Unit-2 (Part 2)

- Instruction Cycle
- Instruction Pipelining

Instruction Cycles:

- In Basic Computer, a machine instruction is executed in the following cycle:

1. Fetch an instruction from memory
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address
4. Execute the instruction

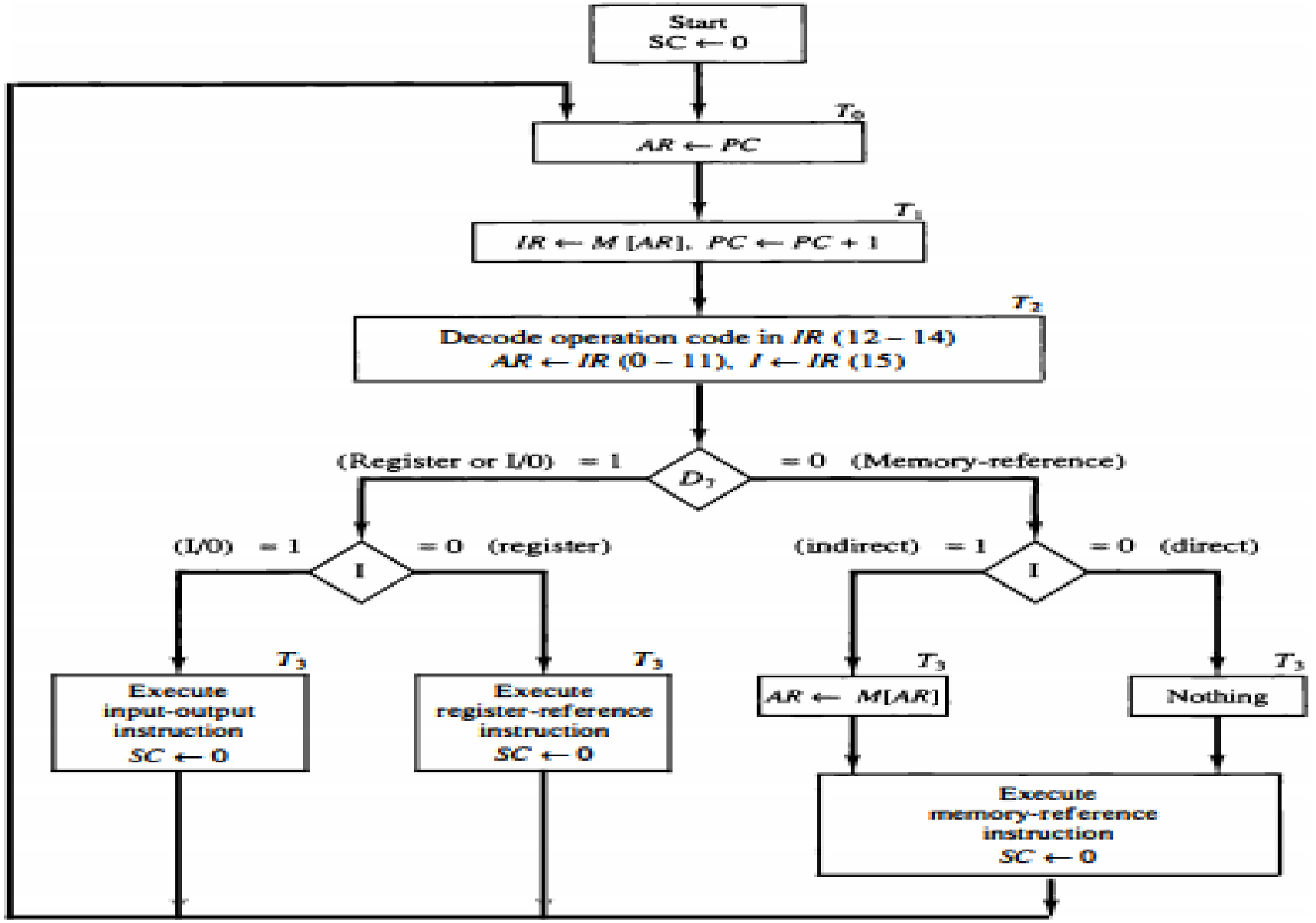
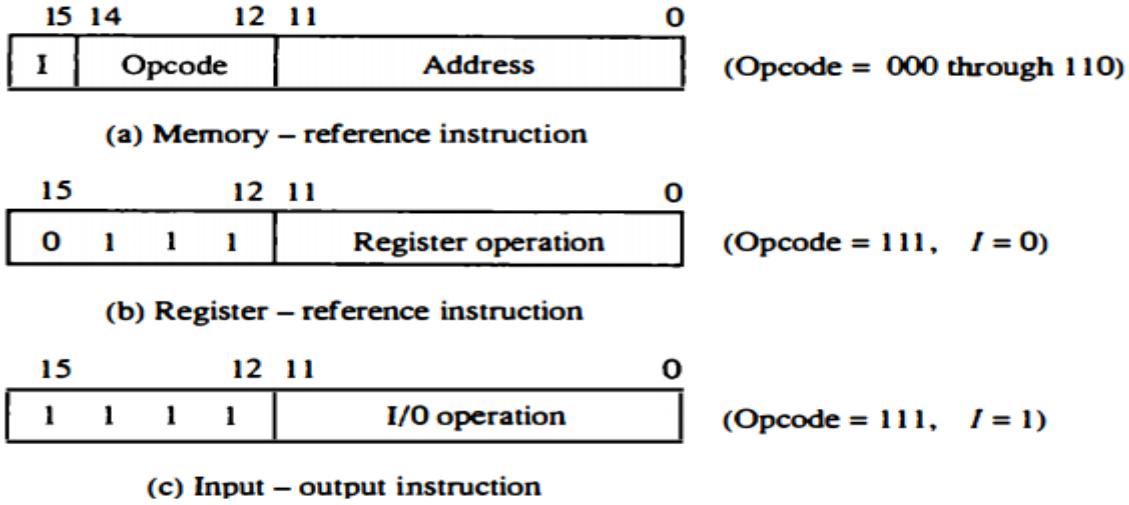


Figure Flowchart for instruction cycle (initial configuration).

- **The timing signal** that is active after the decoding is T_3.
- During time T_3, the control unit determines the type of instruction that was just read from memory.
- **The flowchart of Fig. above** presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding.
- **The three possible** instruction types available in the basic computer are specified in Fig. on basic computer formats.

Figure Basic computer instruction formats.



- **Decoder output D_7 is equal to 1** if the operation code is equal to binary 111. From Fig. on basic computer formats we determine that if $D_7 = 1$, the instruction must be a register-reference or input-output type.
- **If $D_7 = 0$,** the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.
- **Control then inspects the value** of the first bit of the instruction, which is now available in flip-flop I. If D7 = 0 and I = 1, we have a memory reference instruction with an indirect address.
- **It is then necessary** to read the effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement:

$$AR \leftarrow M[AR].$$

- **Initially, AR holds the address part of the instruction**. This address is used during the memory read operation. The word at the address given by AR is read from memory and placed on the common bus. The LD input of AR is then enabled to receive the indirect address that resides in the 12 least significant bits of the memory word.

- **The three instruction types** are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing signal T_3. This can be symbolized as follows:

  ->D'7 IT3: AR ← M[AR]

  ->D'7 I'T3: Nothing

  ->D7 I'T3: Execute a register-reference instruction

  ->D7IT3: Execute an input-output instruction

- **When a memory-reference instruction** with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR.
- **However, the sequence counter SC** must be incremented when $D'_7T_3 = 1$, so that the execution of the memory-reference instruction can be continued with timing variable T_4
- **A register-reference or input-output instruction** can be executed with the clock associated with timing signal T_3. After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with $T_0 = 1$.

Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement SC ← SC + 1, but it will be implied that the control goes to the next timing signal in sequence. When SC is to be cleared, we will include the statement SC ← 0.

Instruction Pipelining

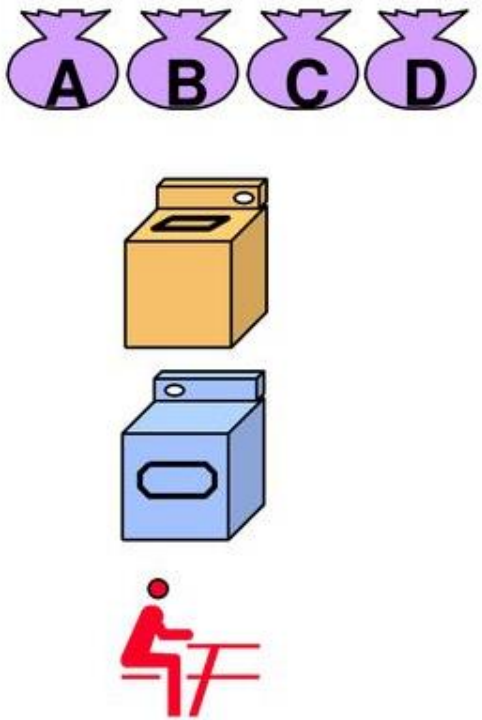
- A program consists of several numbers of instructions, these instructions may be executed in two ways:
    1. Non-pipelined execution
    2. Pipelined execution

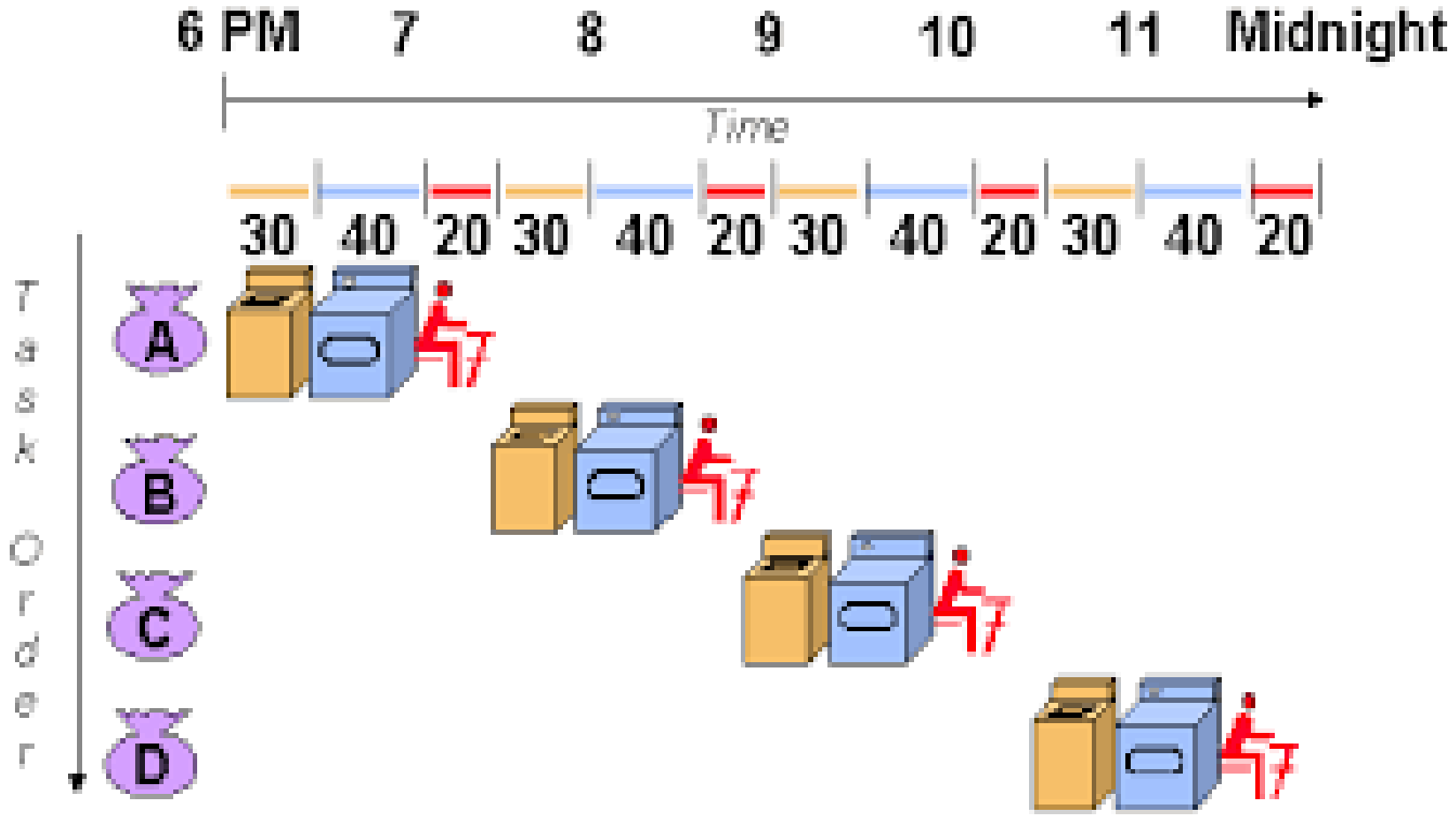
1. Non-pipelined execution:

- All the instructions of a program are executed sequentially one after the other
- A new instruction executed only after the previous instruction has executed completely
- This type of executing the instruction is highly inefficient

- Example:

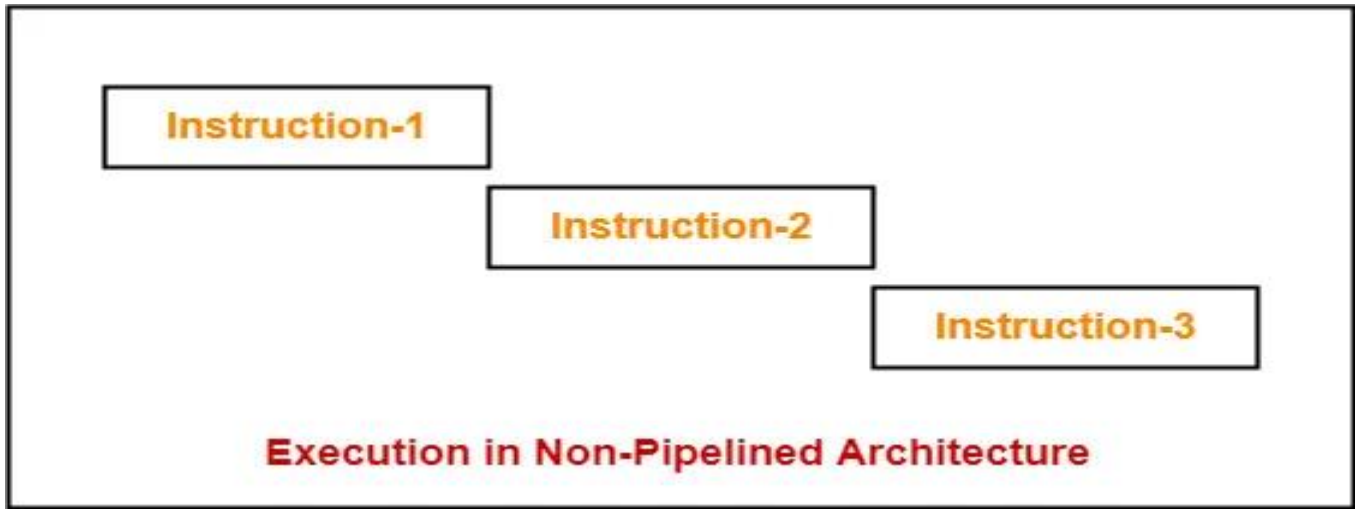
  - Laundry Example
  - Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
  - Washer takes 30 minutes
  - Dryer takes 40 minutes
  - “Folder” takes 20 minutes




6 PM
7
8
9
10
11
Midnight
Time
30
40
20
30
40
20
30
40
20
30
40
20
Task Order
A
B
C
D

Example: Consider a program consisting of three instructions.

In a non-pipelined architecture, these instructions execute one after the other as-



Execution in Non-Pipelined Architecture

If time taken for executing one instruction = t, then-

Time taken for executing ‘n’ instructions = n x t

2. Pipelined execution:

- ✓Multiple instructions are executed parallelly.
- ✓ This type of executing the instructions is highly efficient
- ✓ Instruction Pipelined is a technique that implements a form of parallelism called as instruction level parallelism within a single processor.
- ✓Pipelining is widely used in modern processor as it improves system performances in terms of throughput (Number of work done at a given time)

Pipelined Architecture

In pipelined architecture:

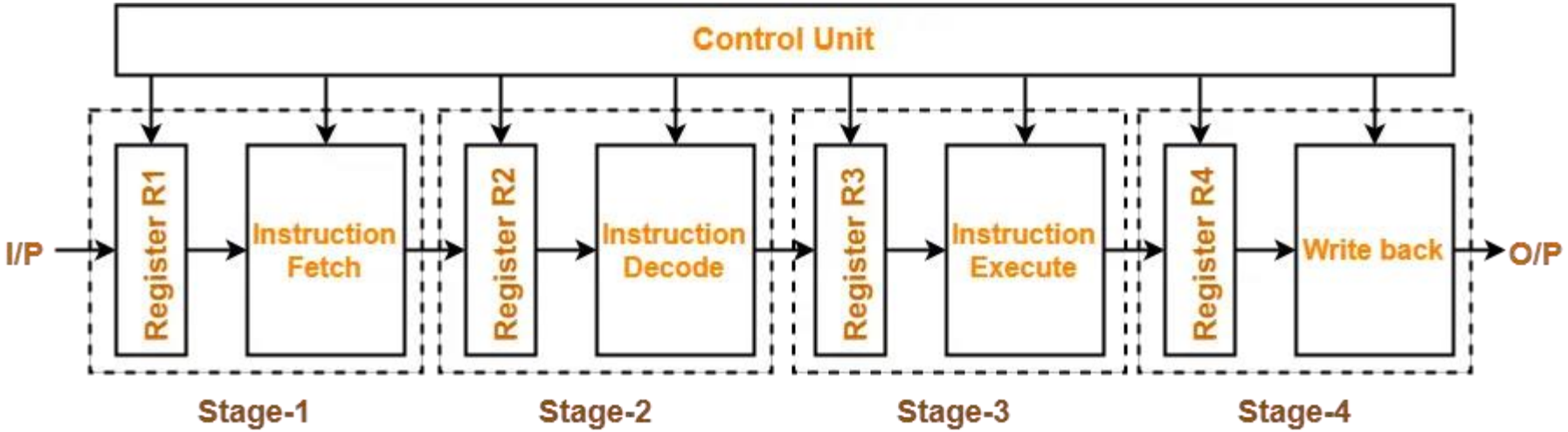
- The hardware of the CPU is split up into several functional units.
- Each functional unit performs a dedicated task.
- The number of functional units may vary from processor to processor.
- These functional units are called as stages of the pipeline.
- Control unit manages all the stages using control signals.
- There is a register associated with each stage that holds the data.
- There is a global clock that synchronizes the working of all the stages.
- At the beginning of each clock cycle, each stage takes the input from its register.
- Each stage then processes the data and feed its output to the register of the next stage.

4-stage Pipeline:

- In 4 stage pipeline architecture, the execution of each instruction is completed in following 4 stage :
  1. Instruction Fetch (IF)
  2. Instruction Decode (ID)
  3. Instruction Execute (IE)
  4. Write Back (WB)
- To implement this 4 stage:

✓The hardware of the CPU is divided into 4 functional units

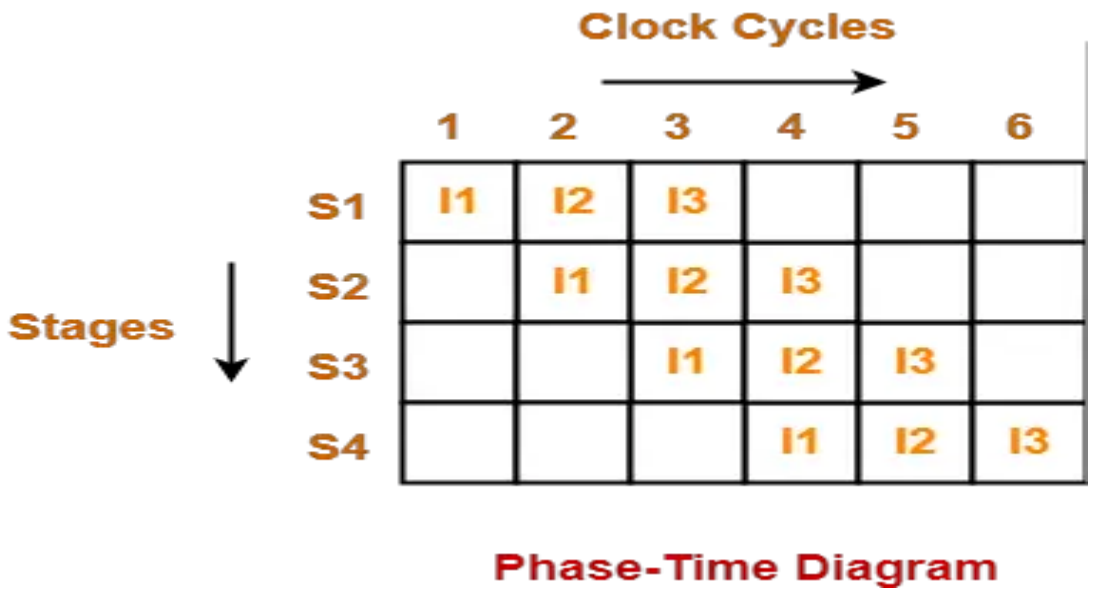
✓Each Functional Unit performs a dedicated task



Stage-1	Stage-2	Stage-3	Stage-4
• First functional unit performs instruction fetch. • It fetches the instruction to be executed.	• Second functional unit performs instruction decode. • It decodes the instruction to be executed.	• Third functional unit performs instruction execution. • It executes the instruction.	• Fourth functional unit performs write back. • It writes back the result so obtained after executing the instruction.

Phase-Time Diagram:

- Phase-time diagram shows the execution of instructions in the pipelined architecture.
- The following diagram shows the execution of three instructions in four stage pipeline architecture.



Phase-Time Diagram

NOTE-

In non-pipelined architecture,
Time taken to execute three instructions would be
= 3 x Time taken to execute one instruction
= 3 x 4 clock cycles
= 12 clock cycles
Clearly, pipelined execution of instructions is far more efficient than non-pipelined execution.

Time taken to execute three instructions in four stage pipelined architecture = 6 clock cycles.