Computer architecture

Processing Unit

Processing Unit

The **central processing unit (CPU)** of a computer is the main unit that dictates the rest of the computer organization

- *Register set*: Stores intermediate data during the execution of instructions;
- *Arithmetic logic unit (ALU)*: Performs the required micro-operations for executing the instructions;
- *Control unit*: supervises the transfer of information among the registers and instructs the ALU as to which operation to perform by generating control signals.

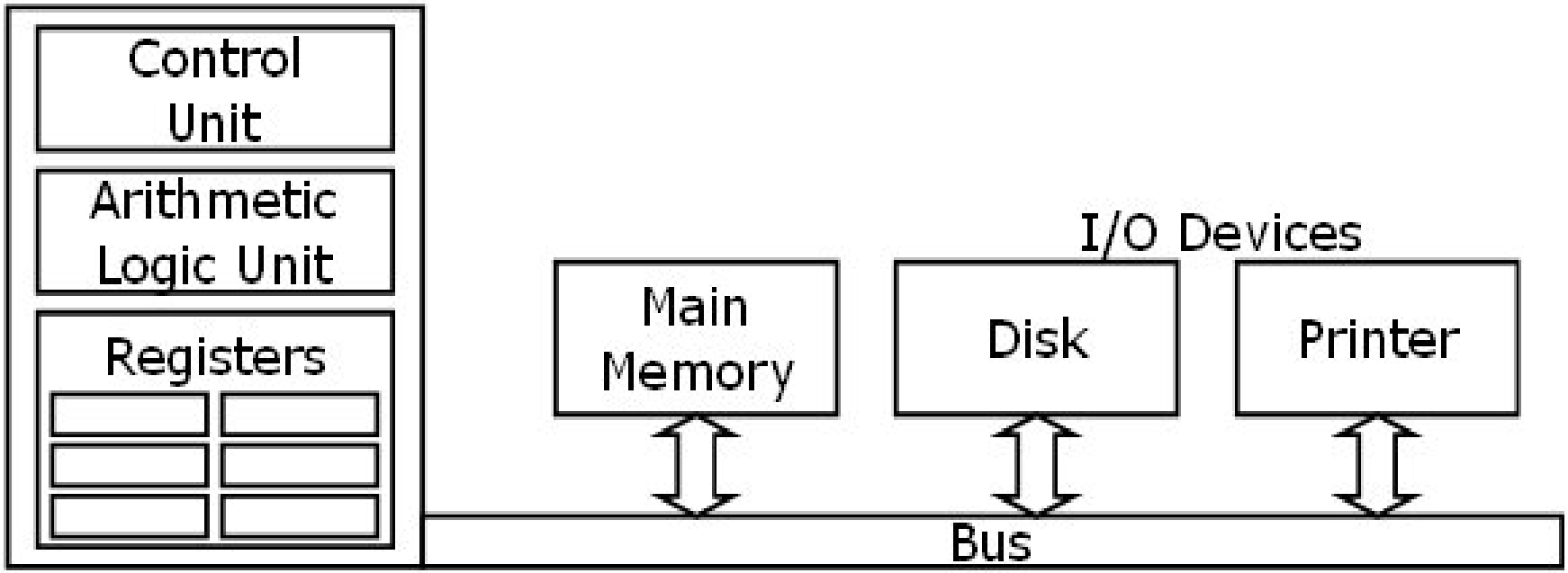
Processing Unit

Functions of CPU

- Central Processing Unit = “brain of computer”
- Executes programs by:
  - Fetching and decoding the next instruction from memory
  - Execute it
- Consists of:
  - Control Unit
  - Arithmetic Logic Unit (ALU)
  - Registers (high-speed memory)
    - Program Counter (PC)
    - Instruction Register (IR)

Processing Unit

Functions of CPU



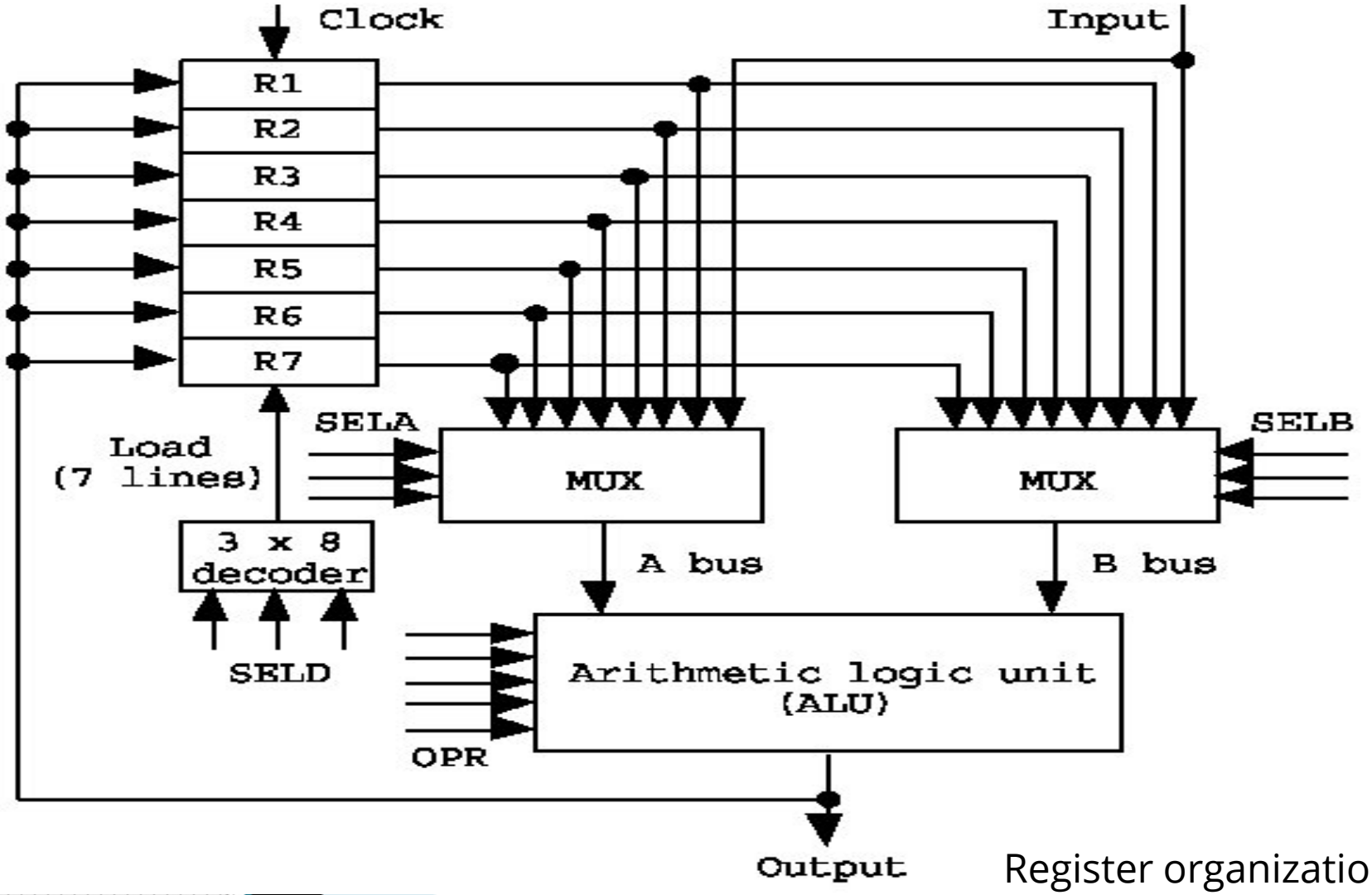
BUS connection with CPU

Processing Unit

General register organization

- CPU must have some working space (fast access and close to CPU)
- This space is efficiently used to store intermediate values
- The most convenient way to communicate registers is trough common bus system

Processing Unit



Register organization

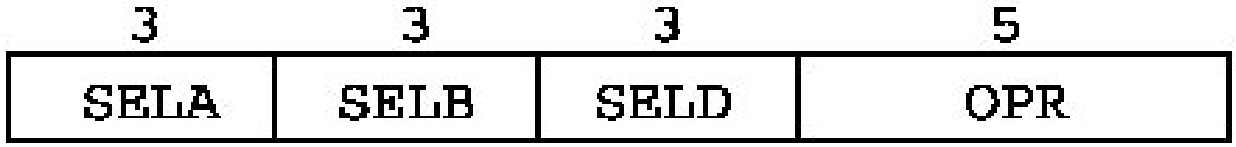
Processing Unit

Selection of operation

- An operation is selected by the ALU operation selector (OPR).
- The result of a micro-operation is directed to a destination register selected by a decoder (SELD).
- Control word: The 14 binary selection inputs (3 bits for SELA, 3 for SELB, 3 for SELD, and 5 for OPR)

Processing Unit

Format of the control word:



Example: R1 ← R2+R3 (01001100110010); 10010 for +

- 1) MUX A selector (**SELA**): to place the content of R2 into BUS A
- 2) MUX B selector (**SELB**): to place the content of R3 into BUS B
- 3) ALU operation selector (**OPR**): to provide the arithmetic addition R2 + R3
- 4) Decoder selector (**SELD**): to transfer the content of the output bus into R1

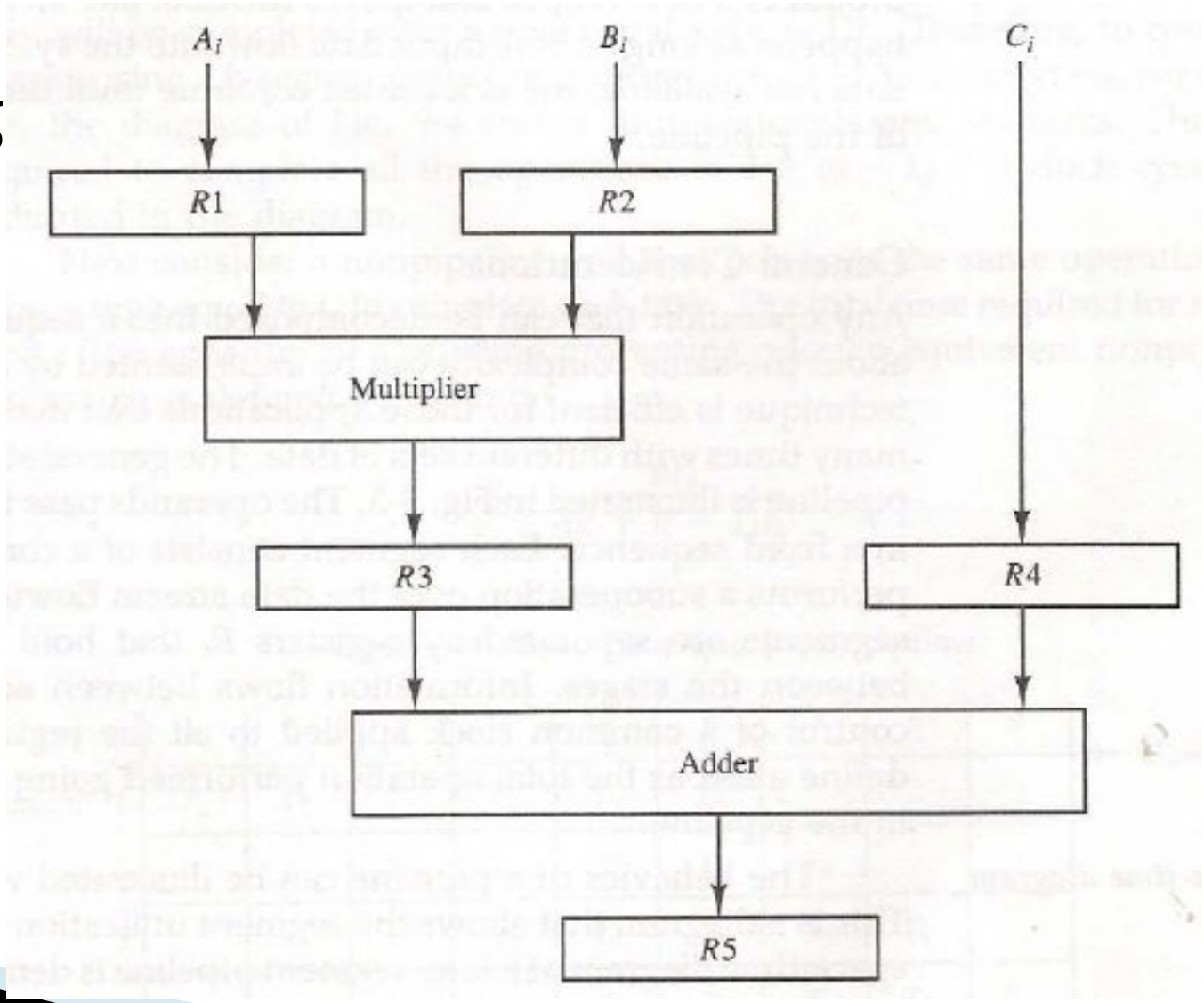
Processing Unit

Pipelining

- Decomposing a sequential process into suboperations
- Each subprocess is executed in a special dedicated segment concurrently
-
- Example:
- Multiply and Add Operation:
  $A_i * B_i + C_i$ (for i = 1, 2, ..., 7)
- 3 Suboperation Segment
  - 1): R1 ← Ai, R2 ← Bi; Input Ai and Bi
  - 2): R3 ← R1*R2, R4 ← Ci; Multiply and input Ci
  - 3): R5 ← R3+R4; Add Ci

Processing Unit

Pipelining

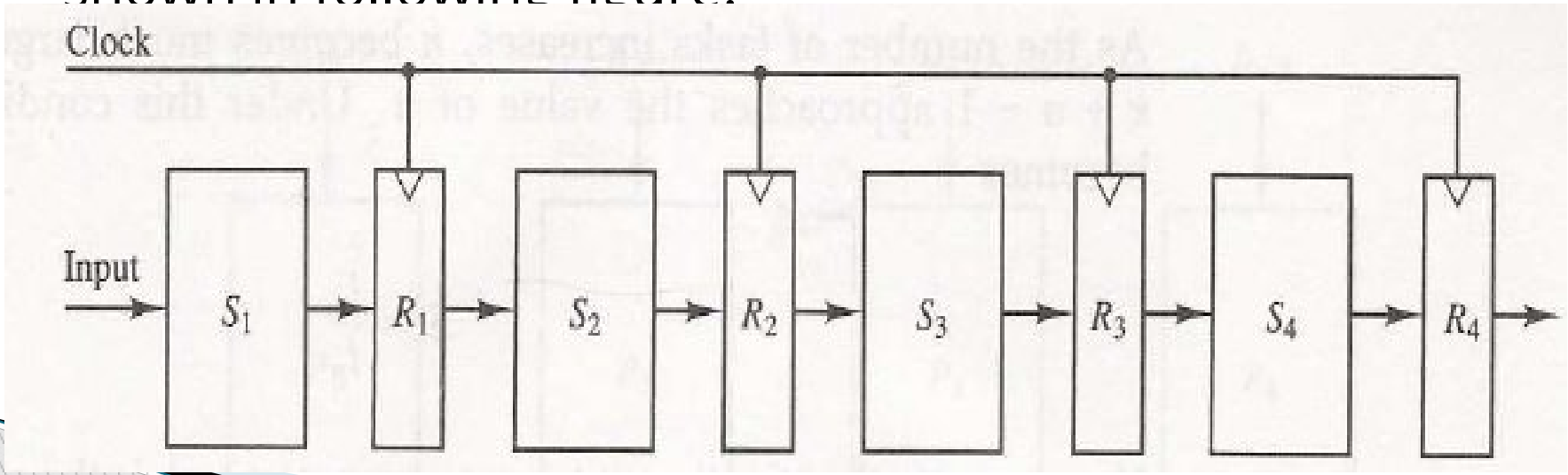


Processing Unit

General consideration of pipelining

- Operation is decomposed into a number of suboperations
- The technique is efficient when the same task is repeated for different data sets

↕ The general structure of four-segment pipelining is shown in following figure:



Processing Unit

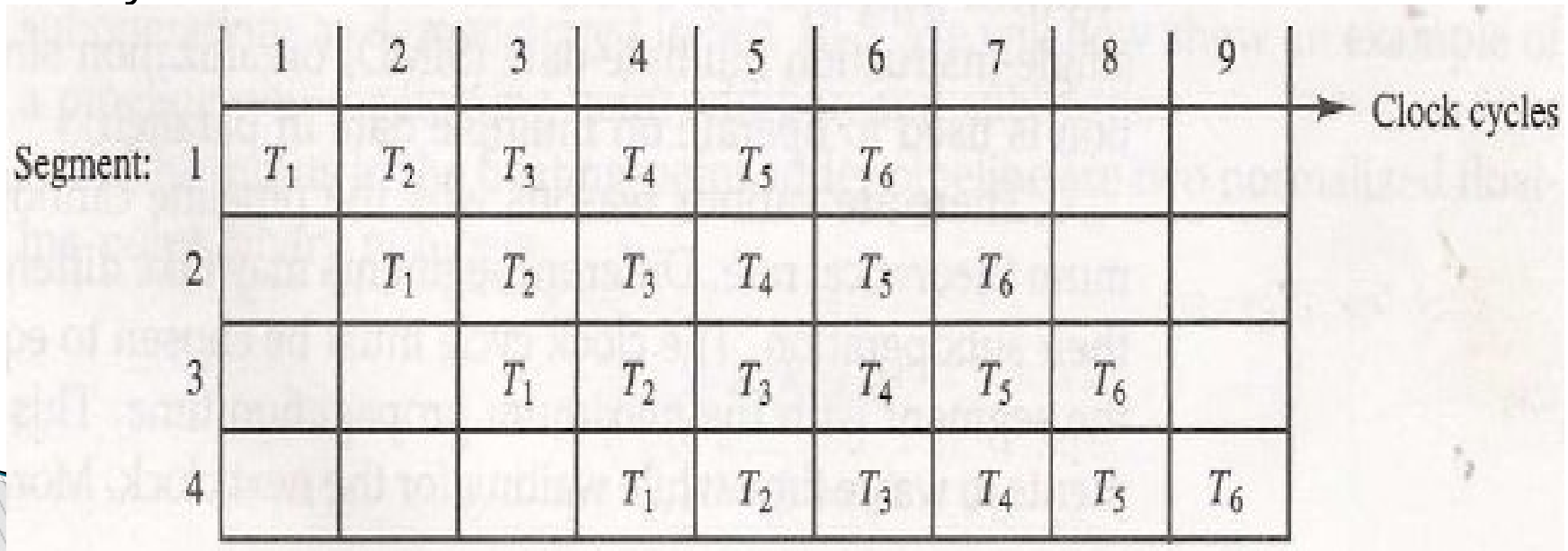
General consideration of pipelining

- Each operation pass through all of four segments
- Section Si performs the suboperation and Ri holds the temporary results
- Common clock is used to transfer the data from Ri

Processing Unit

Space-time representation

- Pipeline can be represented as space-time diagram
- Shows the segment utilization as a function of time
  - The space-time representation of a four-segment pipelining system:



Processing Unit

- The diagram shows six tasks T1 through T6 executed in four segments
- The speedup of pipeline system over nonpipeline is defined as –

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

- where
  - n : *task number* (6)
  - t_p : *clock cycle time*
  - t_n : *time to complete each task in nonpipeline* ($4t_p$)
  - k : *segment number* (4)
  - hence, $S = n \cdot t_n / (k + n - 1) \cdot t_p = 6 \cdot 4t_p / (4+6-1) \cdot t_p = 24\ t_p / 9\ t_p = 2.67$
  - If n ✉ $S = t_n / t_p$
  - So, Nonpipline (t_n)=Pipeline ($k.t_p$) 🖧 $S = k.t_p / t_p = k$

Processing Unit

Instruction Pipeline

- The computer needs to process each instruction with the following sequence of steps:
  - 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

Processing Unit

- *Example*: Four-segment Instruction Pipeline
- The above six steps can be performed with the following Four-segment CPU pipeline:
- 1) **FI**: Instruction Fetch
- 2) **DA**: Decode Instruction & calculate EA
- 3) **FO**: Operand Fetch
- 4) **EX**: Execution

Step:		1	2	3	4	5	6	7	8
Instruction:	1	FI	DA	FO	EX				
	2		FI	DA	FO	EX			
	3			FI	DA	FO	EX		
	4				FI	DA	FO	EX	
	5					FI	DA	FO	EX