COMPUTER ORGANIZATION
MODULE-II (Pipeline)

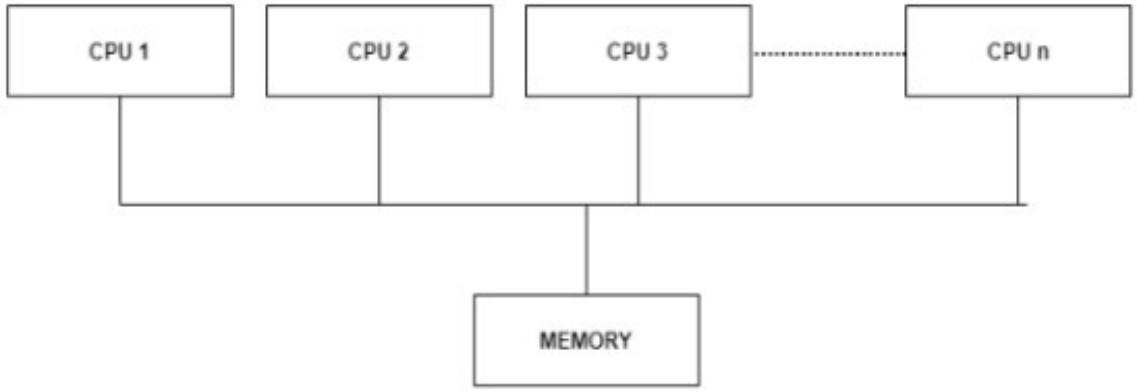
Module-II	**Multiprogramming and Multiprocessing**	R1	2	Introduction to pipeline operation, space-time diagram, speed-up ratio
	Control Unit	R1	1	Instruction types and formats
		https://www.bau.edu.jo/UserPortal/UserProfile/PostsAttach/43038_4306	3	Instruction cycle, execution of a complete instruction.
		R2	1	Hardwired and micro programmed control unit, unconditional and conditional branching.
		R1	1	Microinstruction with next address field, pre-fetching microinstructions, Concept of horizontal and vertical microprogramming.
	Memory	https://www.youtube.com/watch?v=YHfulm7Tub	2	Basic concept of Memory and its hierarchy,
		R1	2	RAM memories, 2D, 2 & 1/2D memory organization. ROM memories.
		R1	2	Cache memories: concept and design issues, performance, address mapping
		R1	1	page replacement algorithms
		R2	2	Virtual memory: concept and implementation.
	Input/Output	R2	1	Peripheral devices, I/O interface, I/O ports,
		R2	2	Interrupts: interrupt hardware, types of interrupts and exceptions.
		R2	1	Buses, bus architecture, types of buses and bus arbitration.
		R2	2	Modes of Data Transfer: Programmed I/O, interrupt initiated I/O and Direct Memory Access., I/O channels and processors.
		R2	1	Standard communication interfaces.

OUTLINE

- **M**ultiprogramming and Multiprocessing
- **P**arallel Processing
- **F**lynn's classification
- **I**ntroduction to Pipeline Operation

Multiprogramming and Multiprocessing:

- **Multiprocessor** systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc.



- 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.

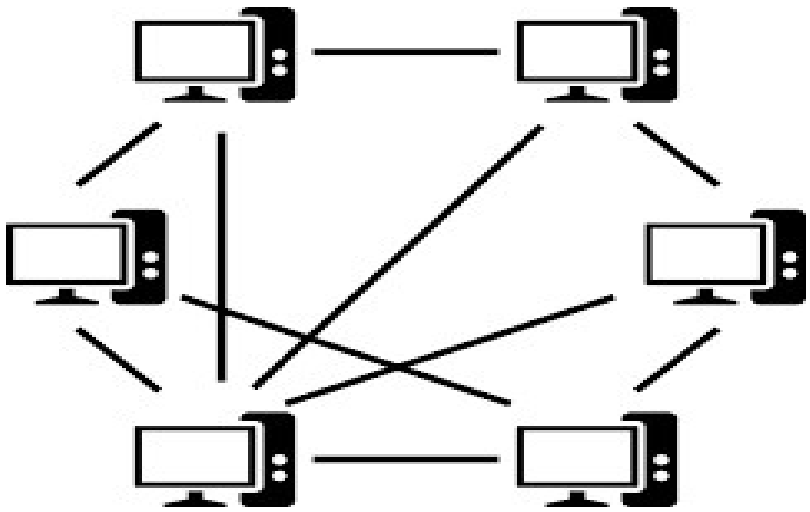
- ✓ **Multiprogramming** is a rudimentary form of parallel processing in which several programs run at the same time on a uniprocessor system. The purpose of multiprogramming is to maximize CPU time.

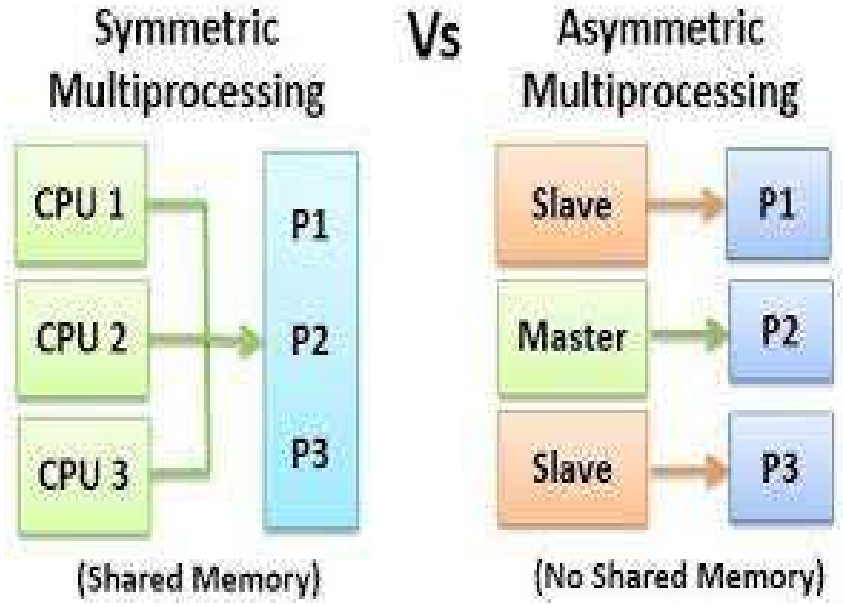
Multiprocessor Architecture:

Types of Multiprocessors:

- There are mainly two types of multiprocessors i.e. symmetric and asymmetric

Symmetric Multiprocessors: Each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master-slave relationship exists between them.





Asymmetric Multiprocessors: In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. It contains a master slave relationship.

Multiprocessor Architecture:

Advantages of Multiprocessor Systems:

- More reliable Systems: In a multiprocessor system, even if one processor fails, the system will not halt.
- Enhanced Throughput: If multiple processors are working in tandem (one behind another), then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.

Disadvantages of Multiprocessor Systems:

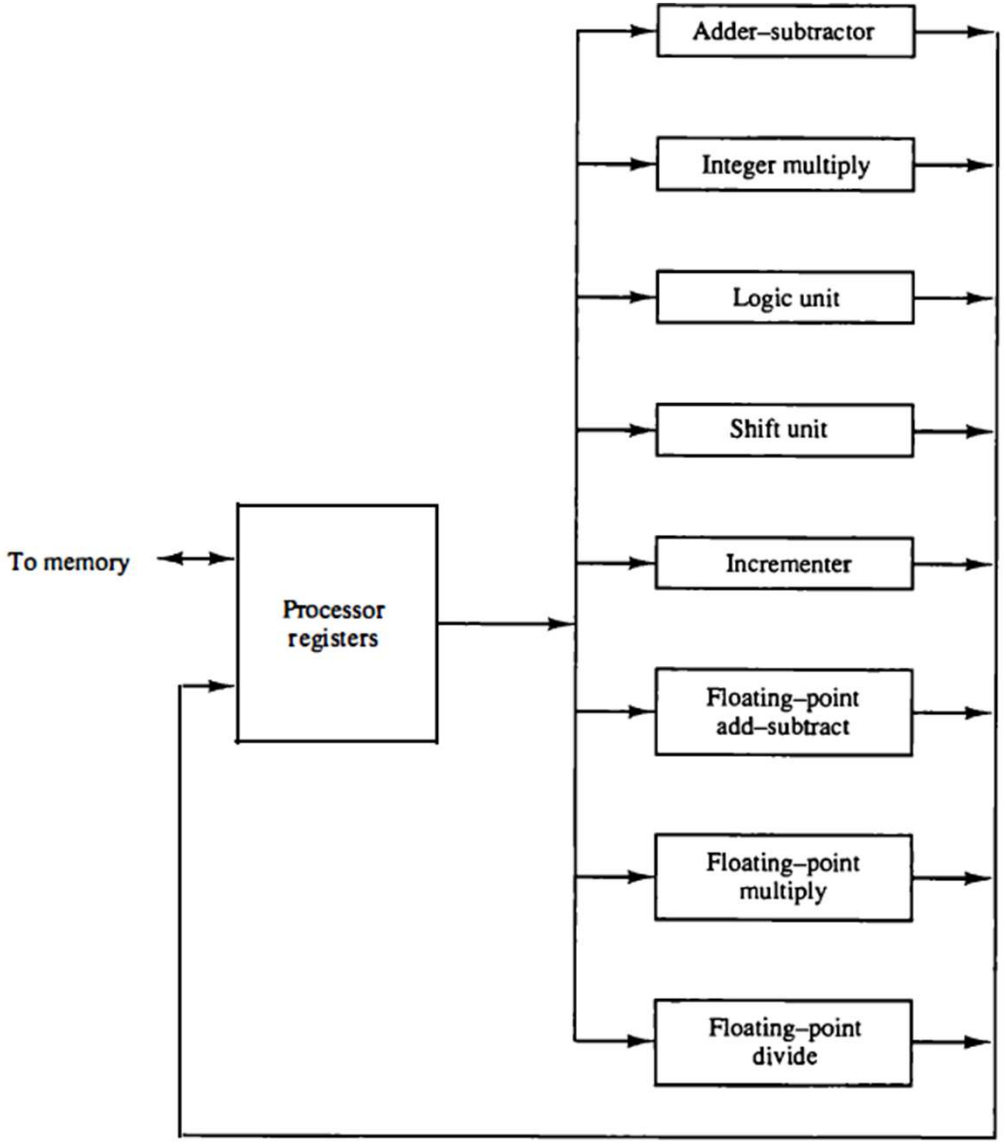
- Increased Expense: It is quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.
- Complicated Operating System Required: There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart resources to processes than in single processor systems.
- Large Main Memory Required: All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

Parallel Processing:

- Parallel processing is used to provide simultaneous data-processing tasks for the purpose of **increasing the computational speed** of a computer system.
- Instead of processing each instruction sequentially as in a conventional computer, a parallel processing system is able to perform concurrent data processing to achieve **faster execution time**.
- For example, while an instruction is being **executed in the ALU**, the next instruction **can be read from memory**.
- The purpose of parallel processing is to **speed up the computer processing capability and increase its throughput**, that is, the amount of processing that can be accomplished during a given interval of time.
- The **amount of hardware increases** with parallel processing. and with it, the cost of the system increases. However, technological developments have reduced hardware costs to the point where parallel processing techniques are economically feasible.

Parallel Processing:

- Below figure shows one possible way of separating the execution unit into eight functional units operating in parallel.

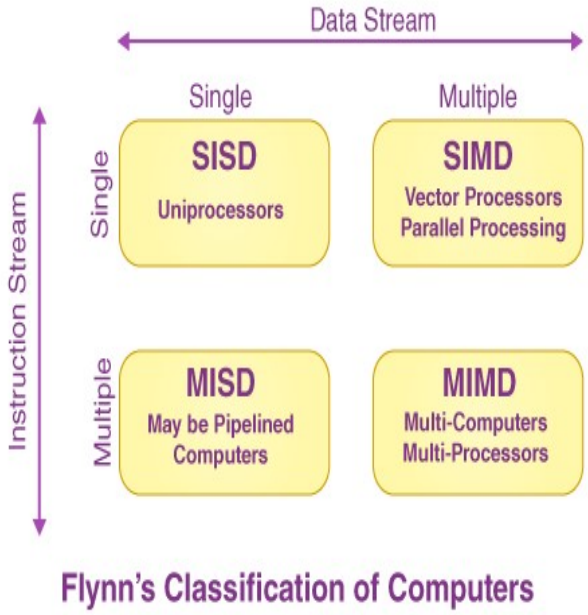


Parallel Processing:

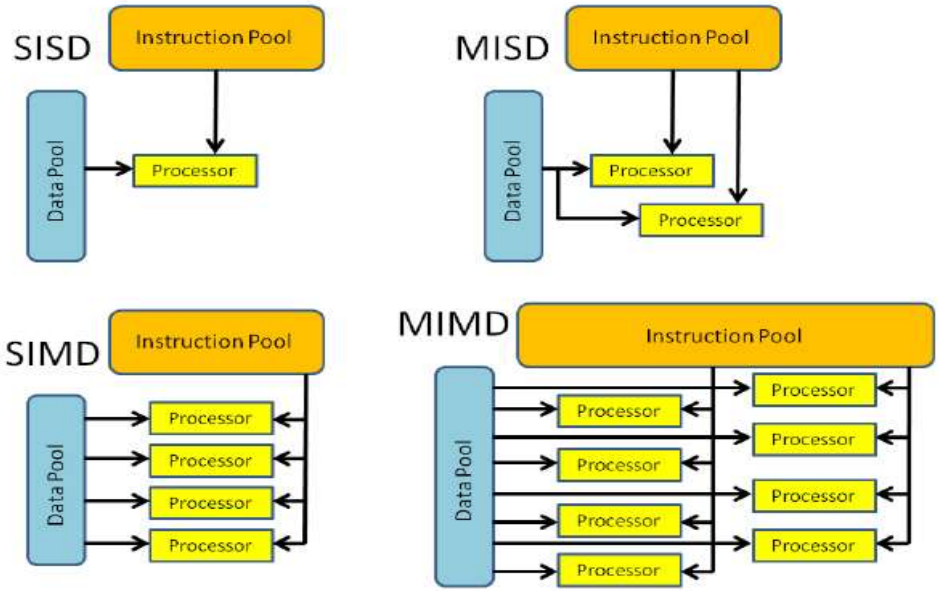
- There are a variety of ways that parallel processing can be classified. It can be considered from the internal organization of the processors, from the interconnection structure between processors, or from the flow of information through the system.
- One classification introduced by **M. J. Flynn** considers the organization of a computer system by the **number of instructions and data items** that are manipulated simultaneously.
- The normal operation of a computer is to fetch instructions from memory and execute them in the processor.
- The sequence of instructions read from memory constitutes an **instruction stream.**
- The operations performed on the data in the processor constitutes a **data stream.**
- Parallel processing may occur in the instruction stream, in the data stream, or in both.

Flynn's classification:

- Flynn's classification divides computers into four major groups as follows:
- **Single instruction stream, single data stream (SISD)**
- **Single instruction stream, multiple data stream (SIMD)**
- **Multiple instruction stream, single data stream (MISD)**
- **Multiple instruction stream, multiple data stream (MIMD)**



Flynn's Classification of Computers

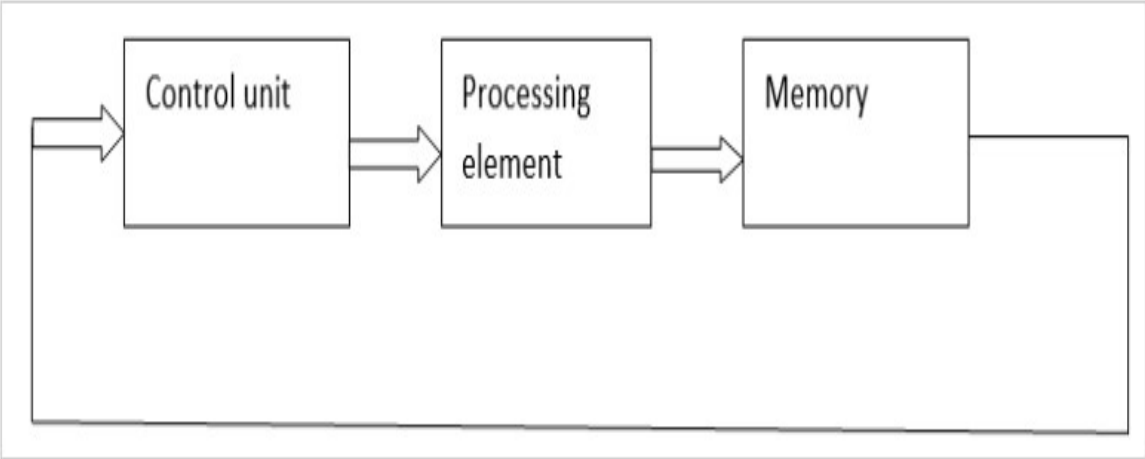


Flynn's classification:

- **Single instruction stream, single data stream (SISD):**

Single instruction: Only one instruction stream is being acted or executed by CPU during one clock cycle.

Single data stream: Only one data stream is used as input during one clock cycle.



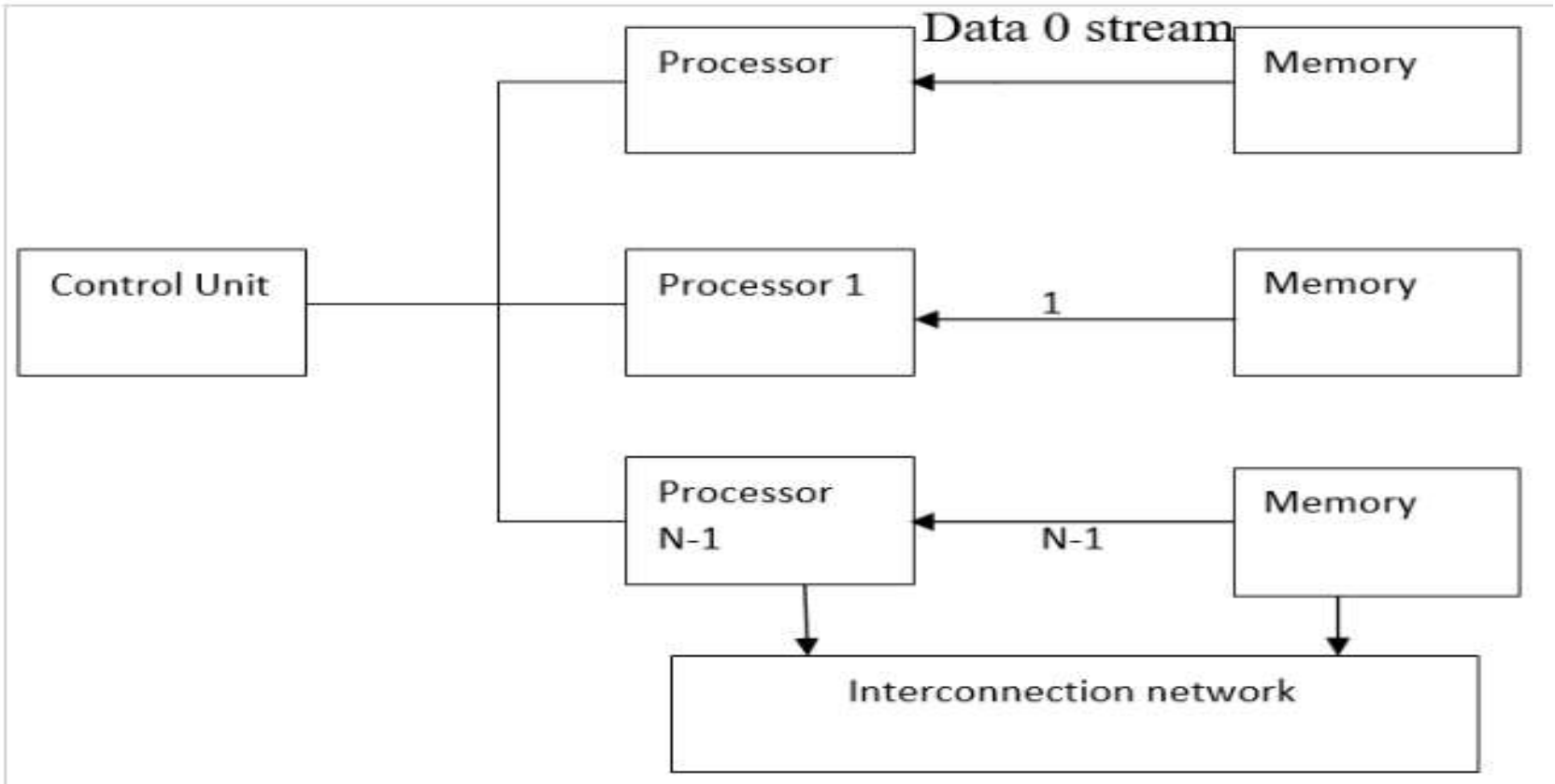
- ✓ A SISD computing system is a uniprocessor machine that is capable of executing a single instruction operating on a single data stream.
- ✓ Most conventional computers have SISD architecture where all the instruction and data to be processed have to be stored in primary memory.
- ✓ **Example**: Von Neumann types of computers.

Flynn's classification:

- **Single instruction stream, multiple data stream (SIMD):**

A SIMD system is a multiprocessor machine, capable of executing the same instruction on all the CPUs but operating on the different data stream. **Example**: Vector computers, array computers

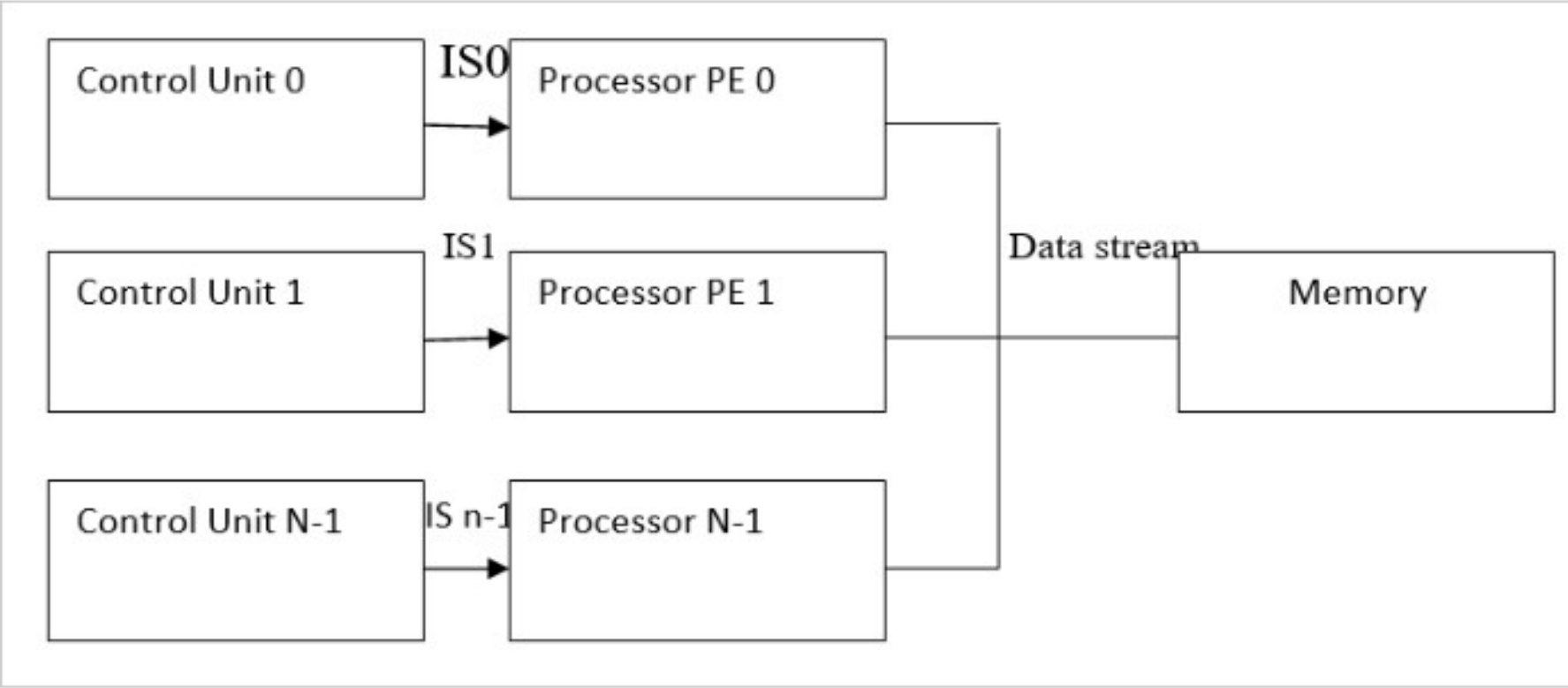
IBM 710 is the real life application of SIMD.



Flynn's classification:

- **Multiple instruction stream, single data stream (MISD):**

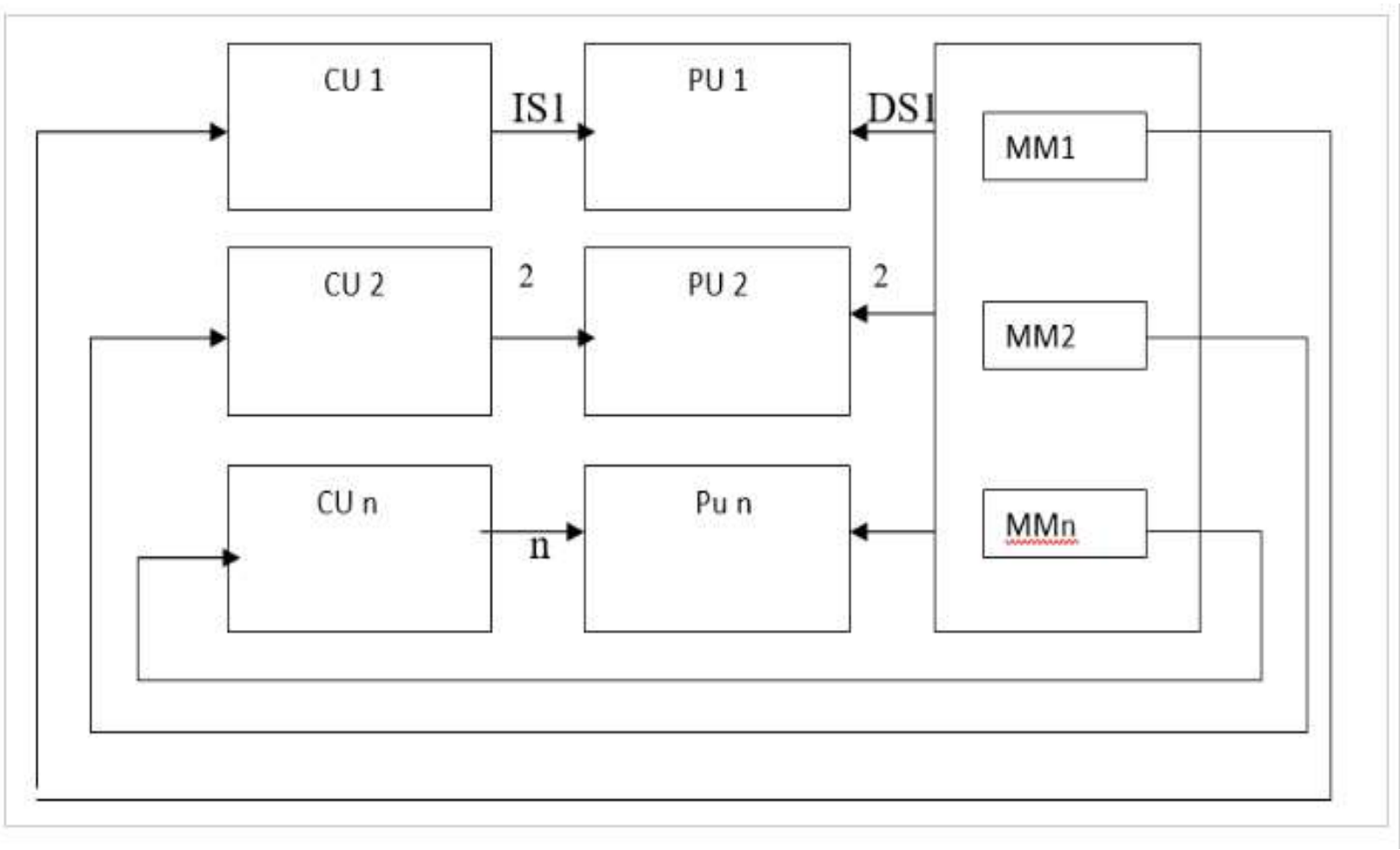
An MISD computing is a multiprocessor machine capable of executing different instructions on processing elements but all of them operating on the same data set.



Flynn's classification:

- **Multiple instruction stream, multiple data stream (MIMD):**

A MIMD system is a multiprocessor machine that is capable of executing multiple instructions over multiple data streams. Each processing element has a separate instruction stream and data stream. **Example:** Distributed computers

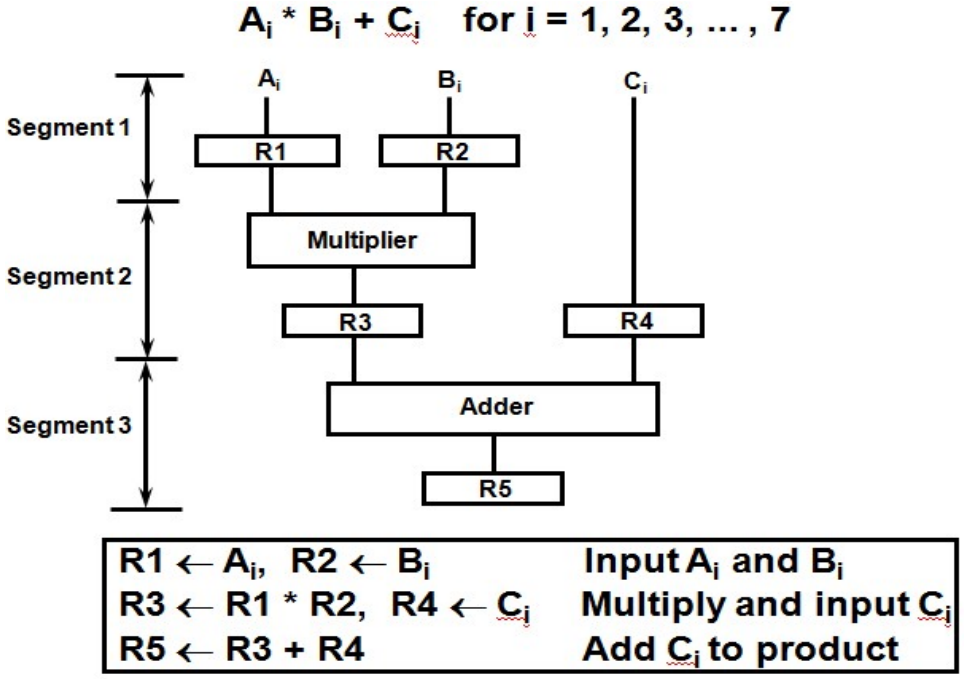


Parallel Processing:

- Flynn's classification depends on the distinction between the performance of the control unit and the data-processing unit. It emphasizes the behavioral characteristics of the computer system rather than its operational and structural interconnections.
- One type of parallel processing that does not fit Flynn's classification is pipelining. The only two categories used from this classification are SIMD array processors and MIMD multiprocessors presented.
- We consider parallel processing under the following main topics:
    1. Pipeline processing
    2. Vector processing
    3. Array processors
- **Pipeline processing** is an implementation technique where arithmetic suboperations or the phases of a computer instruction cycle overlap in execution.
- **Vector processing** deals with computations involving large vectors and matrices.
- **Array processors** perform computations on large arrays of data.

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.



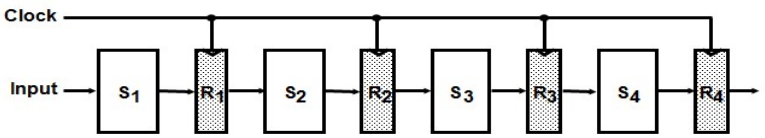
Introduction to Pipeline Operation:

- **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 is called pipeline speedup:

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:

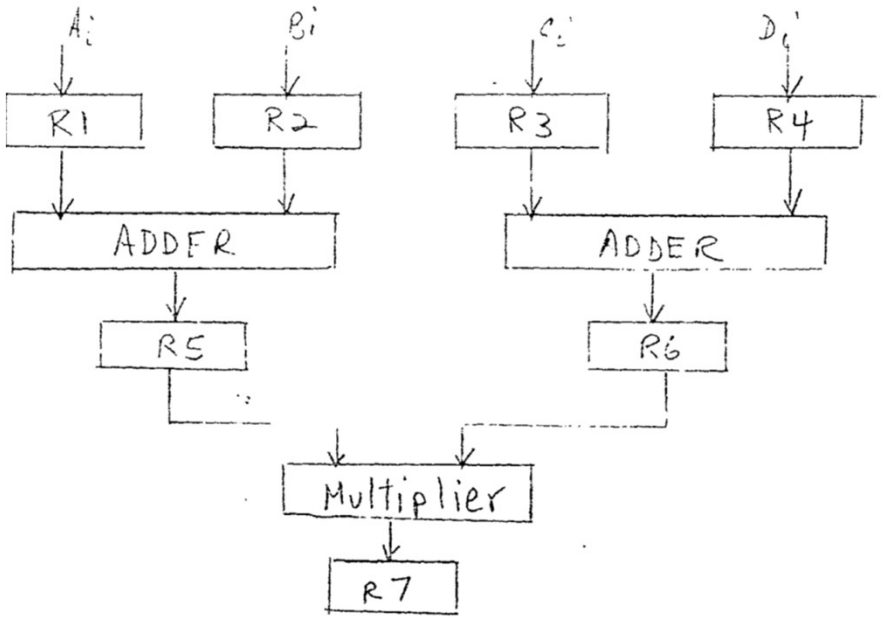
- **Pipeline Speedup:**
- As the number of tasks increases, n becomes much larger than k - 1, and k + n - 1 approaches the value of n. Under this condition, the speedup becomes: $S = t_n/t_p$
- If we assume that the time it takes to process a task is the same in the pipeline and non pipeline circuits, 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.

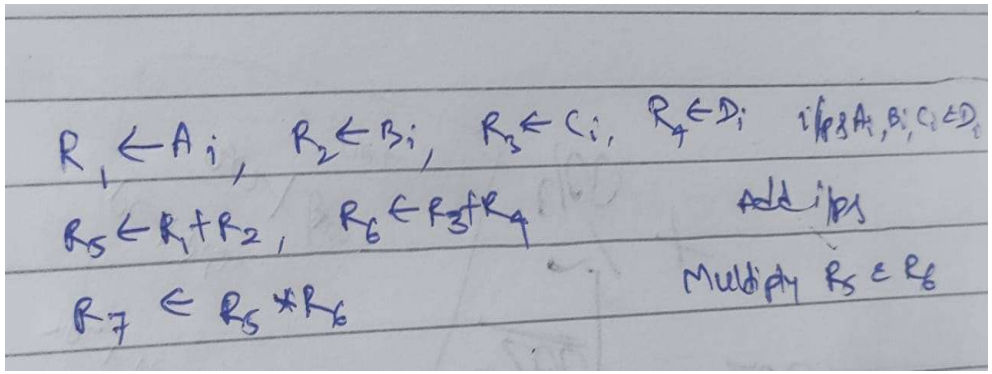
Introduction to Pipeline Operation:

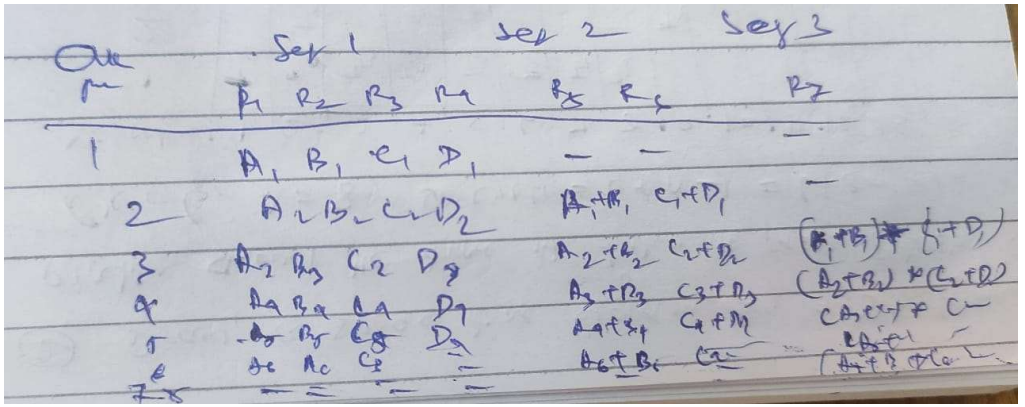
Q1. In certain scientific computation it is necessary to perform the arithmetic operation (Ai+Bi)*(Ci+Di)

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.

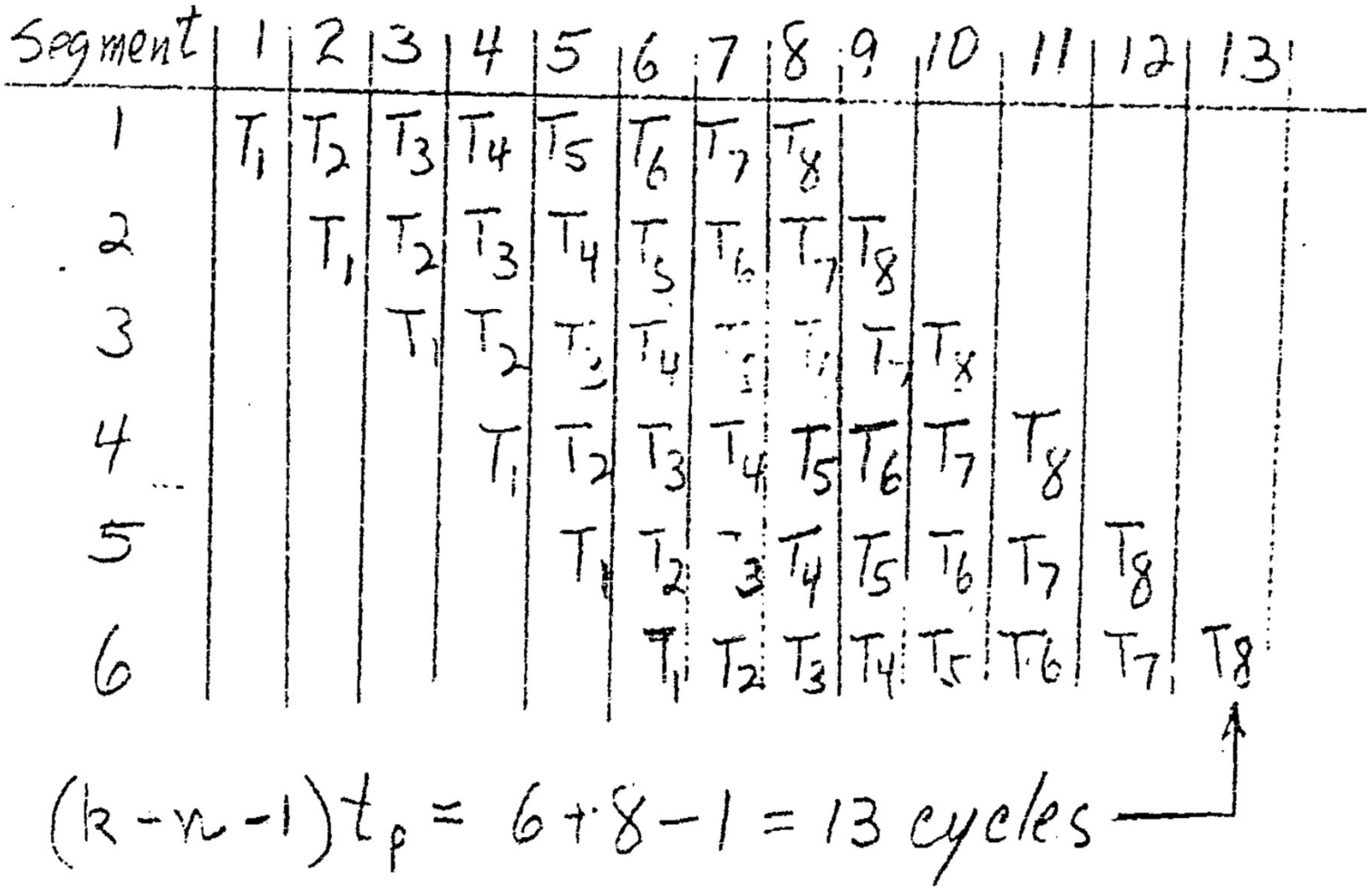






Introduction to Pipeline Operation:

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)



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.

$t_n = 50\ ns$

$k = 6$

$t_p = 10\ ns$

$n = 100$

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

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

Arithmetic Pipeline 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

= 290 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