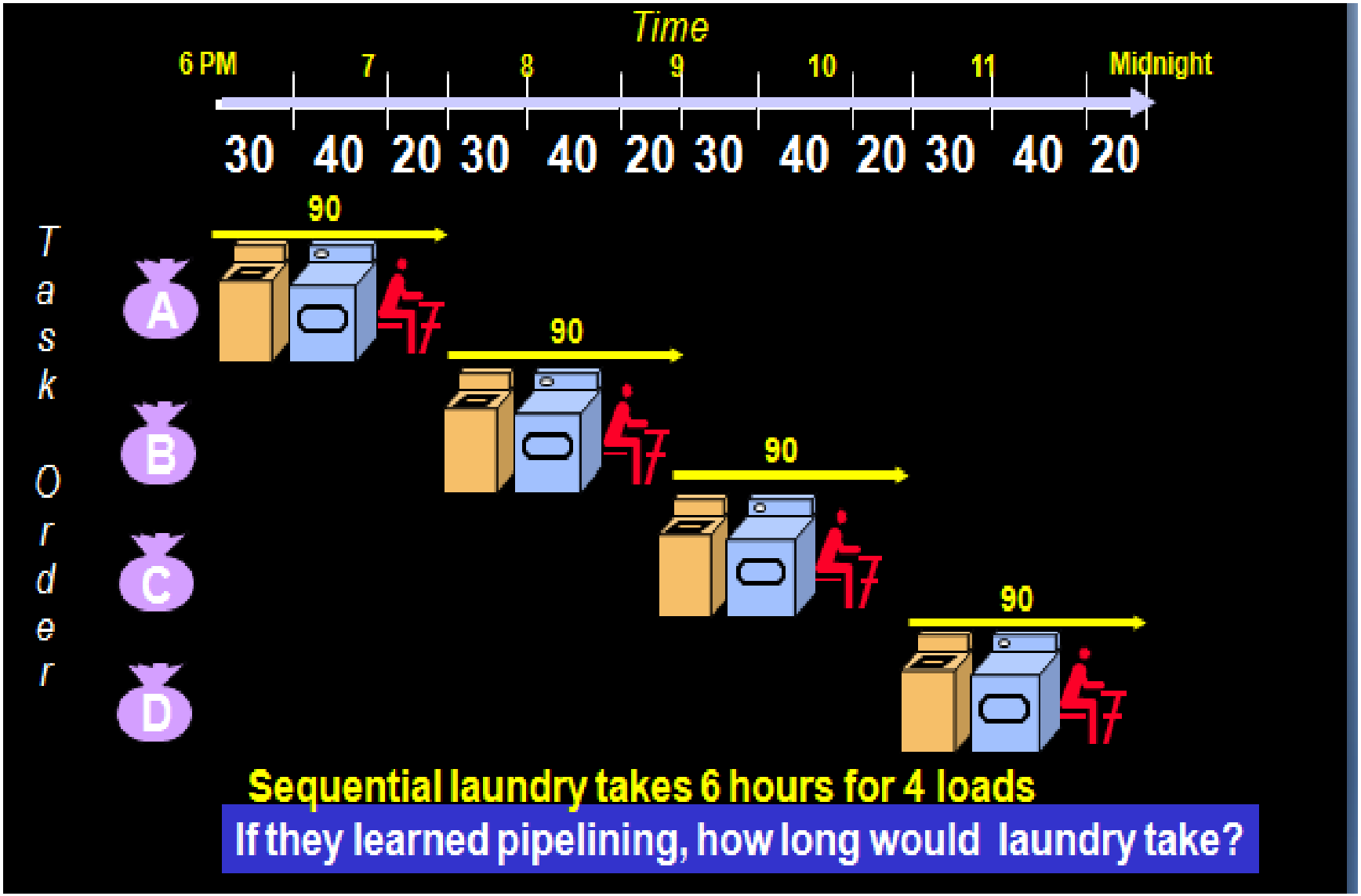
PIPELINING

P. K. ROY

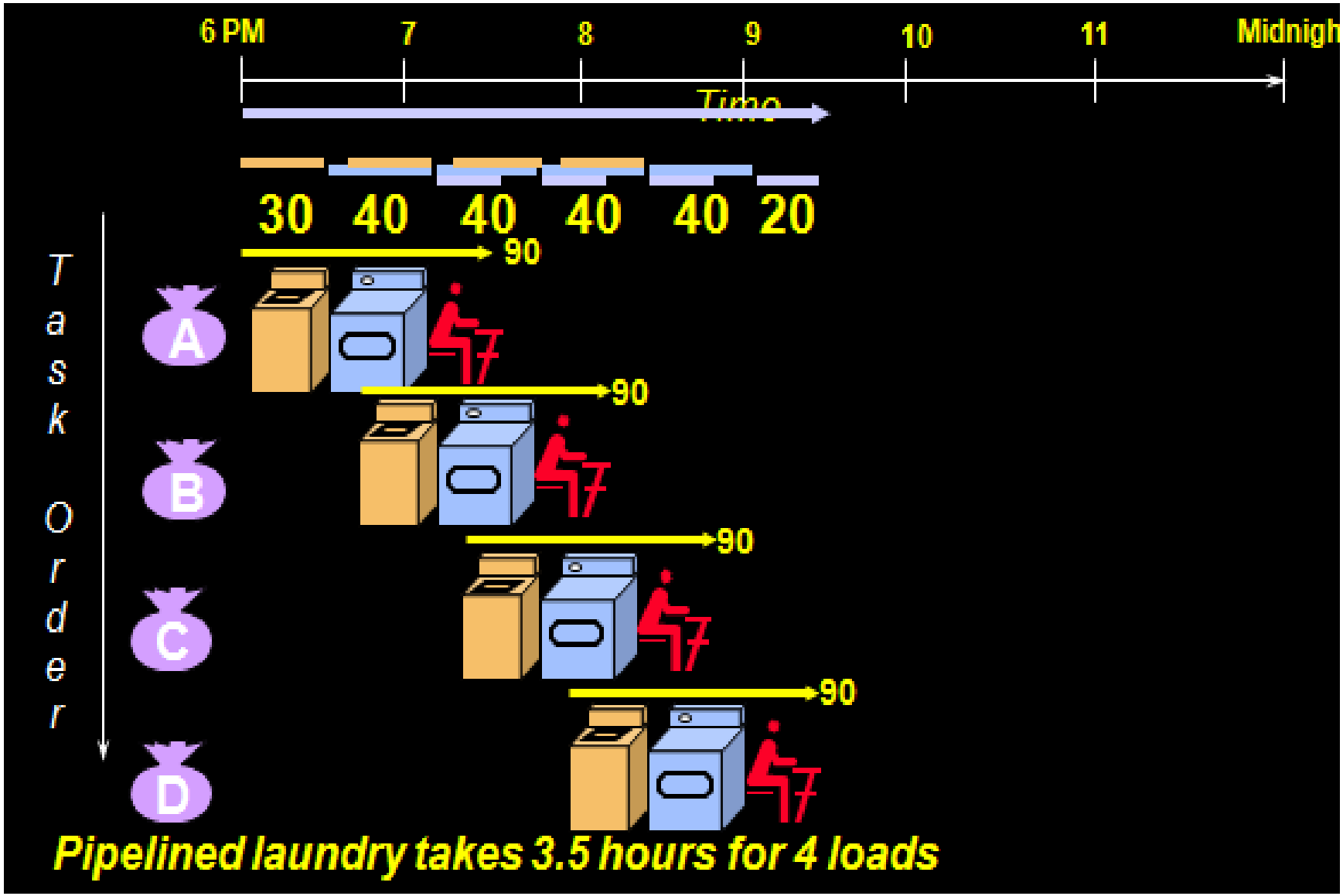
Why Pipelining ?

- A program is a collection of instructions.
- ***Objective*** is to execute these instructions within a limited no. of clock cycles, i.e., speed-up the execution process.
- ***Possible solution*** is to execute multiple instructions concurrently rather than sequentially.
- Pipelining provides the mechanism to execute multiple instructions concurrently.
- Pipelining helps in efficient use of CPU resources.
- By using pipelining, throughput & speed of execution can be increased without any additional H/W requirements.
- Pipelining doesn’t help latency of single task, it helps throughput of entire workload

Basic Concept



Cntd…


6 PM
7
8
9
10
11
Midnigh
Time
30 40 40 40 40 20
90
90
90
90
Task Order
A
B
C
D
Pipelined laundry takes 3.5 hours for 4 loads

How Pipelining Works ?

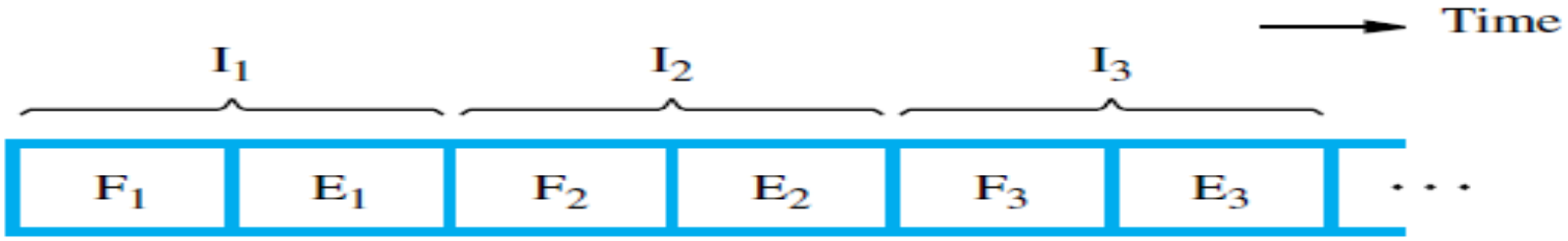
- A technique used in advanced microprocessors where the microprocessor begins executing a second instruction before the first has been completed.

- A Pipeline is a series of stages, where some work is done at each stage. The work is not finished until it has passed through all stages.

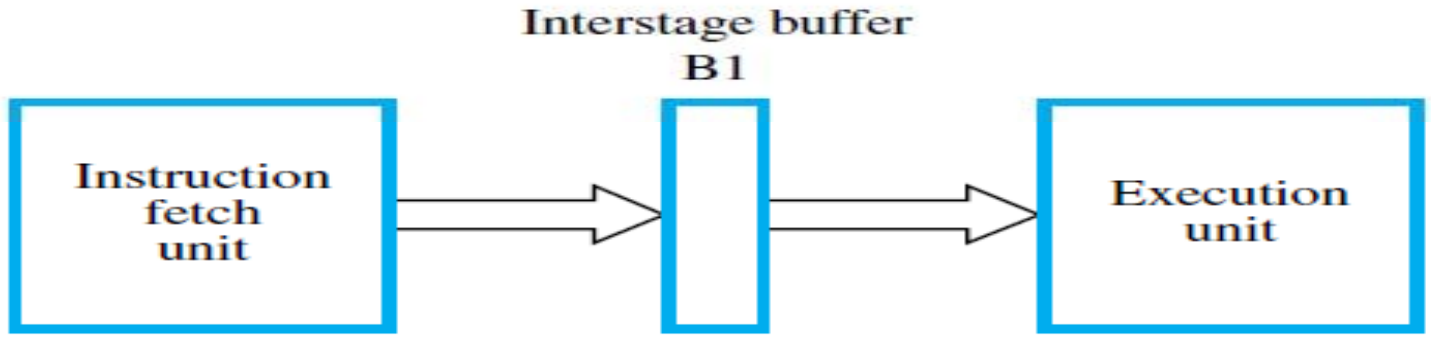
- With pipelining, the computer architecture allows the next instructions to be fetched while the processor is performing arithmetic operations, holding them in a buffer close to the processor until each instruction operation can performed.

- The pipeline is divided into segments and each segment can execute it operation concurrently with the other segments. Once a segment completes an operations, it passes the result to the next segment in the pipeline and fetches the next operations from the preceding segment.

How Pipelining Works ?



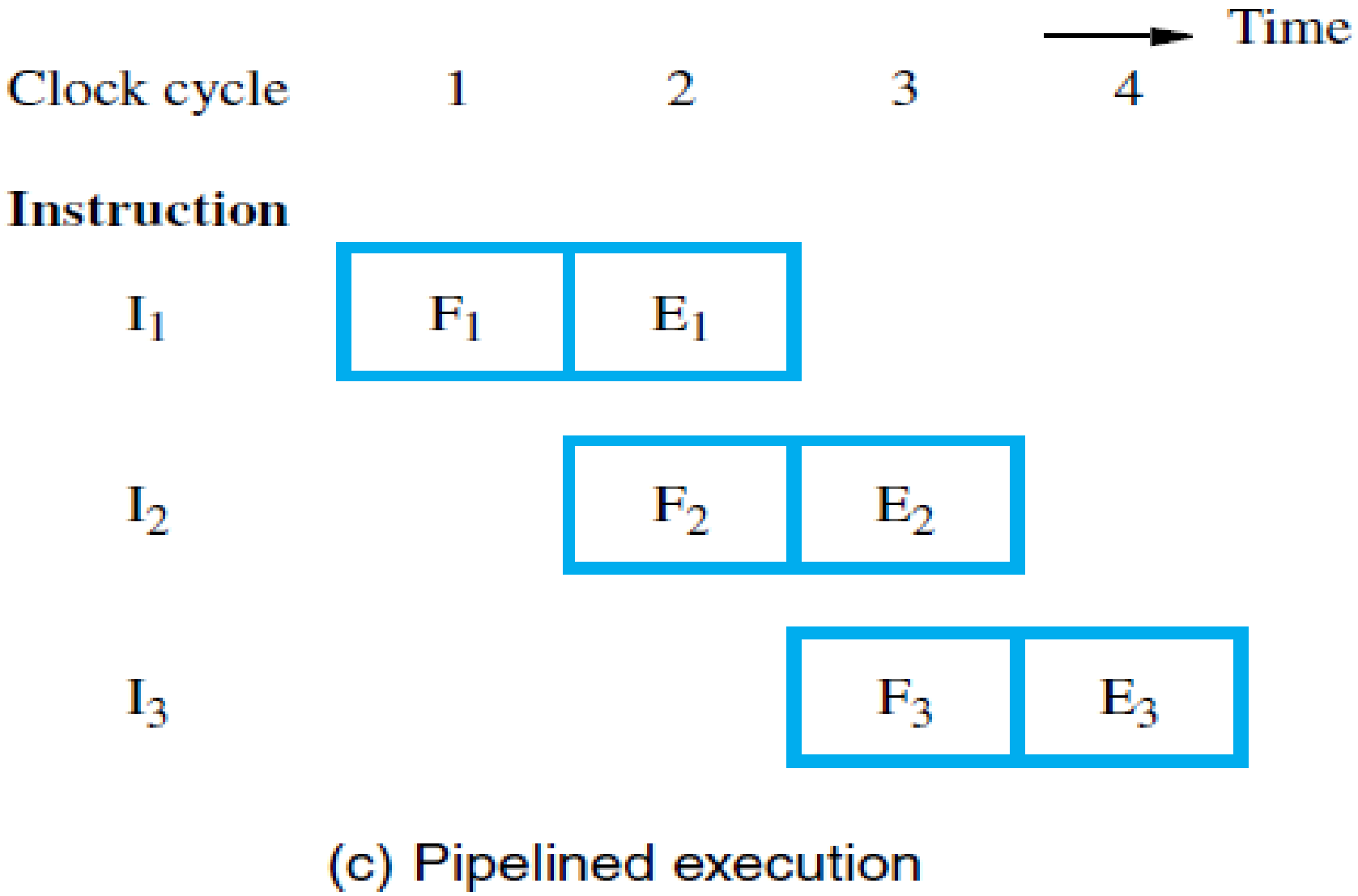
(a) Sequential execution



(b) Hardware organization

An inter-stage storage buffer, B1, is needed to hold the information being passed from one stage to the next. New information is loaded into this buffer at the end of each clock cycle.

How Pipelining Works ?



(c) Pipelined execution

A 4-Stage Pipelining

- A pipelined processor may process each instruction in four steps, as follows:

  Fetch (F) **:** read the instruction from the memory.

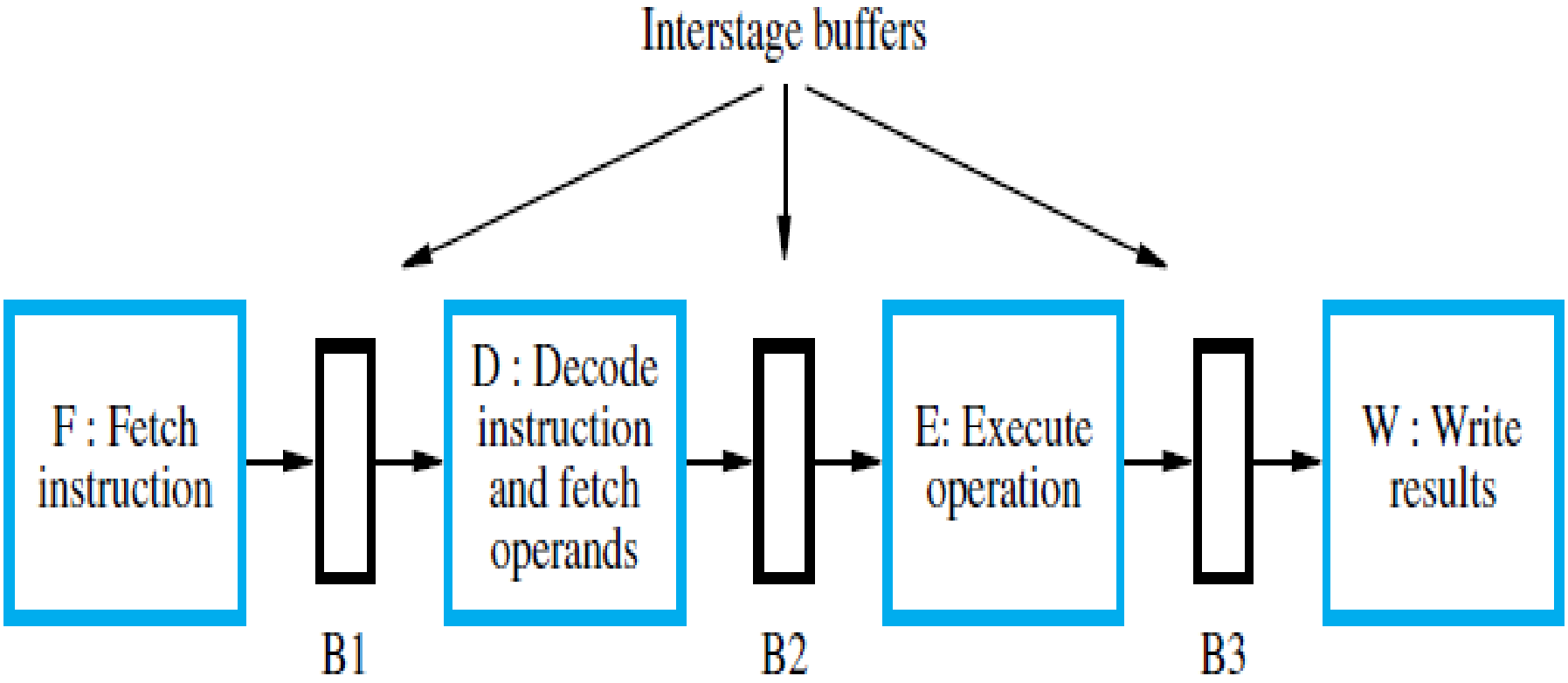
  Decode (D) **:** decode the instruction and fetch the source operand(s).

  Execute (E) : perform the operation specified by the instruction.

  Write (W) **:** store the result in the destination location.

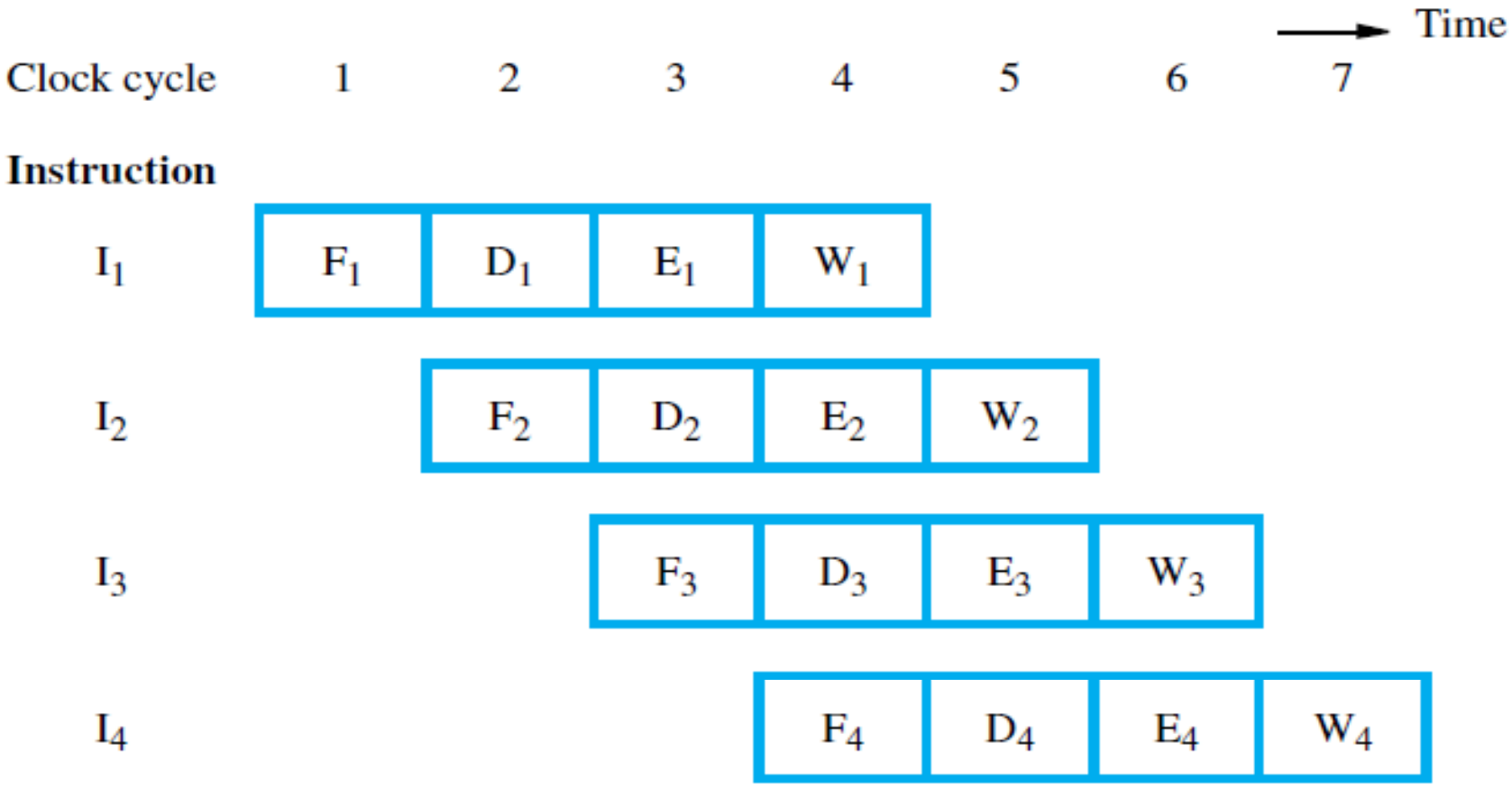
NOTE: Decoding stage implicitly consists of fetch operand stage.

A 4-Stage Pipelining



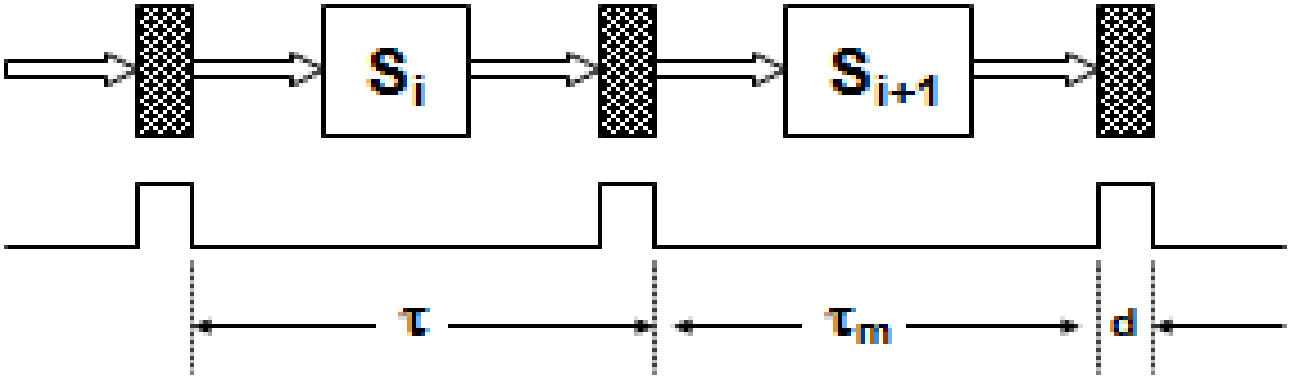
Hardware organization

A 4-Stage Pipelining



Instruction execution divided into four steps

Pipeline Performance



Clock cycle of the pipeline : τ

Latch delay : *d*

$$\tau = \max \{\tau_m\} + d$$

Pipeline frequency : *f*

$$f = 1 / \tau$$

Cntd…

k-stage pipeline processes *n* tasks in k + (n-1) clock cycles:

k cycles for the first task and *n-1* cycles for the remaining *n-1* tasks

Total time to process *n* tasks

$$T_k = [\, k + (n-1)] \, \tau$$

For the non-pipelined processor

$$T_1 = n \, k \, \tau$$

Speedup factor

$$S_k = \frac{T_1}{T_k} = \frac{n \, k \, \tau}{[k + (n-1)] \, \tau} - \frac{n \, k}{k + (n-1)}$$

Cntd…

Efficiency of the k-stages pipeline :

$$E_k = \frac{S_k}{k} = \frac{n}{k + (n-1)}$$

Pipeline throughput (the number of tasks per unit time) : note equivalence to IPC

$$H_k = \frac{n}{[\, k + (n-1)]\, \tau} = \frac{n\, f}{k + (n-1)}$$

An Example

Considering a 4 stage pipeline –

K= 4

Delay of each stage:

t1 = 15ns t2 = 25ns t3 = 45ns t4 = 30ns

Latch delay = 5ns

Total no. of tasks (n) = 100

Find ***Speed-up , Efficiency & Throughput***.

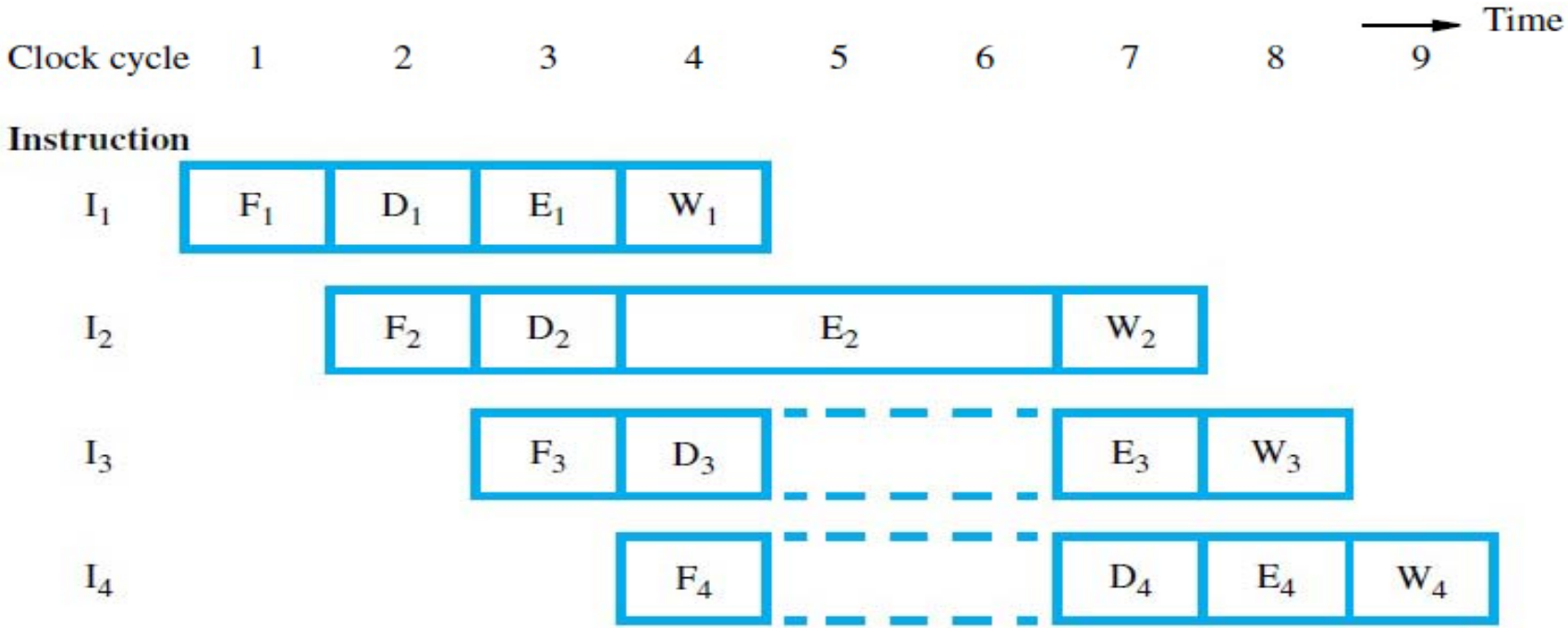
Pipeline Hazards

- Due to variety of reasons, one of the pipeline stages may not be able to complete its processing task for a given instruction within the specific time.

  e.g. Multiplication, division take more time to complete than other arithmetic operations.
- This causes other stages to stall (idle state).
- Any condition that causes a pipeline to stall is called a ***hazard***.
- Are of 3 types –
    1. Data Hazards
    2. Instruction or Control Hazards
    3. Structural Hazards

1. Data Hazards

- A ***data hazard*** is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls.
- stage E in the four-stage pipeline *is responsible for arithmetic and logic operations* ,and one clock cycle is assigned for this task.

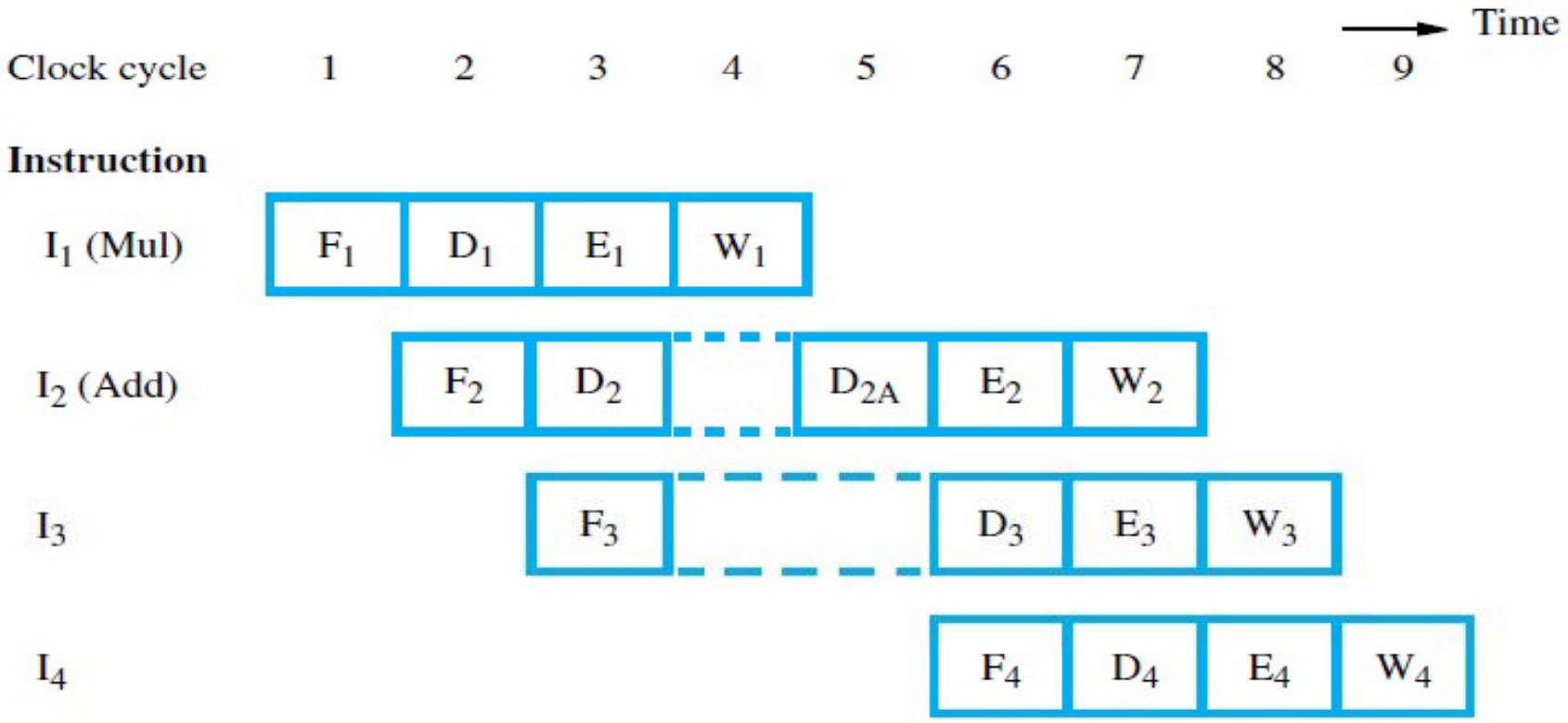


Data dependency

- A data hazard is a situation in which the pipeline is stalled because the data to be operated on are delayed for some reason, i.e. data dependency.

e.g. MUL R4,R2,R3 R4 ← R2 * R3

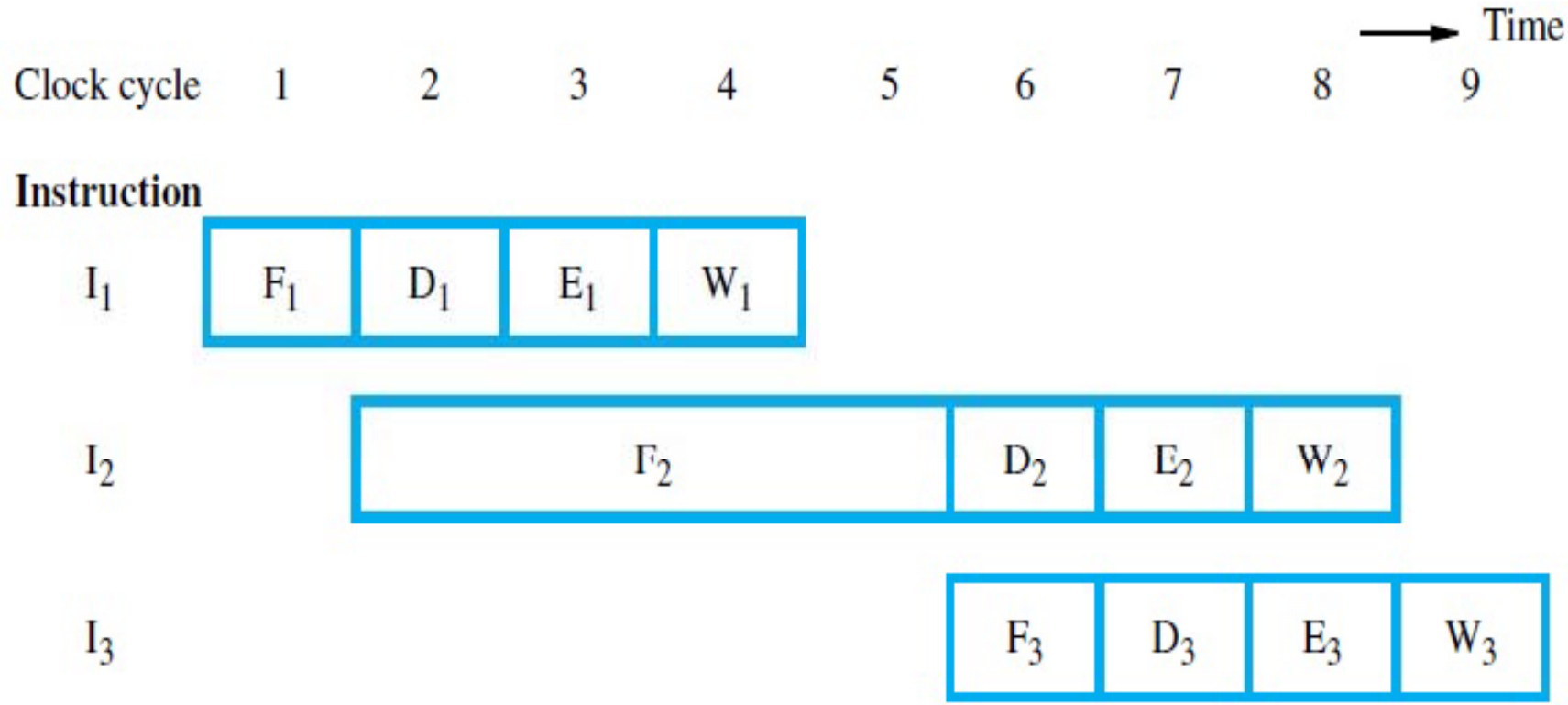
ADD R6,R4,R5 R6 ← R4 + R5



Pipeline stalled by data dependency between D_2 and W_1

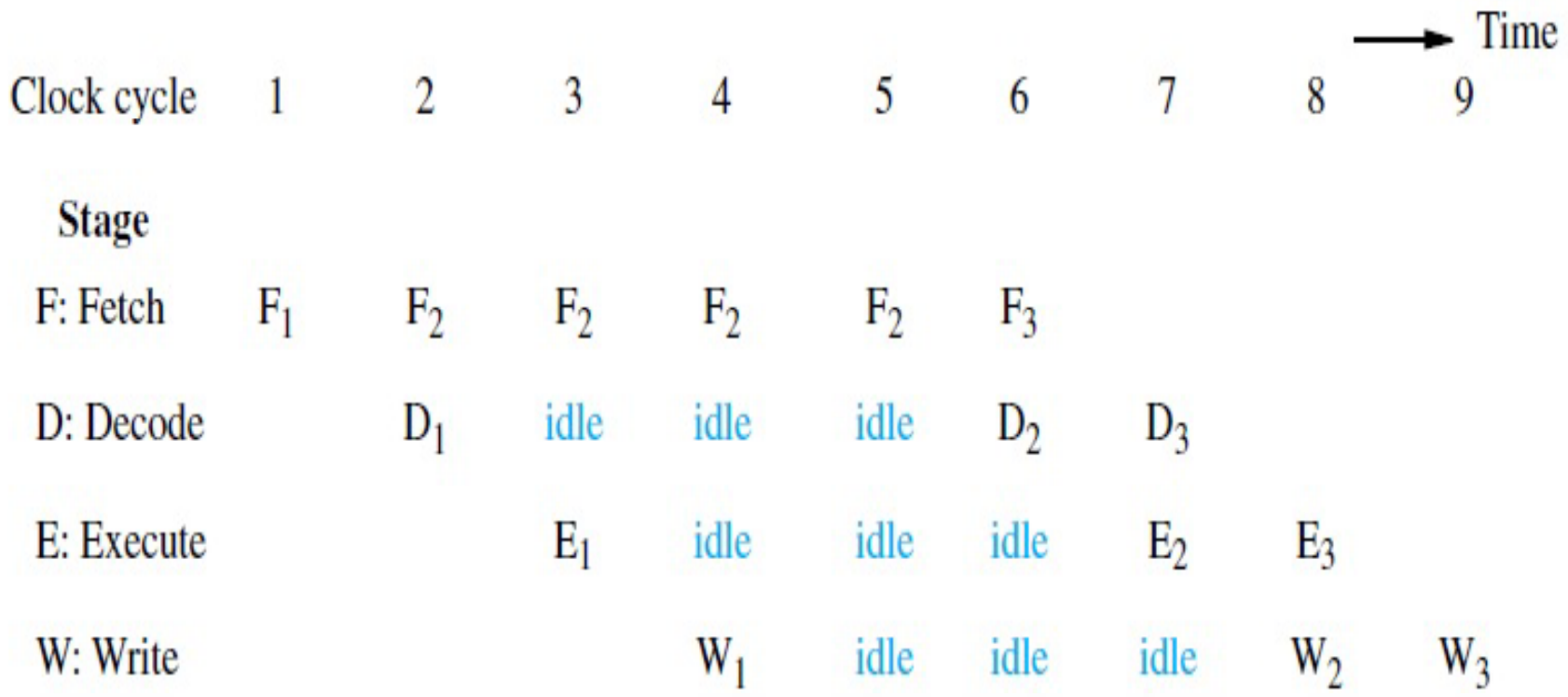
2. Instruction or Control Hazards

- The pipeline may also be stalled because of a delay in the availability of an instruction. For example, this may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. Such hazards are often called ***control hazards or instruction hazards***.



Pipeline stall caused by a cache miss in F2

Cntd…

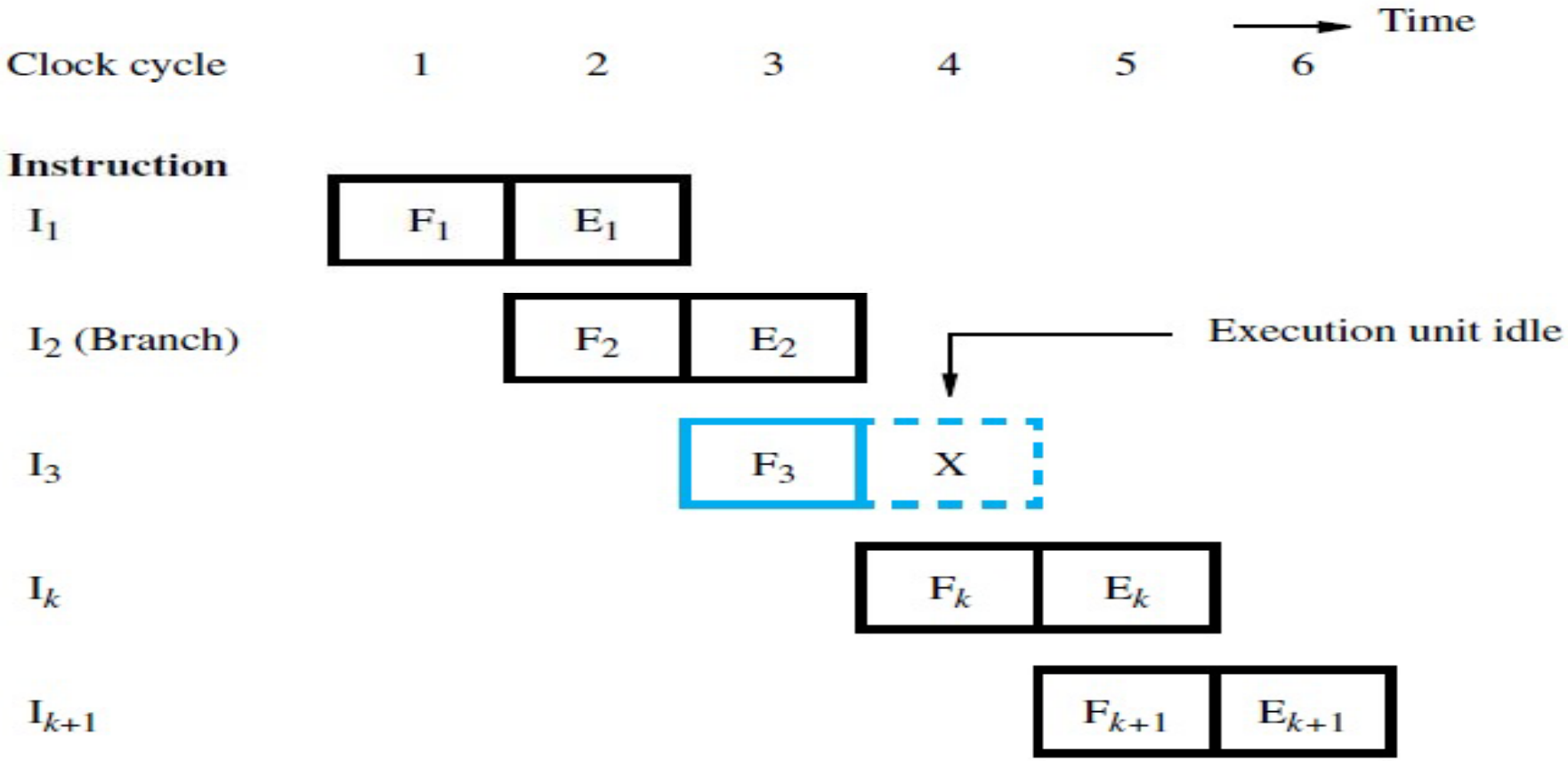


Function performed by each processor stage in successive clock cycles

Pipeline stall caused by a cache miss in F2

Branch Penalty

- The time lost as a result of a branch instruction is often referred to as the ***branch penalty***.
- Can be considered as ***instruction hazard***.



An idle cycle caused by a branch instruction.

3. Structural Hazards

- This is the situation when two instructions require the use of a given hardware resource at the same time. The most common case in which this hazard may arise is in access to memory.
- One instruction may need to access memory as part of the Execute or Write stage while another instruction is being fetched.
- If instructions and data reside in the same cache unit, only one instruction can proceed and the other instruction is delayed. Many processors use separate instruction and data caches to avoid this delay.

e.g. LOAD R1 , X(R2) R1 ⟵ X + [R2]

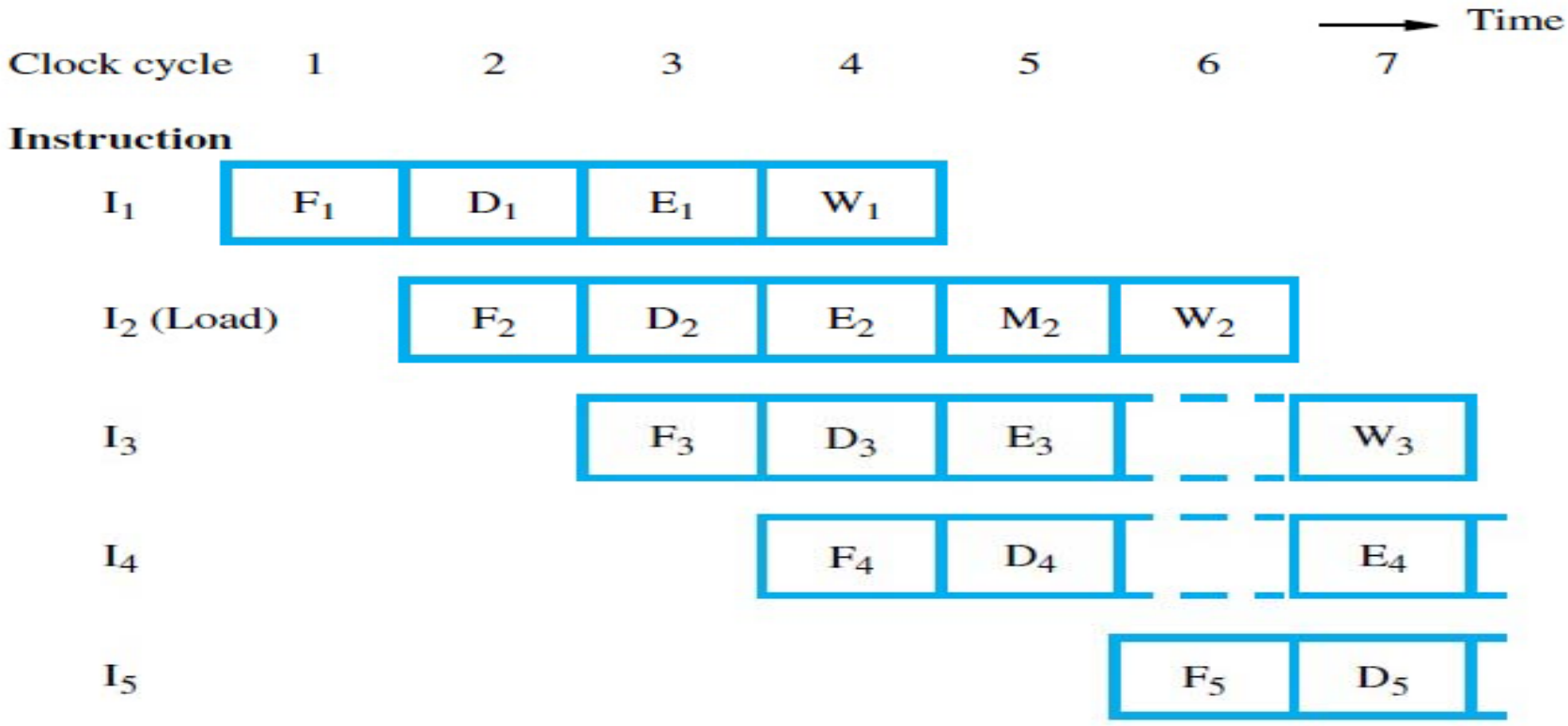
The execution stage performs 2 tasks to execute this instruction –

1. computation of memory address => 1 clock cycle
2. The operand read from memory is written into register R1

=> 1 clock cycle

Therefore, 2 clock cycles are required but only one is assigned => stall in pipeline

Cntd…



Effect of a Load instruction on pipeline timing.

Even though the instructions and their data are all available, the pipeline is stalled because one hardware resource, the register file, cannot handle two operations at once. If the register file had two input ports, that is, if it allowed two simultaneous write operations, the pipeline would not be stalled. In general, structural hazards are avoided by providing sufficient hardware resources on the processor chip.


Q+A

1. Generally pipeline consists of …. Stages.

   a) 2 b) 3 c) 4 d) 5

2. Who is responsible to hold information being passed for one stage to the next ?

   a) Multiplexer b) Flip-Flop c) Encoder d) Buffer

3. For a k-stage pipelining, each instruction requires …….. Clock cycles.

   a) k b) k + (n-1) c) k – (n + 1) d) None of these

4. For a k-stage pipelining, n instructions require …….. Clock cycles.

   a) k b) k + (n-1) c) k – (n + 1) d) None of these

5. Pipeline throughput means …….

   a) no. of instructions /clock cycle

   b) no. of instructions /unit time

   c) no. of instructions /total clock cycle

   d) no. of instructions /total time

6. Data dependency can cause …… hazard.

   a) Instruction b) Data c) Structural d) all of these

7. Branch instruction can cause …… hazard.

   a) Instruction b) Data c) Structural d) all of these

8. Displacement addressing can cause …… hazard.

   a) Instruction b) Data c) Structural d) all of these

9. Delay to fetch instructions can cause …… hazard.

   a) Instruction b) Data c) Structural d) all of these

10. Multiple memory reference at a time can cause …… hazard.

    a) Instruction b) Data c) Structural d) all of these

11. Multiplication/division operation can cause …… hazard.

    a) Instruction b) Data c) Structural d) all of these

12. Performance of a pipeline can be measured by …….

    a) Speed-up Factor b) Efficiency

    c) Throughput d) All of these

13. New information is loaded into the buffer at the ….. each clock cycle.

    a) beginning b) end c) middle d) none of these

14. For pipelining, latency of a single task …….
    a) same as in sequential processing
    b) differs from sequential processing
    c) can't be determined
    d) none of these
15. Pipelining keeps CPU busy all the time
    a) True b) False
16. Which stage is responsible for performing ALU operations ?
    a) Fetch b) Decode c) Execute d) Write back
17. Internal interrupt can cause …….. Hazard.
    a) data b) control c) structural d) all of these

Reference: Carl Hamacher

Thank You