University Of Diyala
College Of Engineering
Computer Engineering Department

COMPUTER ARCHITECTURE I

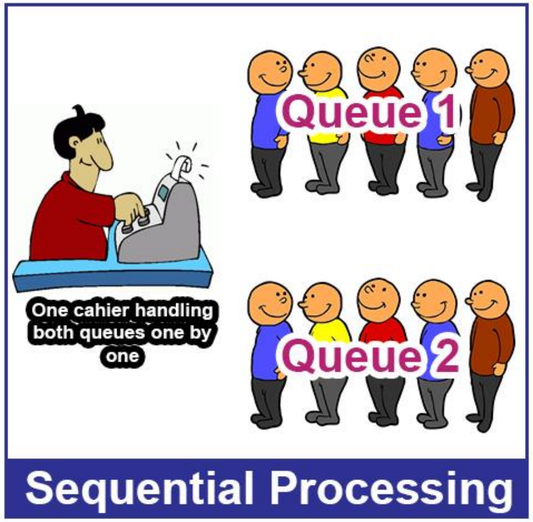
PART 9: PARALLEL PROCESSING

Asst. Prof. Ahmed Salah Hameed
Second stage
2022-2023

Parallel Processing

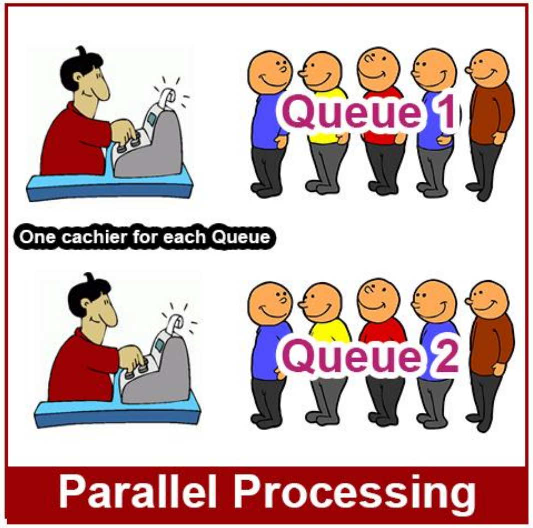
Serial processing (sequential)

It is a processing of completing one task at a time. The processor can not complete more than one task at the time and they are run in a sequence.



Parallel processing (concurrent)

is a term used to denote a large class of techniques that are used to provide simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system.



http://www.itrelease.com/2017/11/difference-serial-parallel-processing/

Parallel Processing

- **The purpose of Parallel processing:**

1. **Speed up the computer processing capability**
2. **Increase the throughput**

Throughput the amount of processing that can be accomplished during a given interval of time.

- **The side effects:**

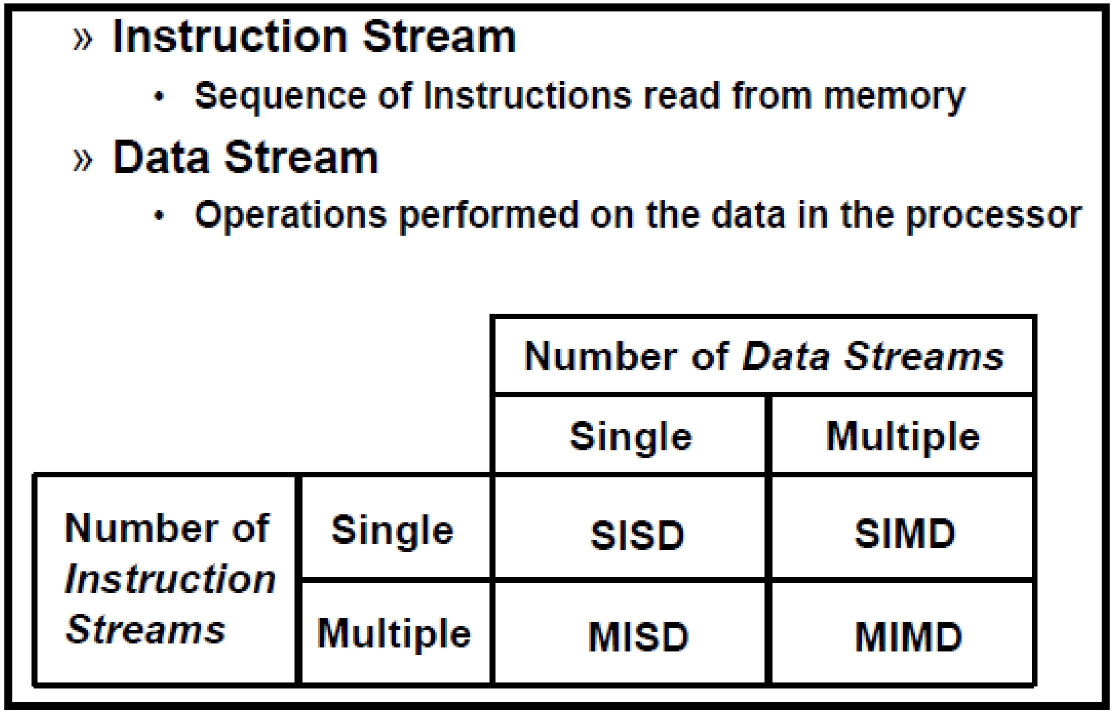
1. **The amount of hardware increases**
2. **The cost of the system increases**

Parallel processing classifications

- **It can be considered**

1. **From the internal organization of the processors**
2. **From the interconnection structure between processors**
3. **From the flow of information through the system.**

Flynn's classification



» **Instruction Stream**

- Sequence of Instructions read from memory

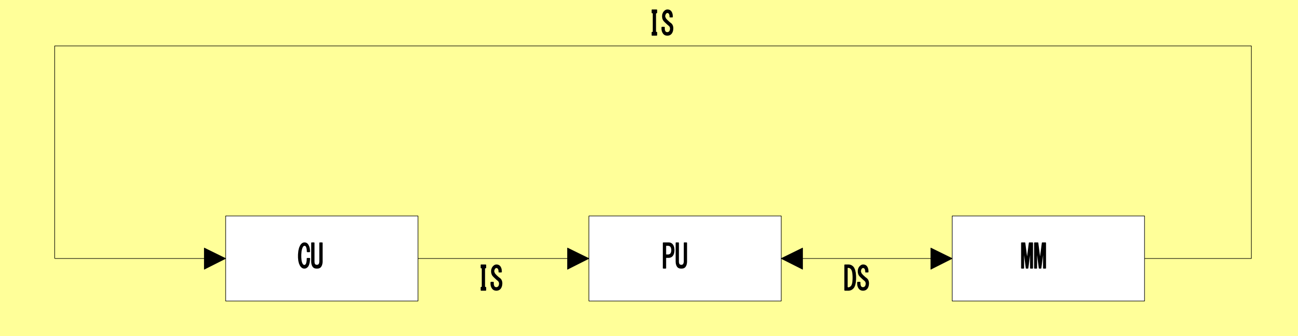
» **Data Stream**

- Operations performed on the data in the processor

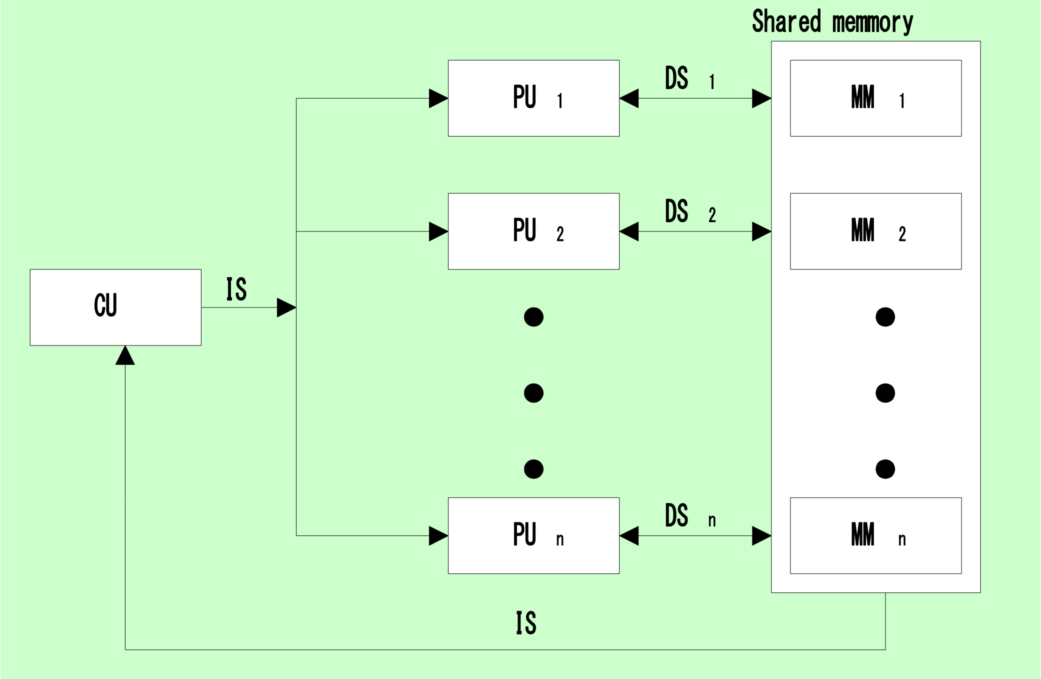
		Number of *Data Streams*	
		Single	Multiple
Number of *Instruction Streams*	Single	SISD	SIMD
	Multiple	MISD	MIMD

Flynn's classification

1. SISD



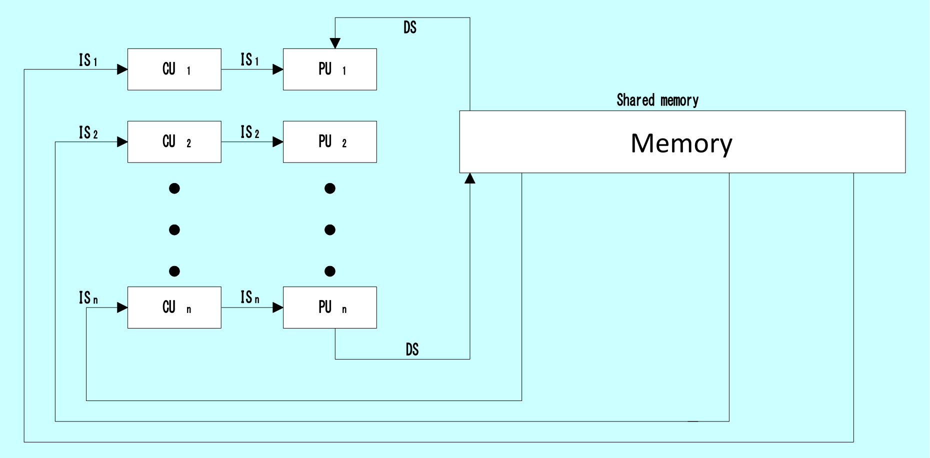
2. SIMD



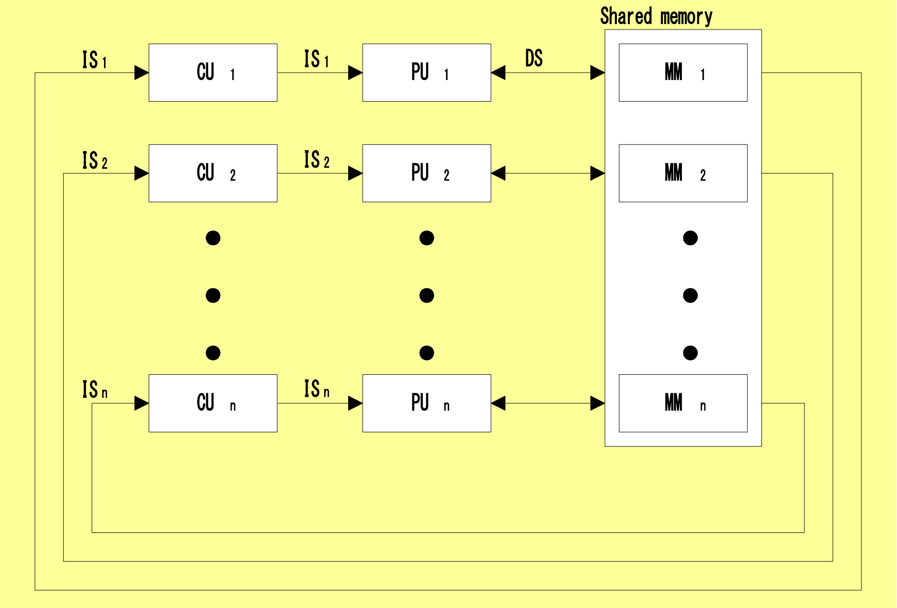
Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed

Flynn's classification

3. MISD



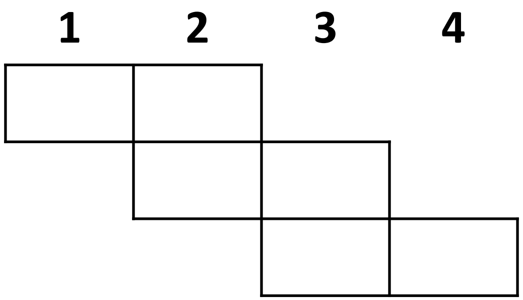
4. MIMD



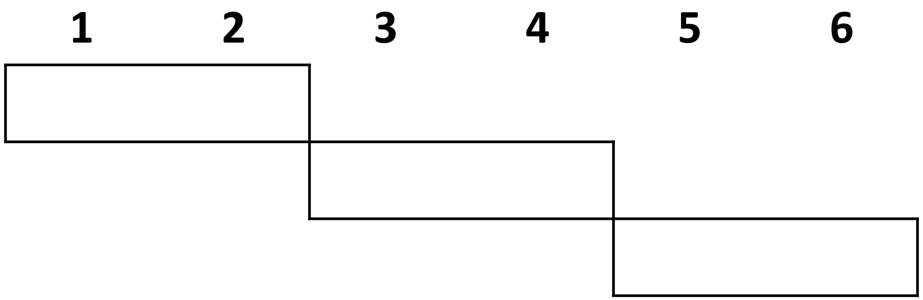
Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed

Pipelining

- **Pipelining** is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.



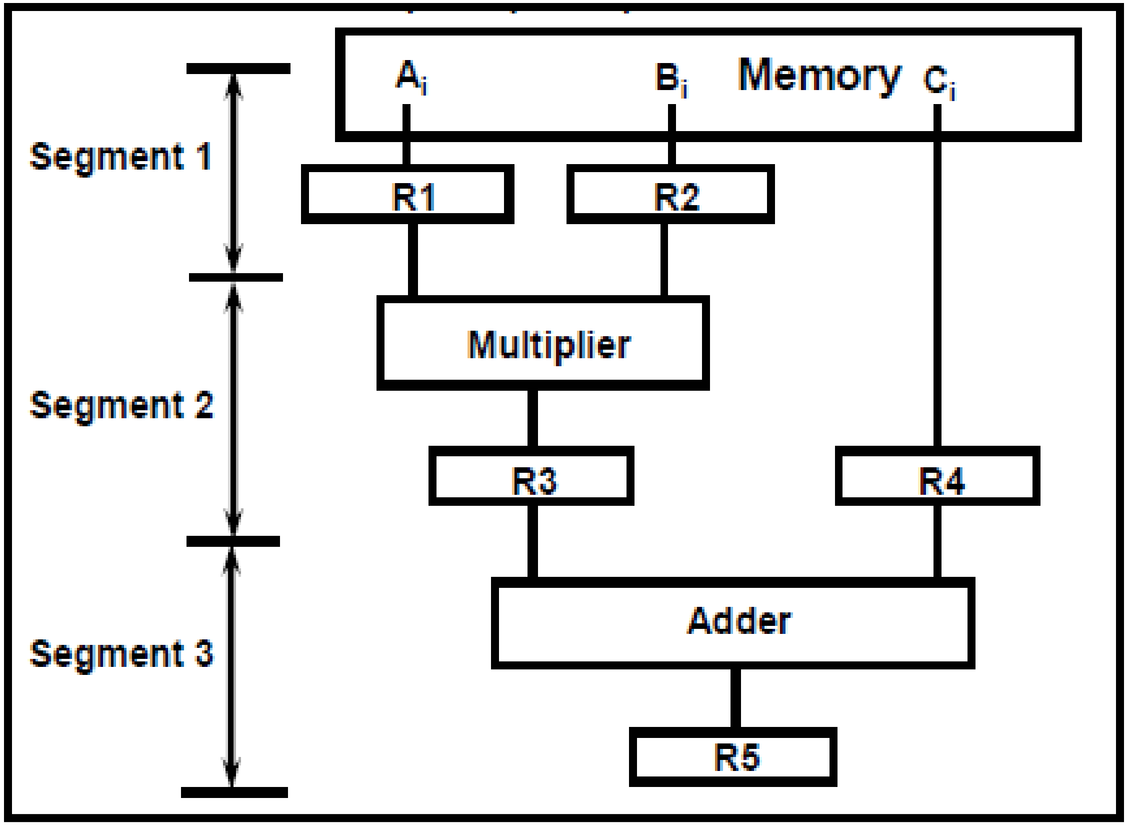
2-segment pipelined



Non pipelined

Example of the Pipeline Organization

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



Based on M. Morris Mano “Computer System Architecture”-- Lecturer Ahmed Salah Hameed

Example of the Pipeline Organization

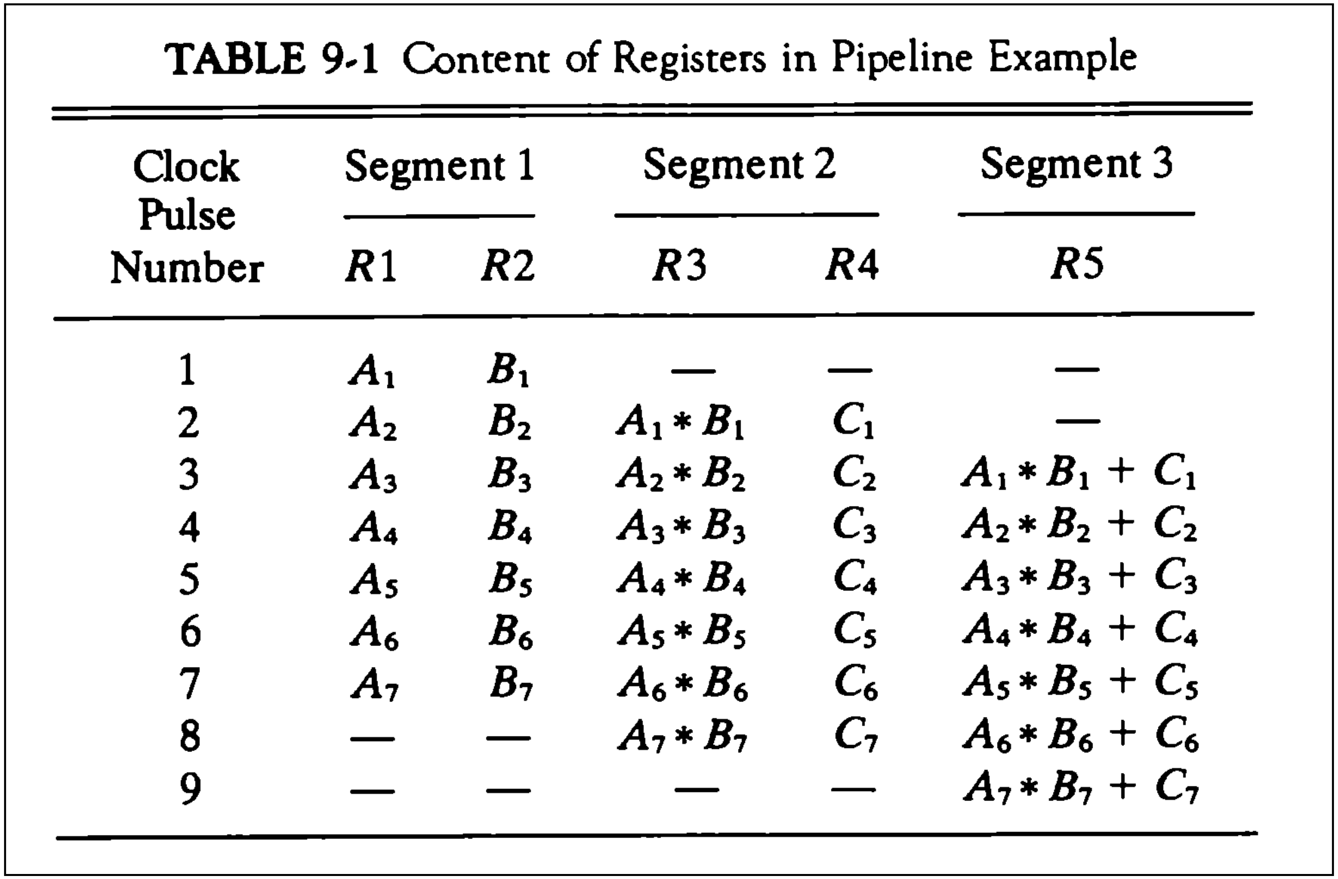


TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	$R1$	$R2$	$R3$	$R4$	$R5$
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

Based on M. Morris Mano “Computer System Architecture”-- Lecturer Ahmed Salah Hameed

SPEED CALCULATION

- **n is number of tasks**
- **Each task divided into k-segments**
- **Each k-segment executed in a clock cycle time (tp)**

- The first task **T1** requires a time equal to **(k*tp)** to complete its operation because we have **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)*tp**.
- Therefore, to complete **n tasks** using a **k-segment** pipeline:

k + (n - 1) clock cycles

For example system with **four** segments and **six** tasks.

The time required to complete all the operations is/

4 + (6 - 1) = 9 clock cycles

Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed

SPEED CALCULATION

Clock cycles

Tasks	1	2	3	4	5	6	7	8	9
1	k1	k2	k3	k4					
2		k1	k2	k3	k4				
3			k1	k2	k3	k4			
4				k1	k2	k3	k4		
5					k1	k2	k3	k4	
6						k1	k2	k3	k4

SPEED CALCULATION

NONPIPELINE UNIT

- Each task take time equal to **tn.**
- The total time required for **n** tasks is

n * tn

PIPELINE UNIT

- To complete **n tasks** using a **k-segment** pipeline:

k + (n - 1) clock cycles

- Total time is: **k + (n - 1) * tp**

Smax

SPEED UP

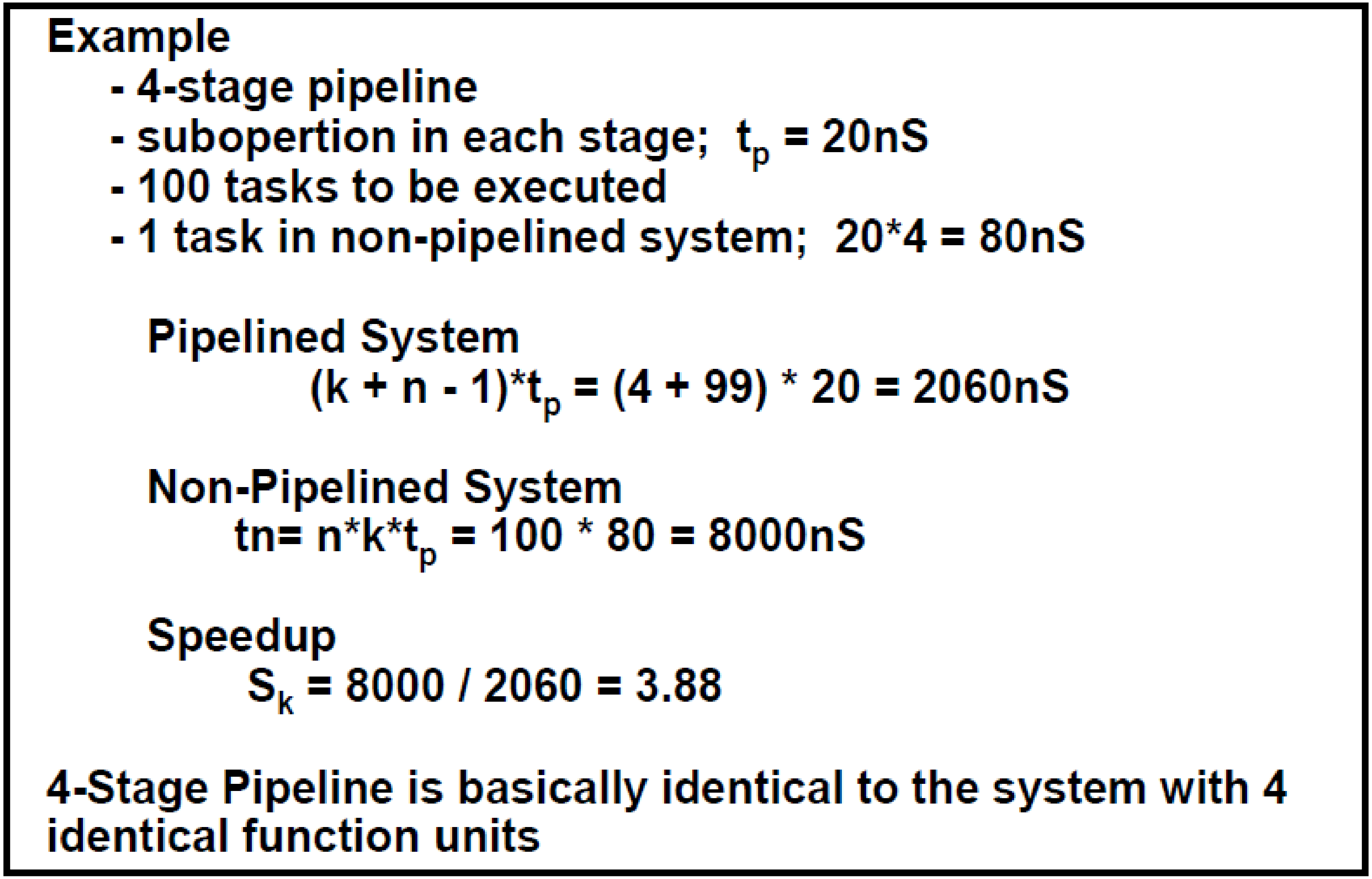
$$S = \frac{nt_n}{(k + n - 1)t_p}$$

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

$$S = \frac{kt_p}{t_p} = k$$

Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed

EXAMPLE ON SPEED CALCULATION



Example

- 4-stage pipeline
- subopertion in each stage; t_p = 20nS
- 100 tasks to be executed
- 1 task in non-pipelined system; 20*4 = 80nS

Pipelined System

$(k + n - 1)*t_p = (4 + 99) * 20 = 2060$nS

Non-Pipelined System

$tn= n*k*t_p = 100 * 80 = 8000$nS

Speedup

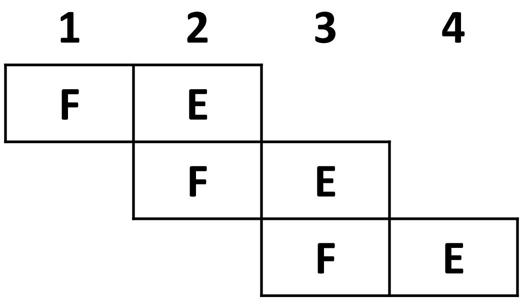
$S_k = 8000 / 2060 = 3.88$

4-Stage Pipeline is basically identical to the system with 4 identical function units

Based on M. Morris Mano “Computer System Architecture”-- Lecturer Ahmed Salah Hameed

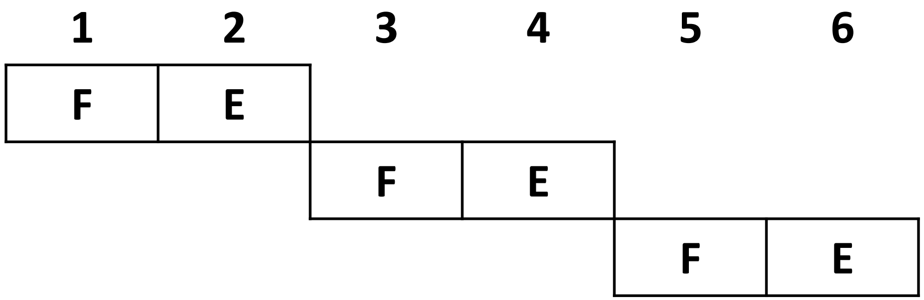
INSTRUCTION PIPELINE

- **An instruction pipeline** reads consecutive instructions from memory while previous instructions are being executed in other segments. This causes the instruction fetch and execute phases to overlap and perform simultaneous operations.
- **Simple Example:**
- Consider a computer with an instruction **fetch unit** and an instruction **execution unit** designed to provide a two-segment pipeline.



1	2	3	4
F	E		
	F	E	
		F	E

2-segment pipelined

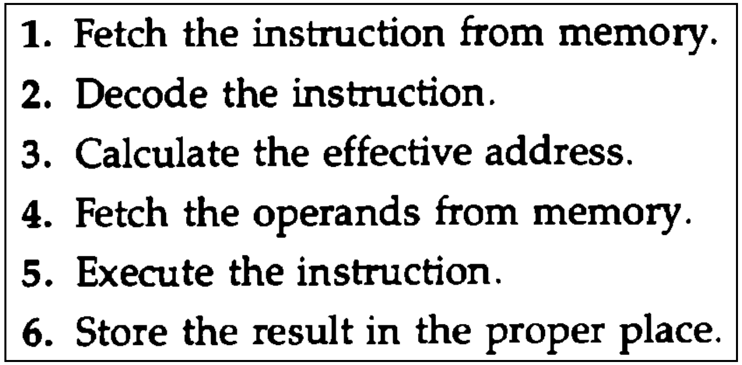


1	2	3	4	5	6
F	E				
		F	E		
				F	E

Non pipelined

INSTRUCTION PIPELINE

- **NOTE 1**: Computers with complex instructions require other phases in addition to the fetch and execute to process an instruction completely.



1. Fetch the instruction from memory.
2. Decode the instruction.
3. Calculate the effective address.
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.

- **NOTE 2:** Difficulties that will prevent the instruction pipeline from operating at its maximum rate.

1. Different segments may take different times to operate on the incoming information.
2. Some segments are skipped for certain operations. For example, a register mode instruction does not need an effective address calculation.
3. Two or more segments may require memory access at the same time, causing one segment to wait until another is finished with the memory.

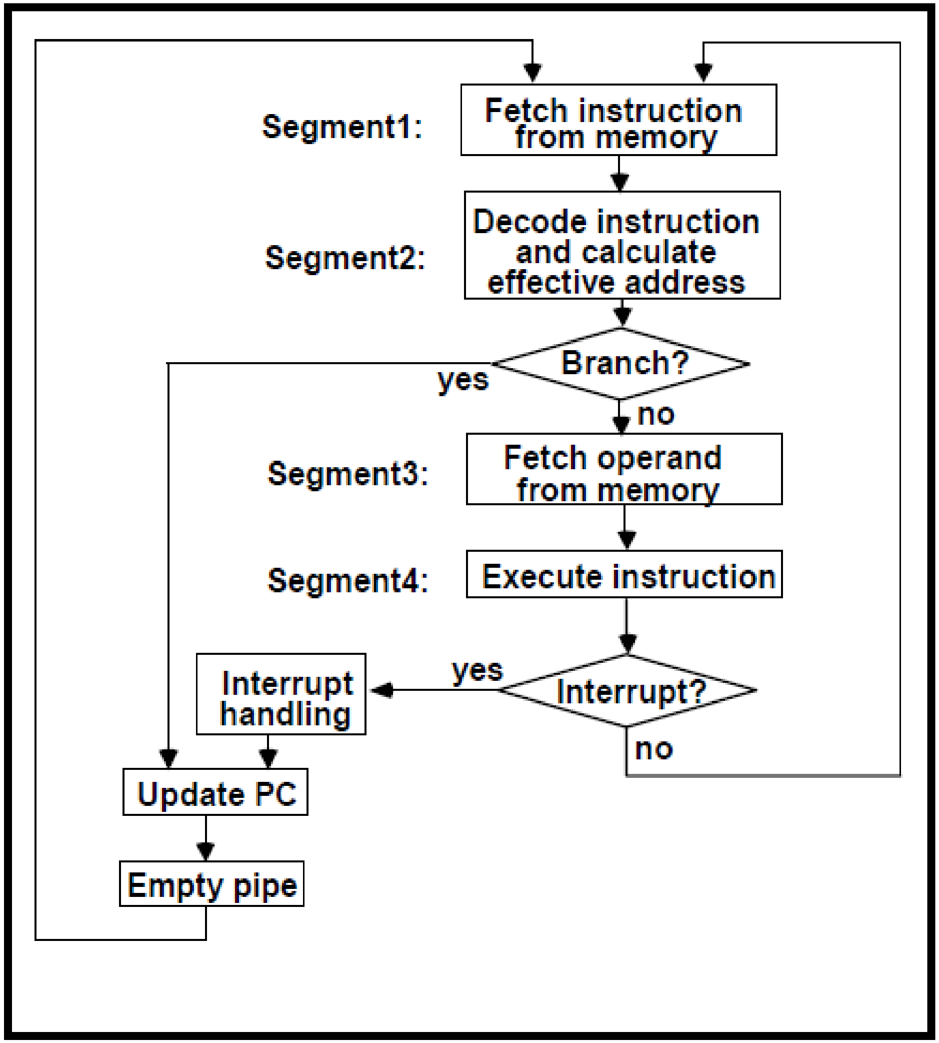
Based on M. Morris Mano “Computer System Architecture”-- Lecturer Ahmed Salah Hameed

INSTRUCTION PIPELINE

- **NOTE 3:** Memory access conflicts are sometimes resolved by using two memory buses for accessing instructions and data in separate modules. In this way, an instruction word and a data word can be read simultaneously from two different modules.

- **NOTE 4:** The design of an instruction pipeline will be most efficient if the instruction cycle is divided into segments of equal duration. The time that each step takes to fulfill its function depends on the instruction and the way it is executed.

EXAMPLE: FOUR-SEGMENT INSTRUCTION PIPELINE



1. FI is the segment that fetches an instruction.

2. DA is the segment that decodes the instruction and calculates the effective address.

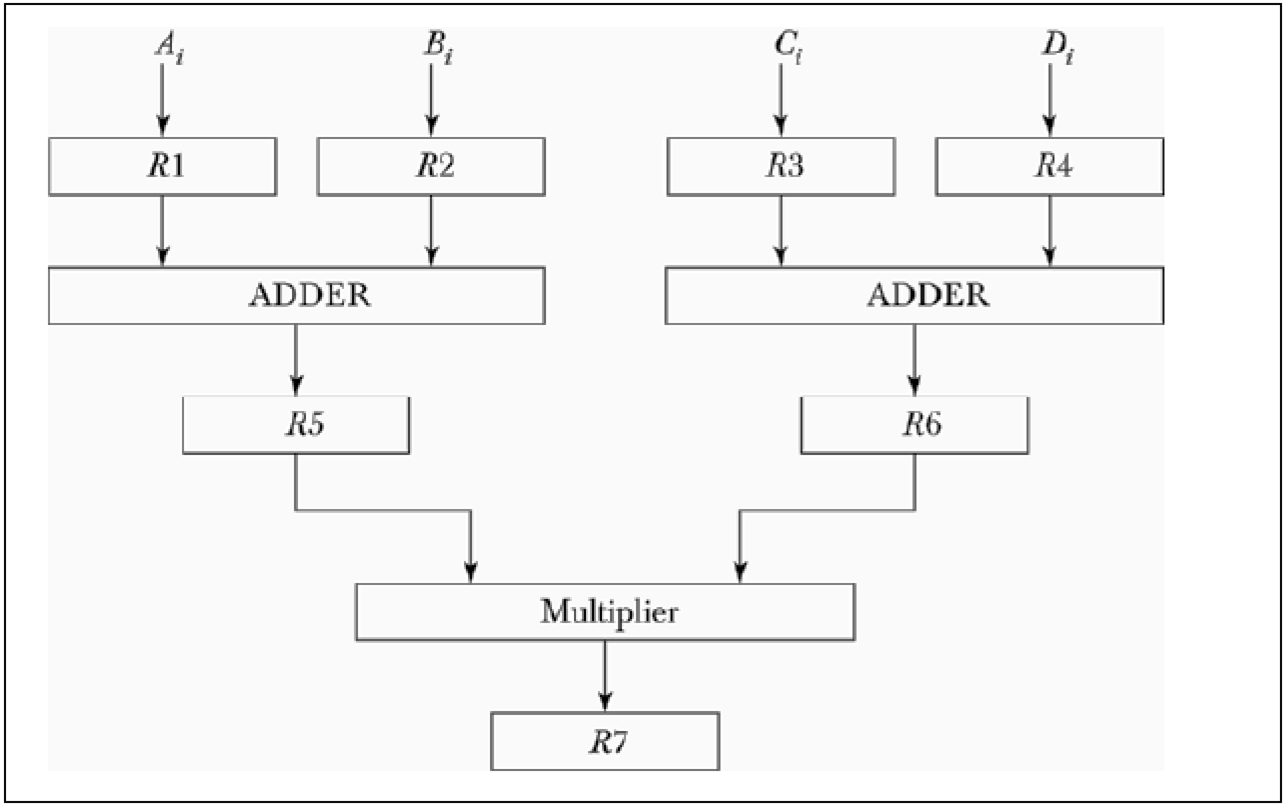
3. FO is the segment that fetches the operand.

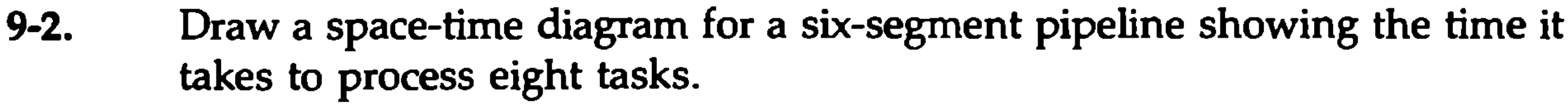
4. EX is the segment that executes the instruction.

Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed

PROBLEMS

9-1. In certain scientific computations it is necessary to perform the arithmetic operation $(A_i + B_i)(C_i + D_i)$ with a stream of numbers. Specify a pipeline configuration to carry out this task. List the contents of all registers in the pipeline for $i = 1$ through 6.

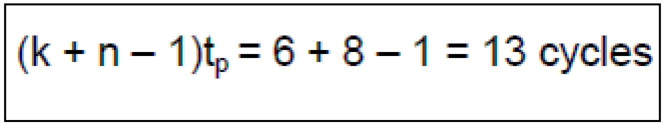




9-2. Draw a space-time diagram for a six-segment pipeline showing the time it takes to process eight tasks.

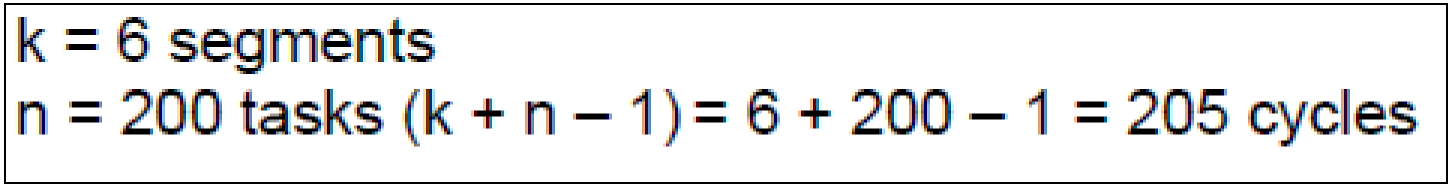
Clock cycles

Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13
1	k1	k2	k3	k4	k5	k6							
2		k1	k2	k3	k4	k5	k6						
3			k1	k2	k3	k4	k5	k6					
4				k1	k2	k3	k4	k5	k6				
5					k1	k2	k3	k4	k5	k6			
6						k1	k2	k3	K4	k5	k6		
7							k1	k2	k3	K4	k5	k6	
8								k1	k2	k3	K4	k5	k6



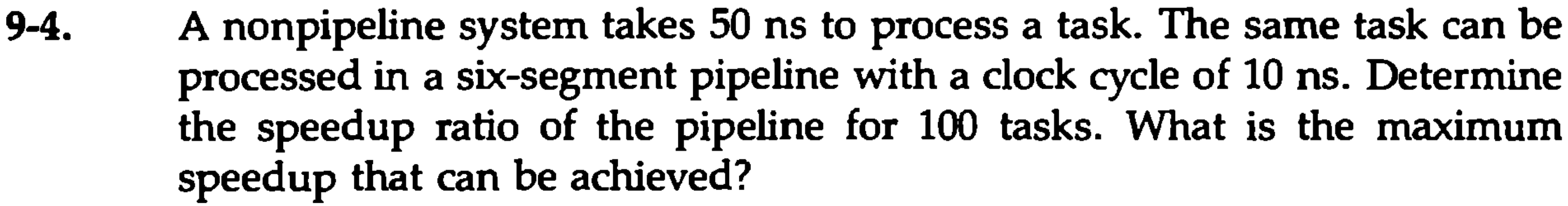
$(k + n - 1)t_p = 6 + 8 - 1 = 13$ cycles

9-3. Determine the number of clock cycles that it takes to process 200 tasks in a six-segment pipeline.

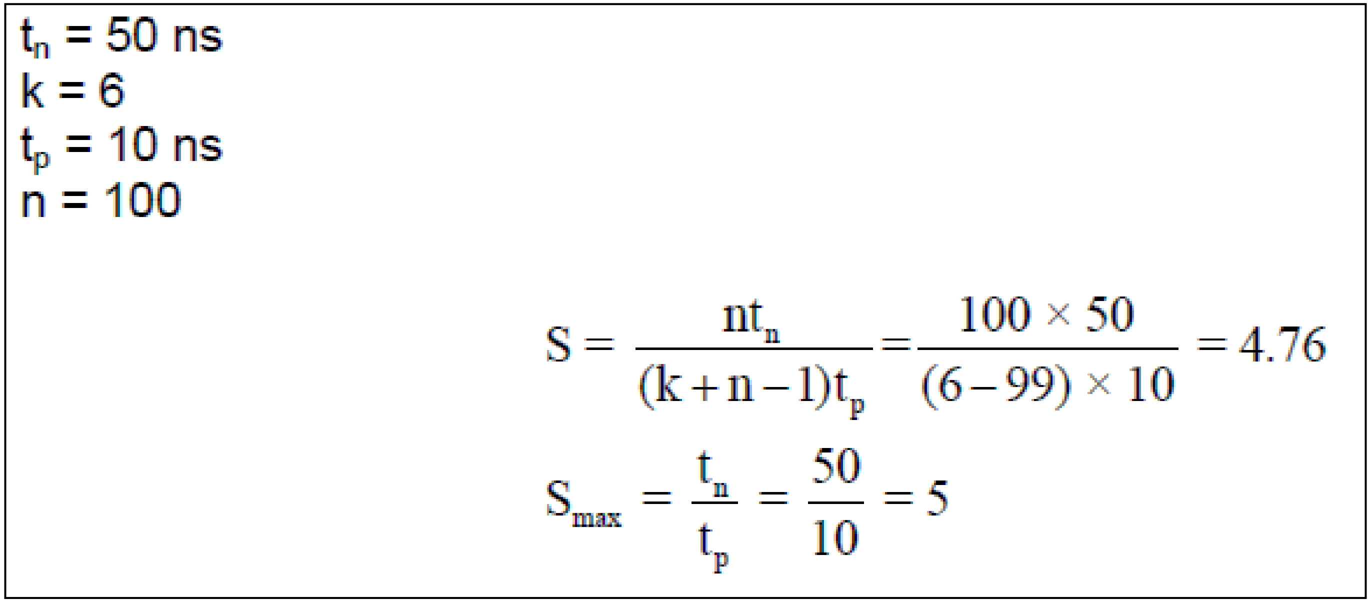


$k = 6$ segments

$n = 200$ tasks $(k + n - 1) = 6 + 200 - 1 = 205$ cycles



9-4. A nonpipeline 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 speedup ratio of the pipeline for 100 tasks. What is the maximum speedup that can be achieved?



t_n = 50 ns

k = 6

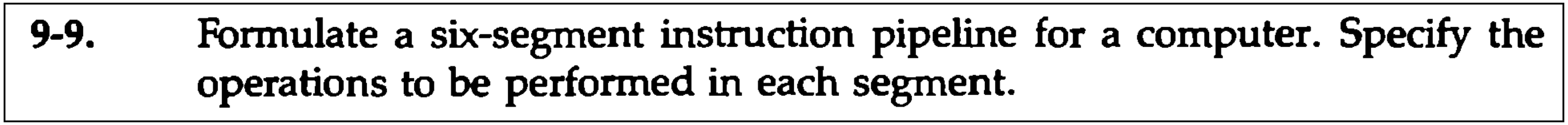
t_p = 10 ns

n = 100

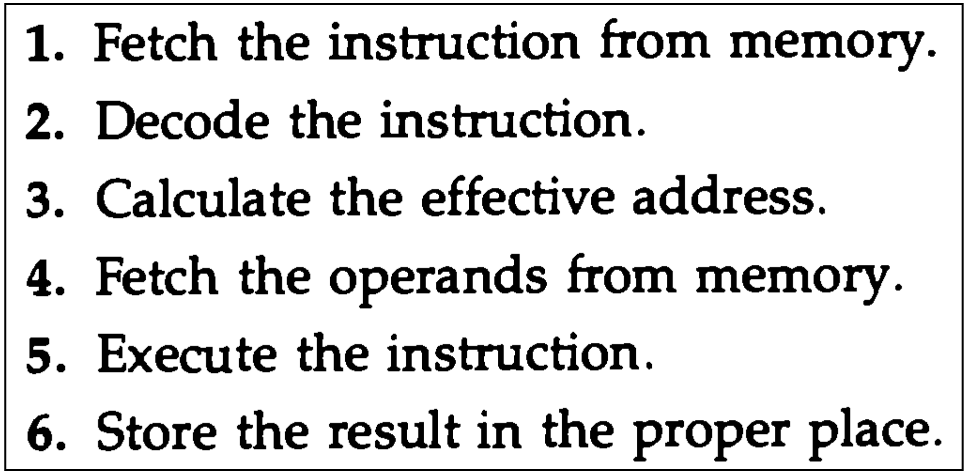
$$S = \frac{nt_n}{(k+n-1)t_p} = \frac{100 \times 50}{(6-99) \times 10} = 4.76$$

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

Based on M. Morris Mano "Computer System Architecture"-- Lecturer Ahmed Salah Hameed



9-9. Formulate a six-segment instruction pipeline for a computer. Specify the operations to be performed in each segment.



1. Fetch the instruction from memory.
2. Decode the instruction.
3. Calculate the effective address.
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.