UNIT 3

COA

PIPELINING

A technique of decomposing a sequential process into suboperations, with each subprocess being executed in a partial dedicated segment that operates concurrently with all other segments.

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

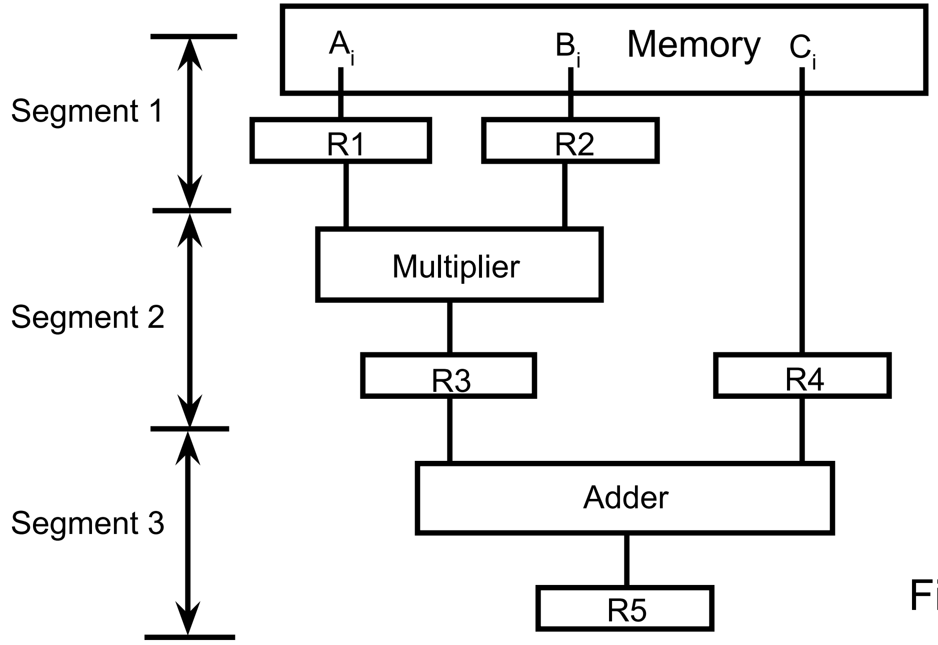


Figure 9.2 Example of pipeline processing

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

OPERATIONS IN EACH PIPELINE STAGE

Table 9.1 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

GENERAL PIPELINE

General Structure of a 4-Segment Pipeline

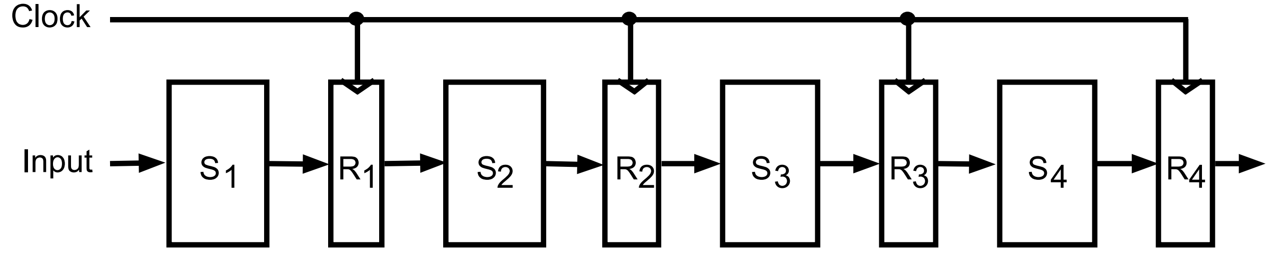
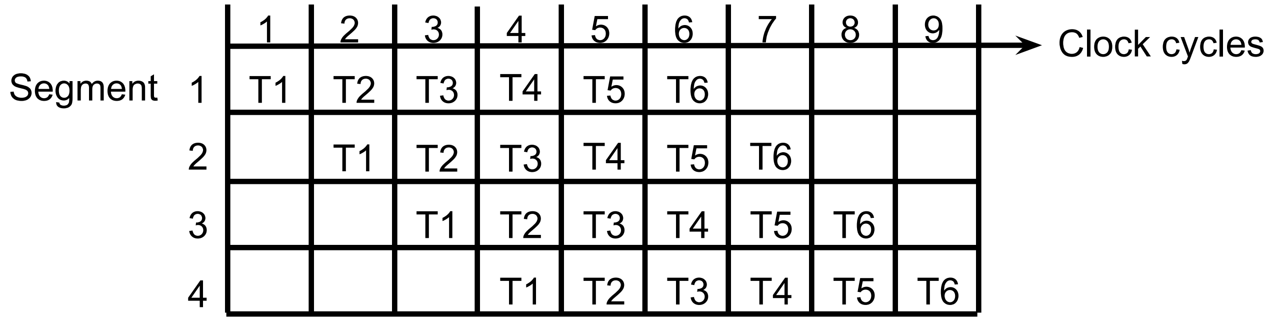


Figure 9.3 Four-segment pipeline

Space-Time Diagram illustrates the behavior of a pipeline



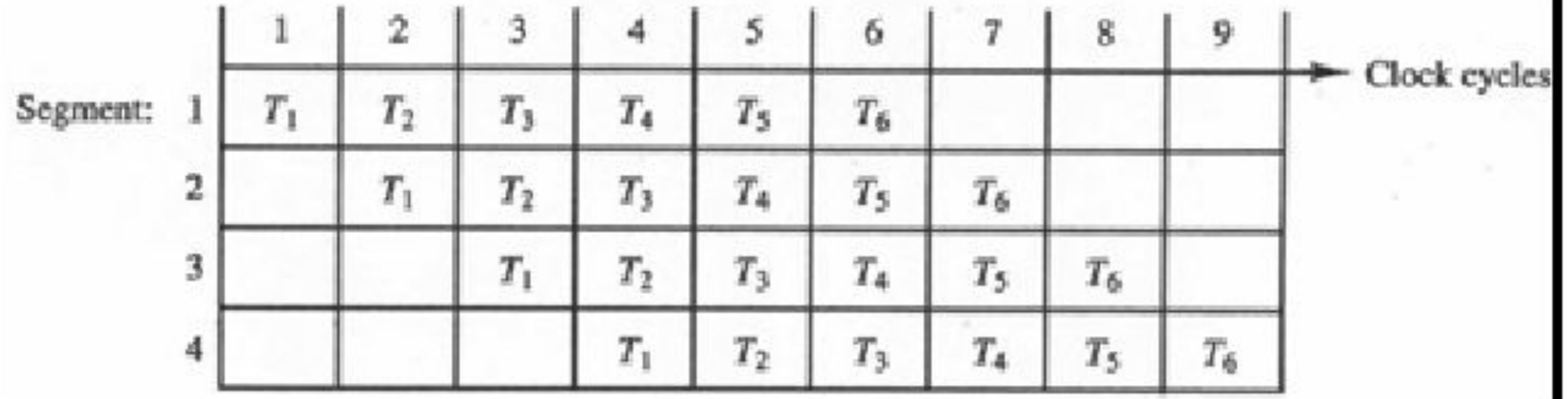
Clock cycles

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

Figure 9.4 Space-time diagram for pipeline.

- Define a task as the total operation performed going through all the segments in the pipeline

Space-Time Diagram



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

4 segments
6 tasks
9 clock cycles to complete via pipeline
24 clock cycles without pipeline

Once pipeline is full, output generated every clock cycle

Pipeline Time Math

• k	segment pipeline
• n	tasks
• t_p	clock cycle time
• kt_p	time to complete task T_1
• $(n\text{-}1)t_p$	time to complete remaining n-1 tasks
• $k+(n\text{-}1)$	clock cycles to complete n tasks using a k segment pipeline
• t_n	time to complete each task
• nt_n	time to complete n tasks w/o pipeline

Pipeline Best Case

- Pipeline always full
- Theoretical speedup limit is a factor of k, where k is the number of pipeline segments

PIPELINE SPEEDUP

n: Number of tasks to be performed

k-segment pipeline (assume)

Conventional Machine (Non-Pipelined)

t_n: time to complete each task

τ_1: Time required to complete the n tasks

$\tau_1 = n * t_n$

Pipelined Machine (k segments)

t_p: Clock cycle time (time to complete each suboperation)

τ_κ: Time required to complete the n tasks

$\tau_\kappa = (k + n - 1) * t_p$

Speedup

S_k: Speedup

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

$$\lim_{n \to \infty} S_k = \frac{t_n}{t_p}$$ ($= k$, if $t_n = k * t_p$ assuming that a task takes the same time in pipeline and nonpipeline circuits)

PIPELINE AND MULTIPLE FUNCTION UNITS

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

Pipelined System

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

Non-Pipelined System

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

Speedup

$S_k = 8000 / 2060 = 3.88$

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

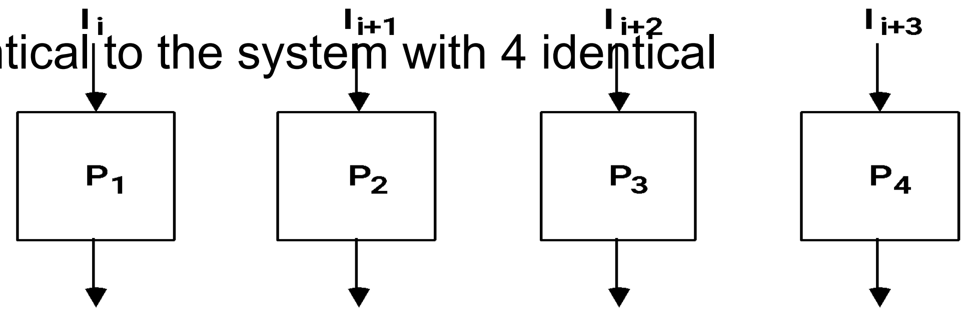


Figure 9.5 Multiple Functional Units in parallel

Pipeline Applications

- Pipeline organization is applicable to
  - Arithmetic pipeline
    - It divides an arithmetic operation into suboperations for execution in the pipeline segments
  - Instruction pipeline
    - It operates on a stream of instructions by overlapping the fetch, decode, and execute phases of the instruction cycle