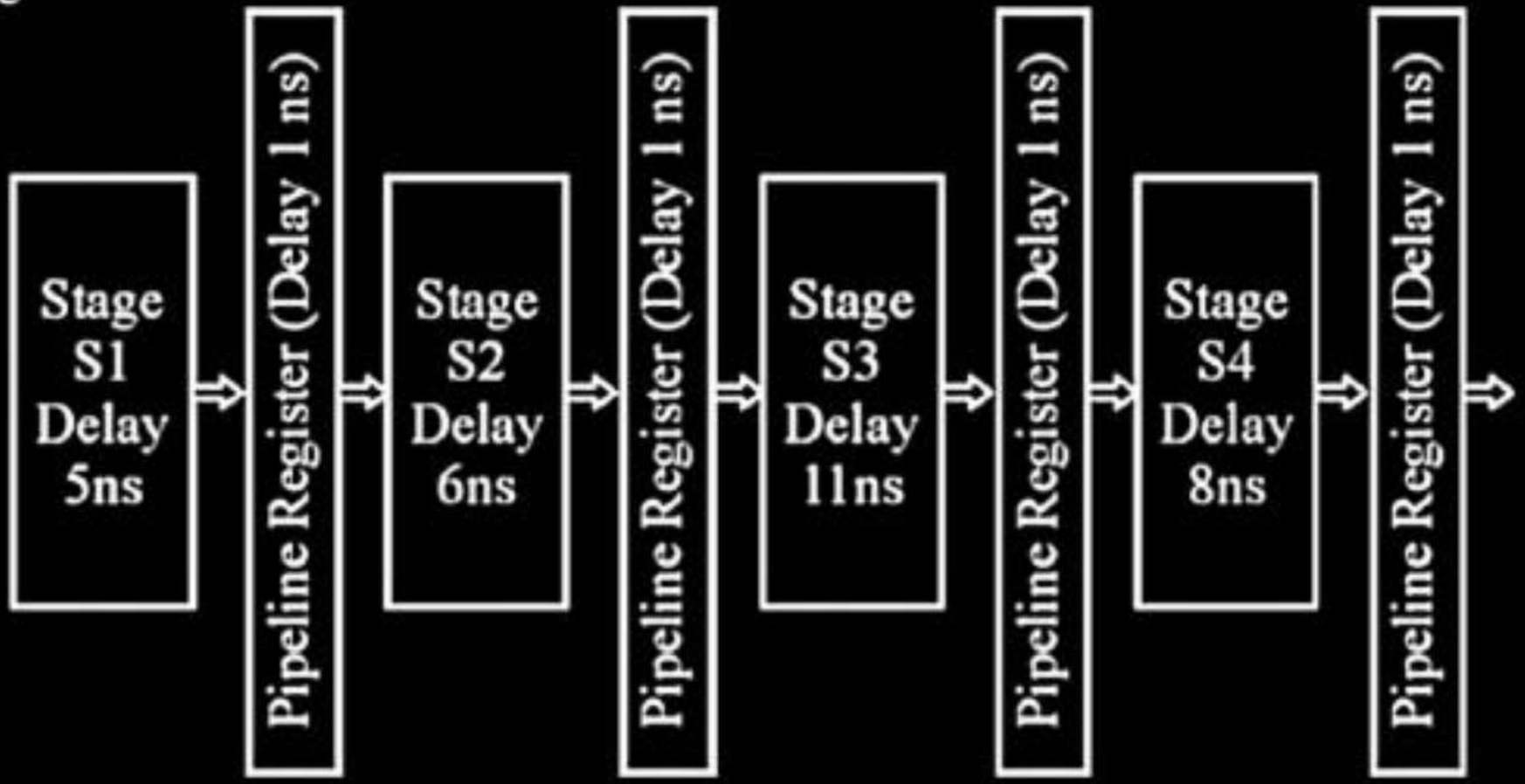
A 4.5

B 4.0

C 3.33

D 3.0

Q. 4 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?

[GATE-2011: 2 Marks]

A 4.0

B 2.5

C 1.1

D 3.0

NAT Q. 5

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.

[GATE-2021(Set-1)-CS: 2M]

NAT Q. 6

Consider a 3-stage pipelined processor having a delay of 10 ns (nanoseconds), 20 ns, and 14 ns, for the first, second, and the third stages, respectively. Assume that there is no other delay and the processor does not suffer from any pipeline hazards. Also assume that one instruction is fetched every cycle.

The total execution time for executing 100 instructions on this processor is _____ns.

[GATE-2023-CS: 1M]

NAT Q. 7

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

(I)a naive pipeline implementation (NP) with 5 stages and

(II)an efficient pipeline (EP) where the OF stage is divided into stages OF1 and OF2 with execution times of 12 ns and 8 ns respectively.

The speedup (correct to two decimal places) achieved by EP over NP in executing 20 independent instructions with no hazards is _____.

[GATE-2017(Set-1)-CS: 2M]

THANK YOU!