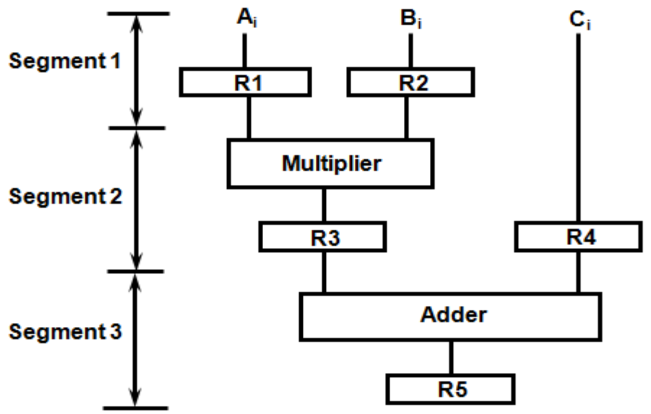
Introduction to Pipeline Operation MODULE-II

Introduction to Pipeline Operation:

- **Pipelining** is a technique of decomposing a sequential process into sub-operations, with each sub-process being executed in a segment that operates concurrently with all other segment.
- Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.
- Pipelining improves system performance.
- **Example for Pipeline Processing:** To perform the combined multiply and add operations with a stream of numbers.

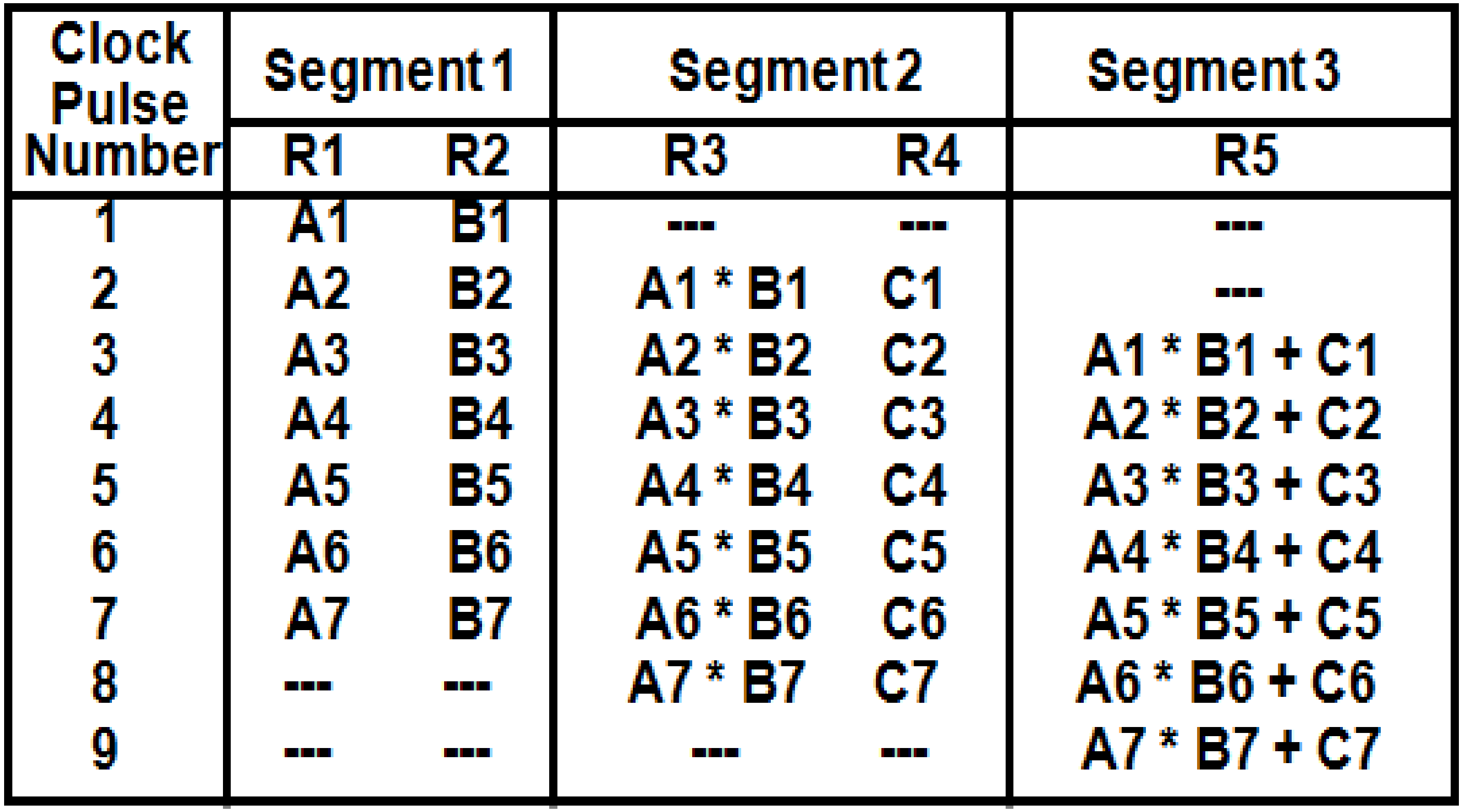
$A_i * B_i + C_i$ for $i = 1, 2, 3, \ldots, 7$



$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Input A_i and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and input C_i
$R5 \leftarrow R3 + R4$	Add C_i to product

Introduction to Pipeline Operation:

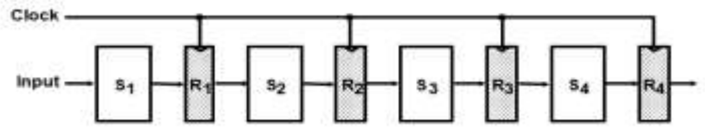
- **Operations in each pipeline stage (Content of Registers in Pipeline Example):**



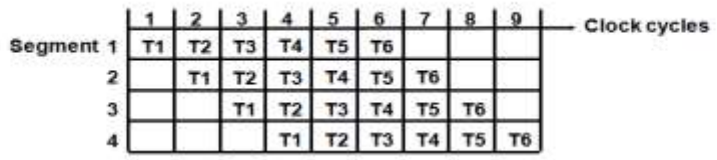
Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1	---	---	---
2	A2	B2	A1 * B1	C1	---
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8	---	---	A7 * B7	C7	A6 * B6 + C6
9	---	---	---	---	A7 * B7 + C7

Introduction to Pipeline Operation:

- **General Four-Segment Pipeline:**
- Each segment consists of a combinational circuit Si that performs a suboperation over the data stream flowing through the pipe. The segments are separated by registers Ri that hold the intermediate results between the stages.
- The behavior of a pipeline can be illustrated with a space-time diagram.
- This is a diagram that shows the segment utilization as a function of time.
- Ex: Four segments and six tasks. The time required to complete all the operations is 4 + (6 - 1) = 9 clock cycles.
- The diagram shows six tasks T1 through T6 executed in four segments. Initially, task T1 is handled by segment 1. After the first clock, segment 2 is busy with T1, while segment 1 is busy with task T2. Continuing in this manner, the first task T1 is completed after the fourth clock cycle.



Space-Time Diagram

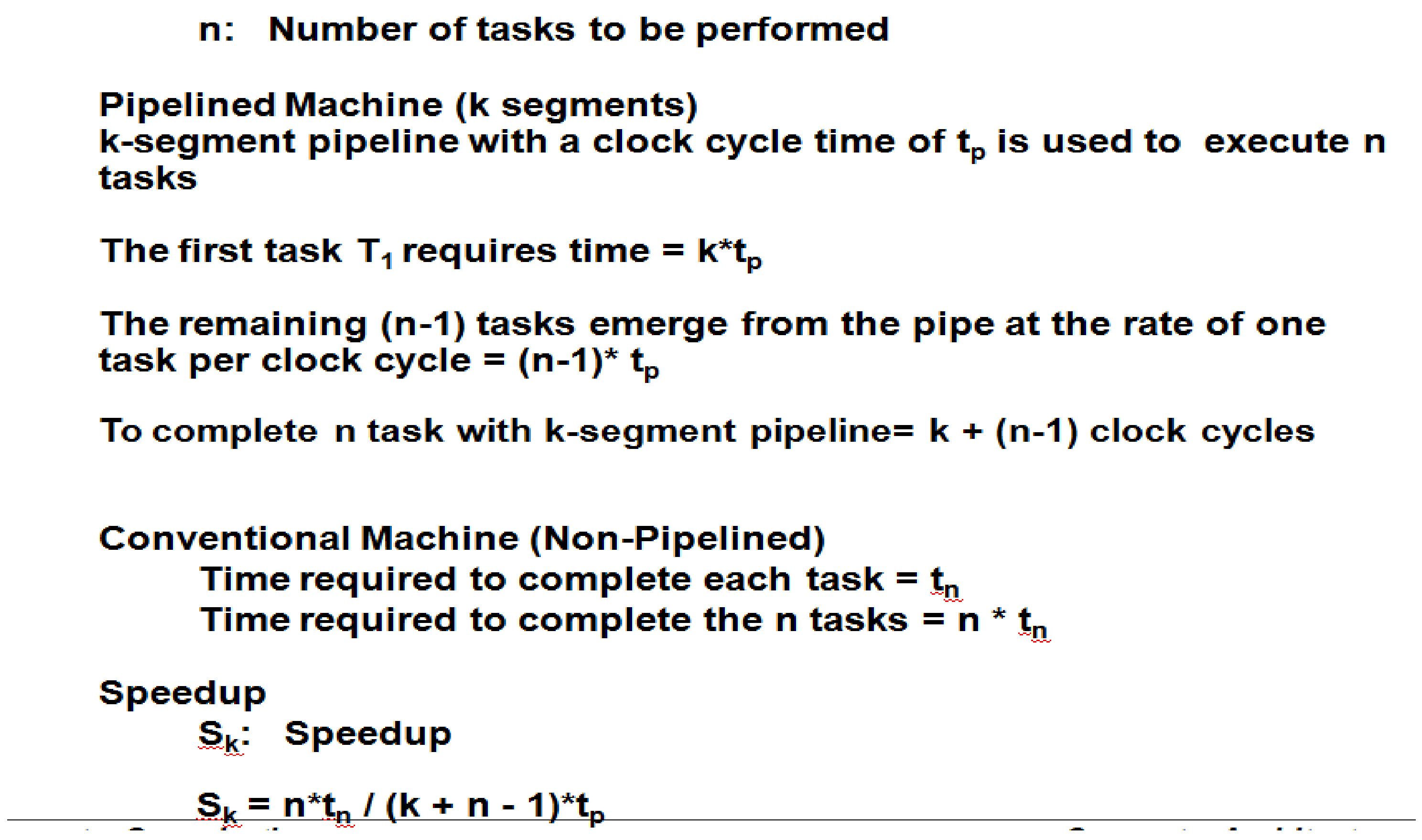


Segment	1	2	3	4	5	6	7	8	9
1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

Clock cycles

Introduction to Pipeline Operation:

- **Pipeline Speedup:** Now consider the case where a k-segment pipeline with a clock cycle time t_p, is used to execute n tasks. The speedup of a pipeline processing over an equivalent nonpipelined processing is defined by the ratio S:



n: Number of tasks to be performed

Pipelined Machine (k segments)
k-segment pipeline with a clock cycle time of t_p is used to execute n tasks

The first task T_1 requires time = $k*t_p$

The remaining (n-1) tasks emerge from the pipe at the rate of one task per clock cycle = $(n-1)*t_p$

To complete n task with k-segment pipeline= k + (n-1) clock cycles

Conventional Machine (Non-Pipelined)
Time required to complete each task = t_n
Time required to complete the n tasks = $n * t_n$

Speedup
S_k: Speedup

$S_k = n*t_n / (k + n - 1)*t_p$

Introduction to Pipeline Operation:

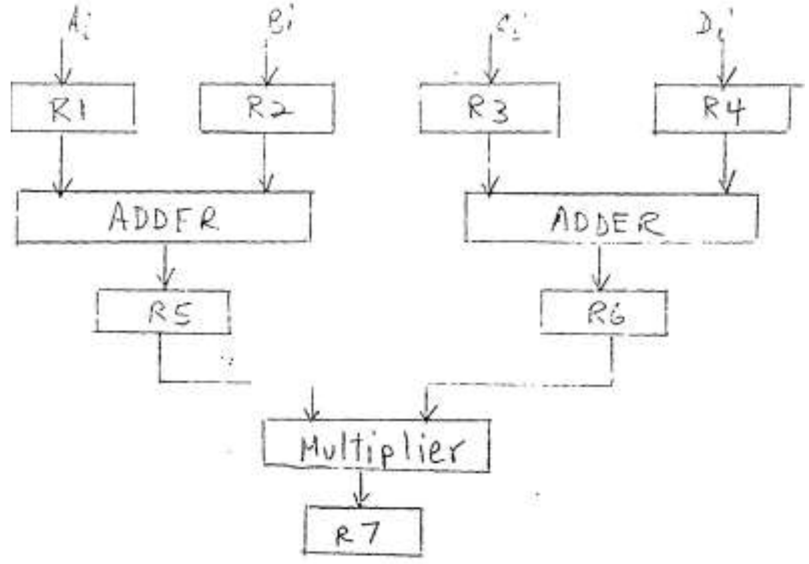
- **Pipeline Speedup:**
- As the number of tasks increases, n becomes much larger than k - 1, and k + n - 1 approaches the value of n. Under this condition, the speedup becomes: $S = t_n/t_p$
- If we assume that the time it takes to process a task is the same in the pipeline and non pipeline circuits, we will have $t_n = kt_p$. Including this assumption, the speedup reduces to: $S = kt_p/t_p = k$
- This shows that the theoretical maximum speedup that a pipeline can provide is k, where k is the number of segments in the pipeline.

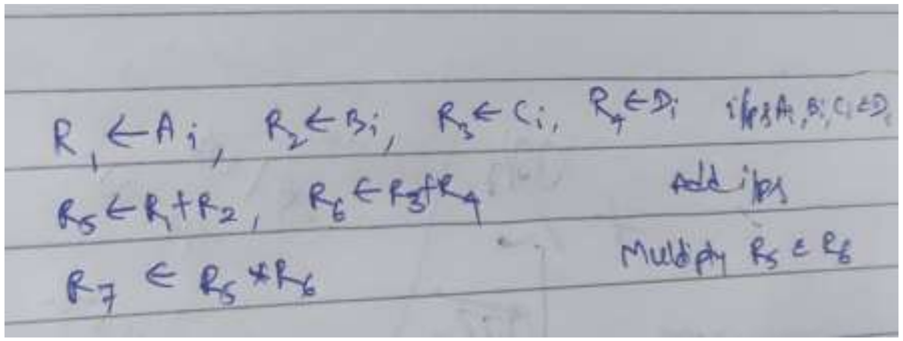
Introduction to Pipeline Operation:

Q1. In certain scientific computation it is necessary to perform the arithmetic operation (Ai+Bi)*(Ci+Di)

a. Specify the pipeline configuration to carry out this task.

b. List the content of all the registers in the pipeline for i=1 to 6.

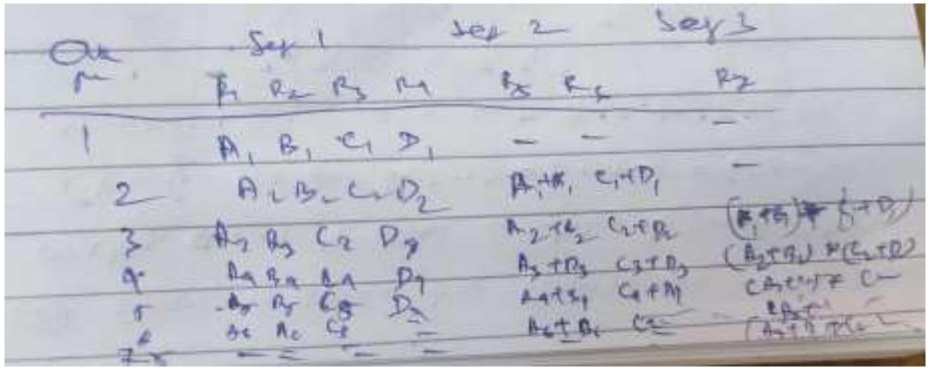




$R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$, $R_3 \leftarrow C_i$, $R_4 \leftarrow D_i$ i/p A, B, C, D,

$R_5 \leftarrow R_1 + R_2$, $R_6 \leftarrow R_3 + R_4$ Add i/ps

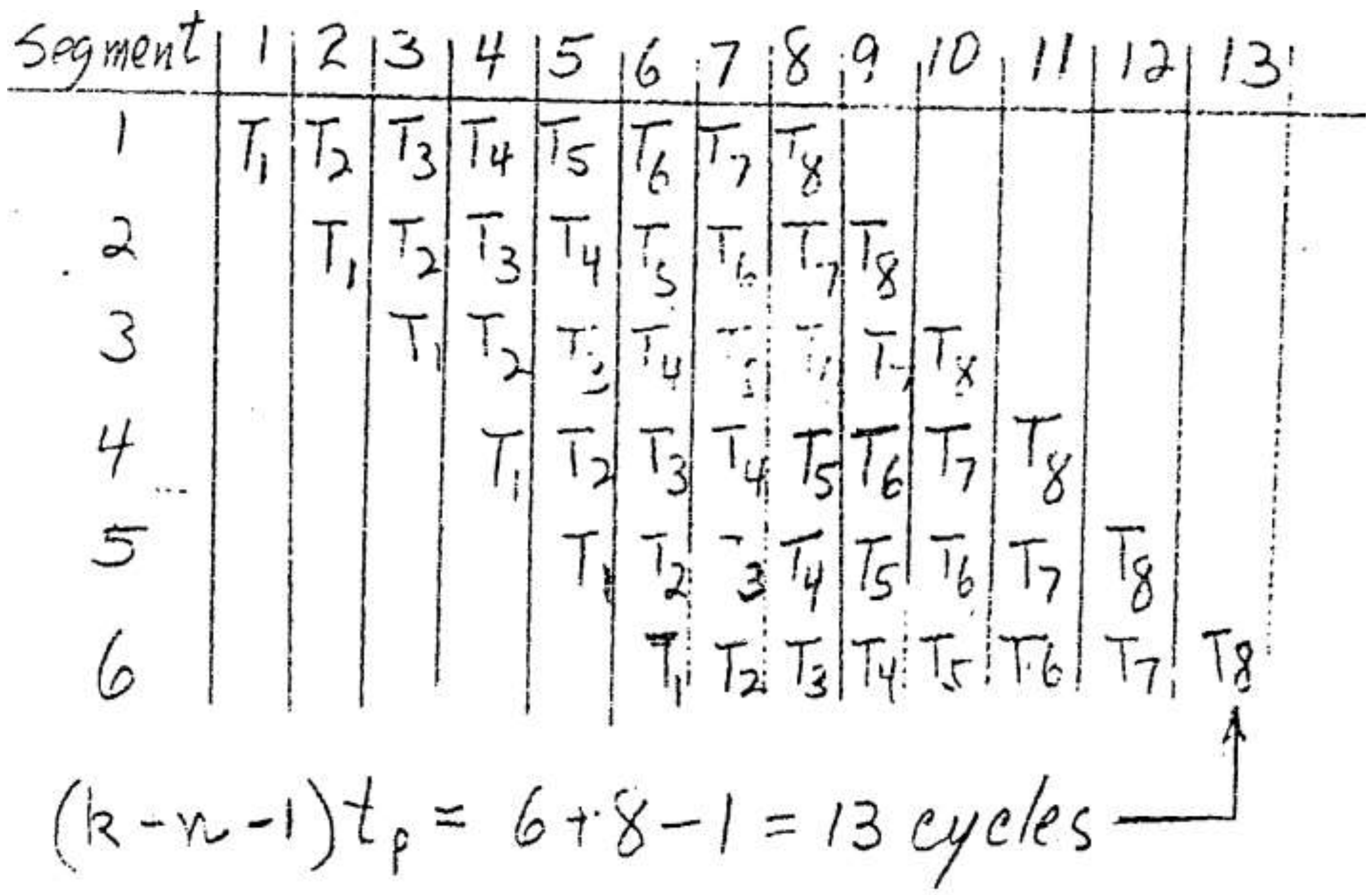
$R_7 \leftarrow R_5 * R_6$ Multiply R_5 & R_6



	Seg 1				Seg 2		Seg 3
Clk	R_1	R_2	R_3	R_4	R_5	R_6	R_7
1	A_1	B_1	C_1	D_1	—	—	—
2	A_2	B_2	C_2	D_2	A_1+B_1	C_1+D_1	—
3	A_3	B_3	C_3	D_3	A_2+B_2	C_2+D_2	$(A_1+B_1)*(C_1+D_1)$
4	A_4	B_4	C_4	D_4	A_3+B_3	C_3+D_3	$(A_2+B_2)*(C_2+D_2)$
5	A_5	B_5	C_5	D_5	A_4+B_4	C_4+D_4	[illegible]
6	A_6	B_6	C_6	—	A_5+B_5	C_5	[illegible]
7	—	—	—	—			

Introduction to Pipeline Operation:

Q2. Draw a space time diagram for a six-segment pipeline showing the time it takes to process eight tasks. (k =6, n=8, so cycles= k+n-1)



Segment	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

Introduction to Pipeline Operation:

Q3. A non-pipelined system takes 50ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10ns. Determine the speedup ratio of the pipeline for 100 tasks.

$t_n = 50\ \text{ns}$

$k = 6$

$t_p = 10\ \text{ns}$

$n = 100$

$$S = \frac{n t_n}{(k+n-1)t_p} = \frac{100 \times 50}{(6-99) \times 10} = 4.76$$

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

Arithmetic Pipeline:

- ❖ Pipeline arithmetic units are usually found in very high speed computers.
- ❖ They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.
- ❖ **Example:** A pipeline unit for floating-point addition and subtraction.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers.

$$X = A \times 2^a$$

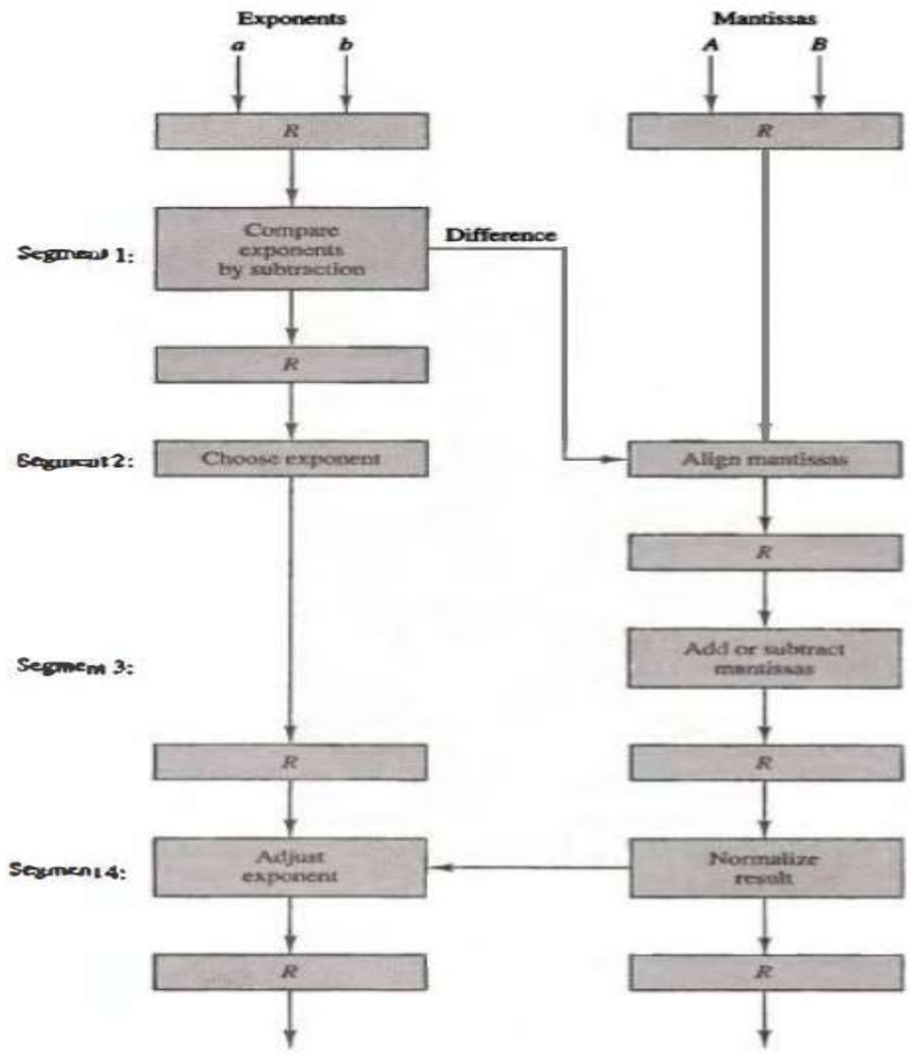
$$Y = B \times 2^b$$

- ❖ A and B are two fractions that represent the mantissas and a and b are the exponents. The floating-point addition and subtraction can be performed in 4 segments.
- ❖ The registers labeled R are placed between the segments to store intermediate results. The suboperations that are performed in the 4 segments are:

1. Compare the exponents.
2. Align the mantissas.
3. Add or subtract the mantissas.
4. Normalize the result.

Arithmetic Pipeline:

❖ A pipeline unit for floating-point addition and subtraction in binary format.



Arithmetic Pipeline:

- ❖ Consider the two normalized floating-point numbers:

  $X = 0.9504 \times 10^3$ and $Y = 0.8200 \times 10^2$

- ❖ The two exponents are subtracted in the first segment to obtain 3 - 2 = 1. The larger exponent 3 is chosen as the exponent of the result. The next segment shifts the mantissa of Y to the right to obtain

  $X = 0.9504 \times 10^3$ and $Y = 0.0820 \times 10^3$

- ❖ This aligns the two mantissas under the same exponent. The addition of the two mantissas in segment 3 produces the sum

  $$Z = 1.0324 \times 10^3$$

- ❖ The sum is adjusted by normalizing the result so that it has a fraction with a nonzero first digit. This is done by shifting the mantissa once to the right and incrementing the exponent by one to obtain the normalized sum.

  $$Z = 0.10324 \text{ X } 10^4$$

- Suppose that the time delays of the four segments are t1 = 60 ns, t2 = 70 ns, t3 = 100 ns, t4 = 80 ns, and the interface registers have a delay of tr = 10 ns. The clock cycle is chosen to be tp = t3 + tr = 110 ns.
- An equivalent non pipeline floating point adder-subtractor will have a delay time tn = t1 + t2 + t3 + t4 + tr = 320 ns. In this case the pipelined adder has a speedup of 320/110 = 2. 9 over the nonpipelined adder.

Arithmetic Pipeline Problem:

Q1. Consider a pipeline having 4 phases with duration 60, 50, 90 and 80 ns. Given latch delay is 10 ns. Calculate (a) Pipeline cycle time (b) Non-pipeline execution time (c) Speed up ratio (d) Pipeline time for 1000 tasks (e) Sequential time for 1000 tasks (f) Throughput.

Sol.: Four stage pipeline is used.

Delay of stages = 60, 50, 90 and 80 ns

Latch delay or delay due to each register = 10 ns

Part-01: Pipeline Cycle Time-

Cycle time = Maximum delay due to any stage + Delay due to its register

= Max { 60, 50, 90, 80 } + 10 ns = 90 ns + 10 ns = 100 ns

Part-02: Non-Pipeline Execution Time-

Non-pipeline execution time for one instruction = 60 ns + 50 ns + 90 ns + 80 ns + 10 ns = 290 ns

Part-03: Speed Up Ratio-

Speed up = Non-pipeline execution time / Pipeline execution time

= 280 ns / Cycle time = 290 ns / 100 ns = 2.9

Part-04: Pipeline Time For 1000 Tasks-

Pipeline time for 1000 tasks = Time taken for 1st task + Time taken for remaining 999 tasks

= 1 x 4 clock cycles + 999 x 1 clock cycle = 4 x cycle time + 999 x cycle time

= 4 x 100 ns + 999 x 100 ns = 400 ns + 99900 ns = 100300 ns

Part-05: Sequential Time For 1000 Tasks-

Non-pipeline time for 1000 tasks = 1000 x Time taken for one task

= 1000 x 290 ns = 290000 ns

Part-06: Throughput-

Throughput for pipelined execution = Number of instructions executed per unit time

= 1000 tasks / 100300 ns

THANK YOU