VLSI SYSTEMS AND ARCHITECTURE

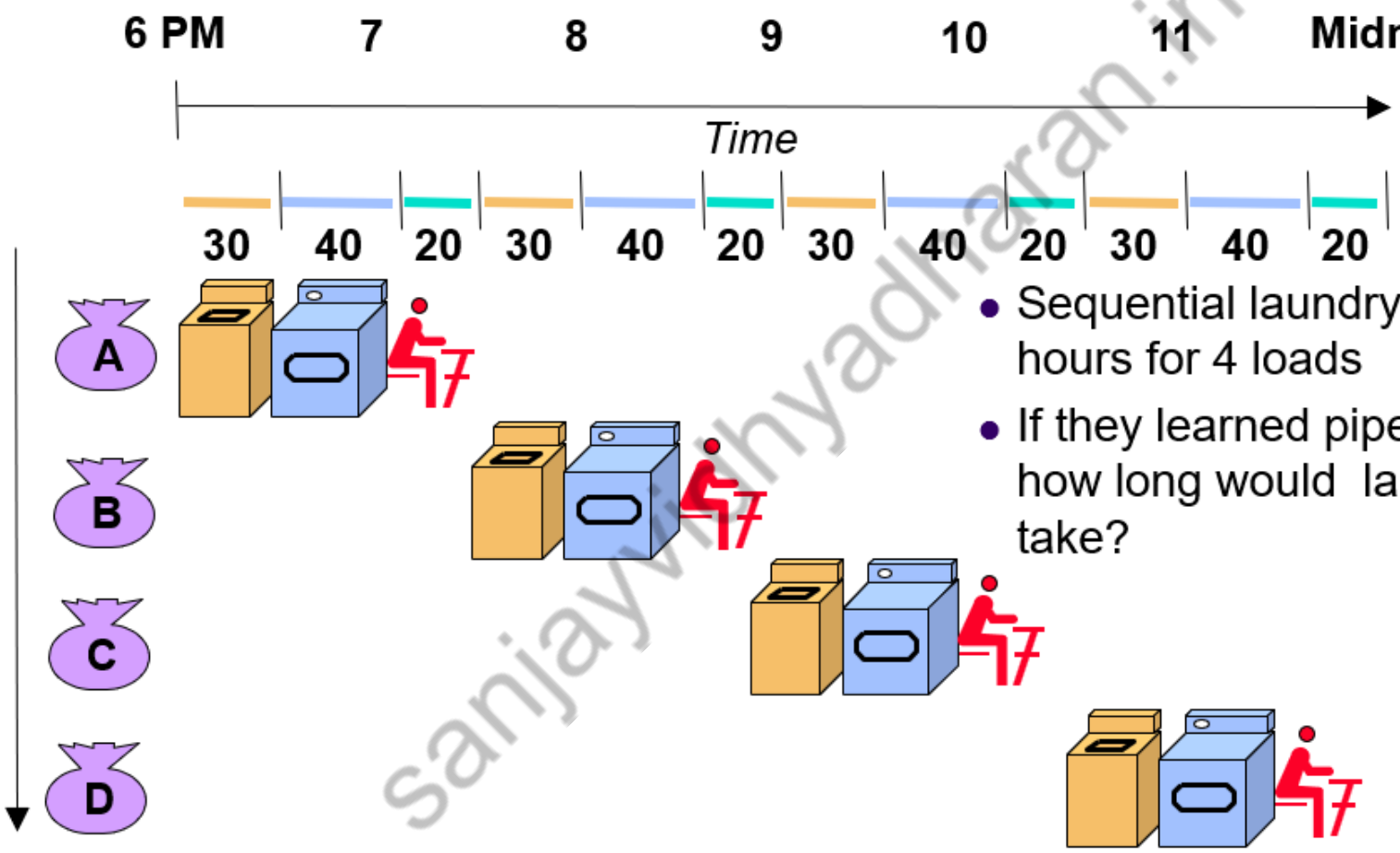
2021-22

Lecture 11

Pipelined Architecture

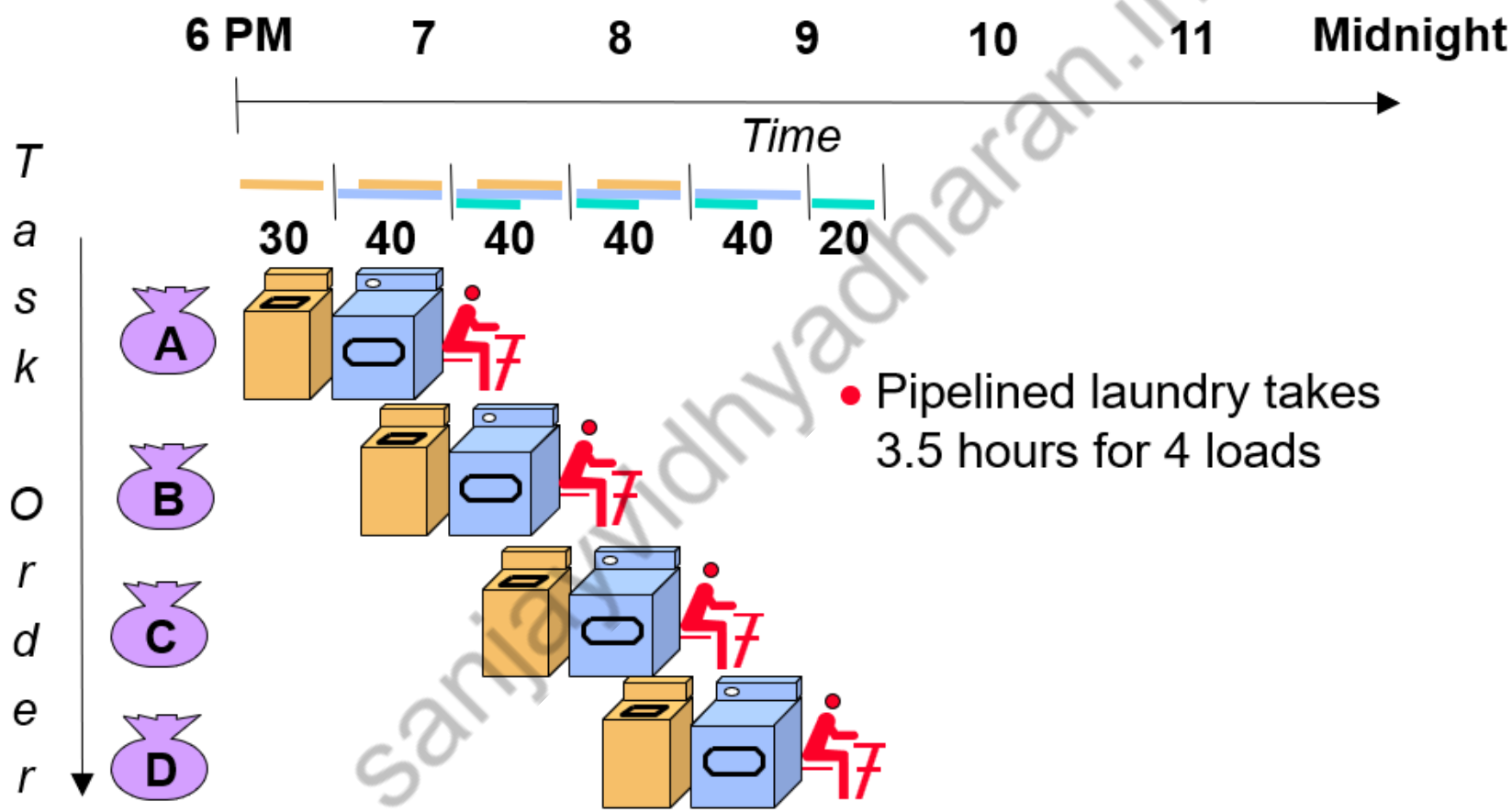
By Dr. Sanjay Vidhyadharan

Pipeline Concept



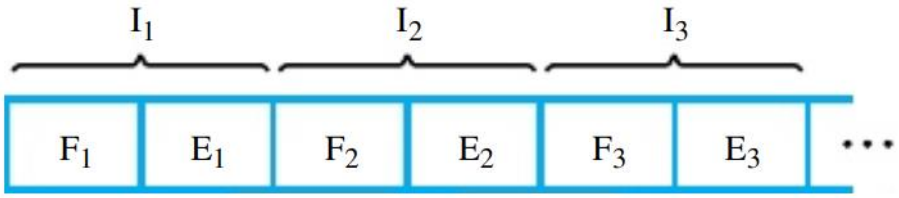
- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Pipeline Concept

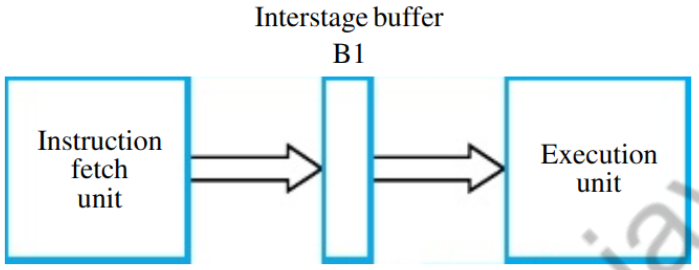


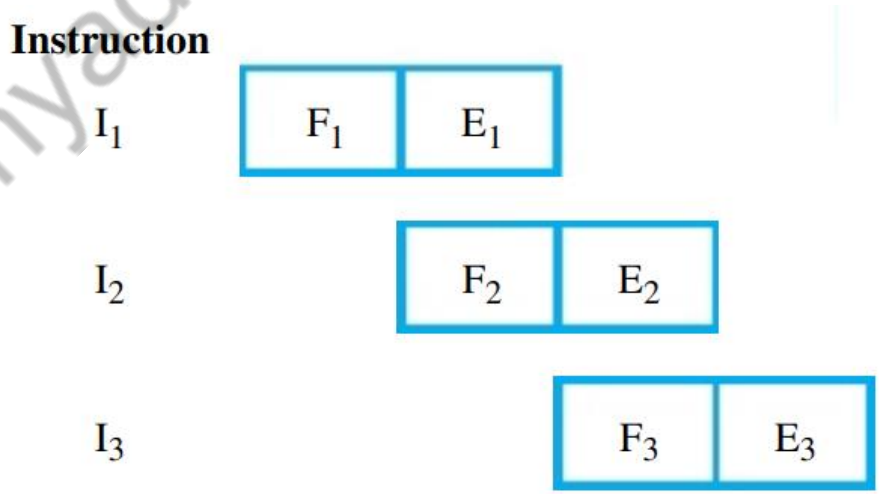
Pipeline Concept

Sequential execution



Pipelined execution





1. Two separate hardware units
2. Buffer (Register o Hold data)

Pipeline Concept

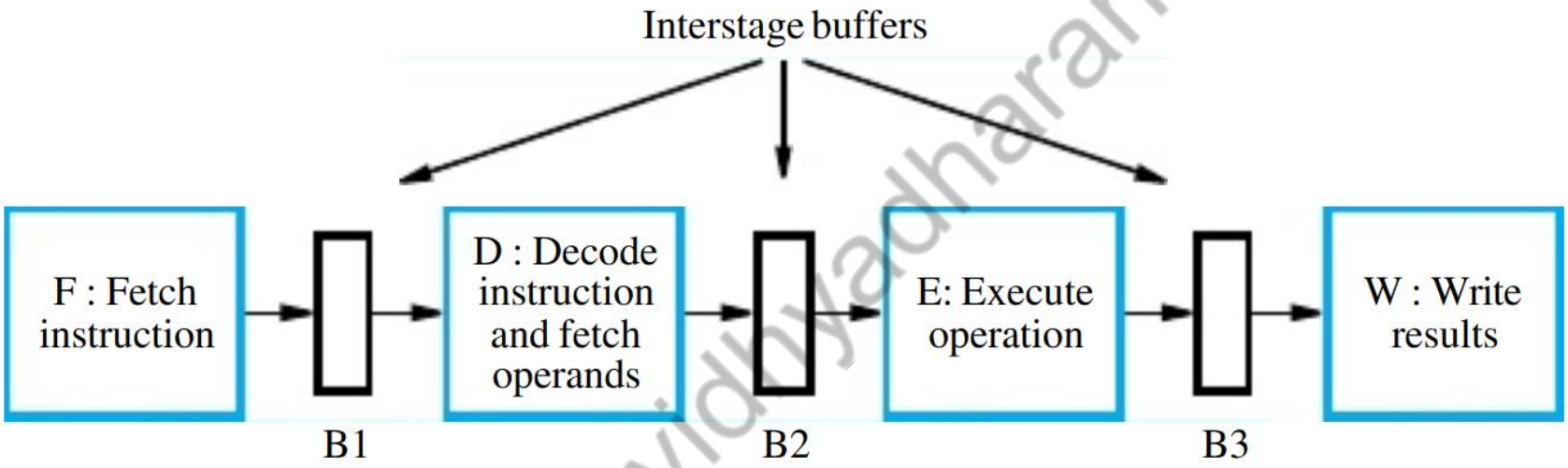
Pipelined execution

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

- F : Fetch: read the instruction from the memory.
- D : Decode: decode the instruction and fetch the source operand(s).
- E : Execute: perform the operation specified by the instruction.
- W : Write: store the result in the destination location.

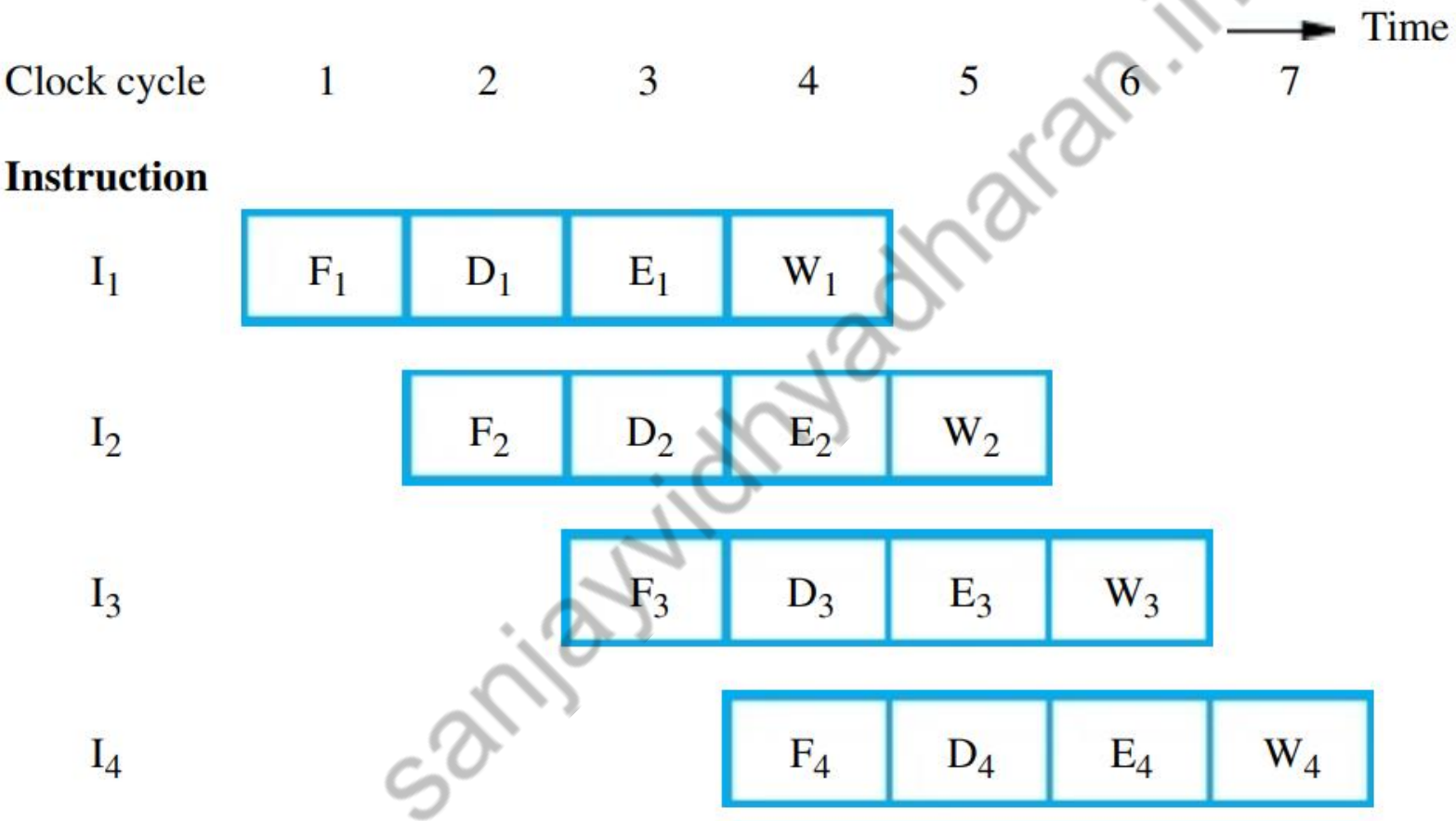
Pipeline Concept

Hardware organization



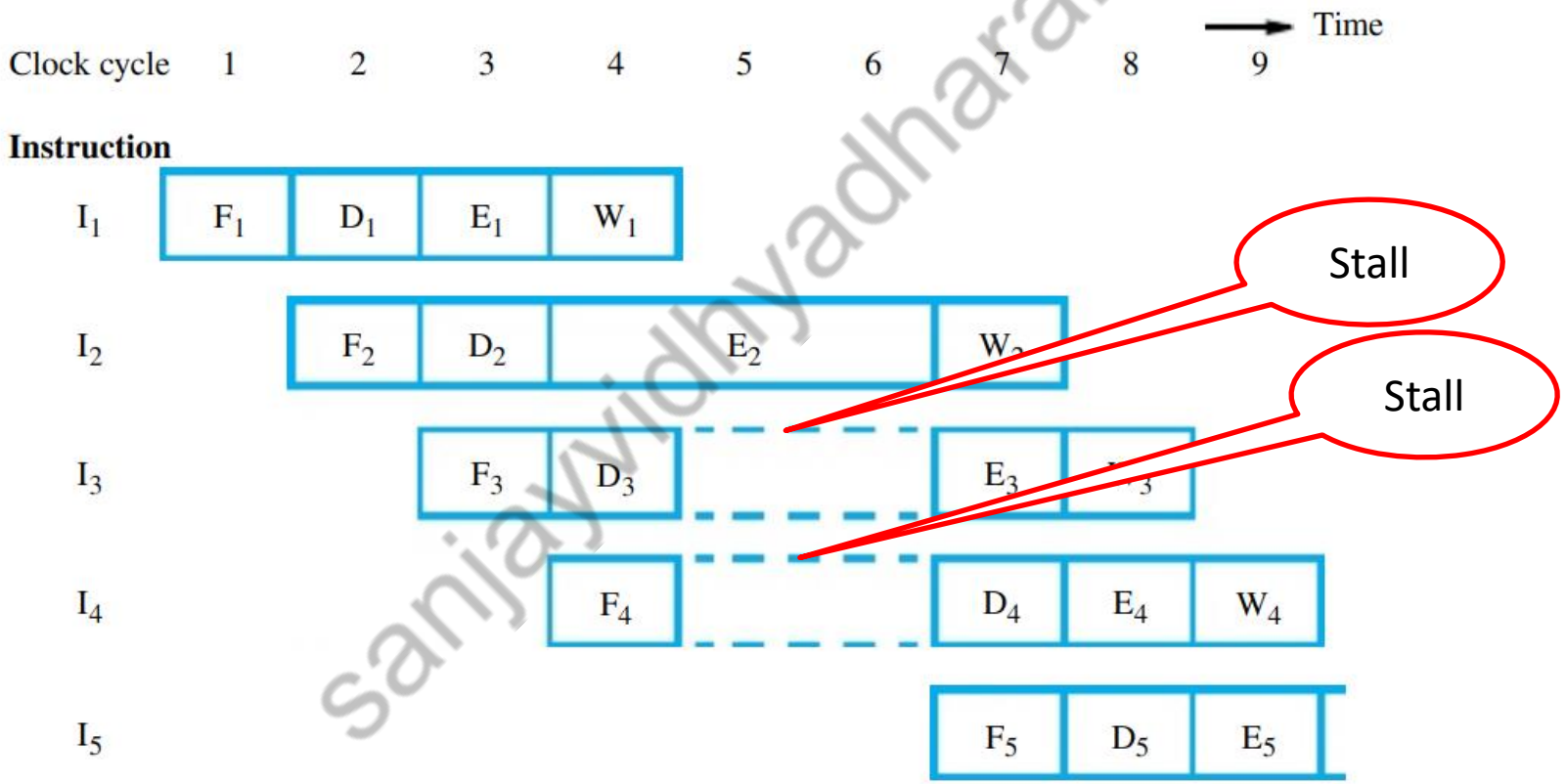
Each stage in a pipeline is expected to complete its operation in one clock cycle. Hence, the clock period should be sufficiently long to complete the task being performed in any stage.

Pipeline Concept

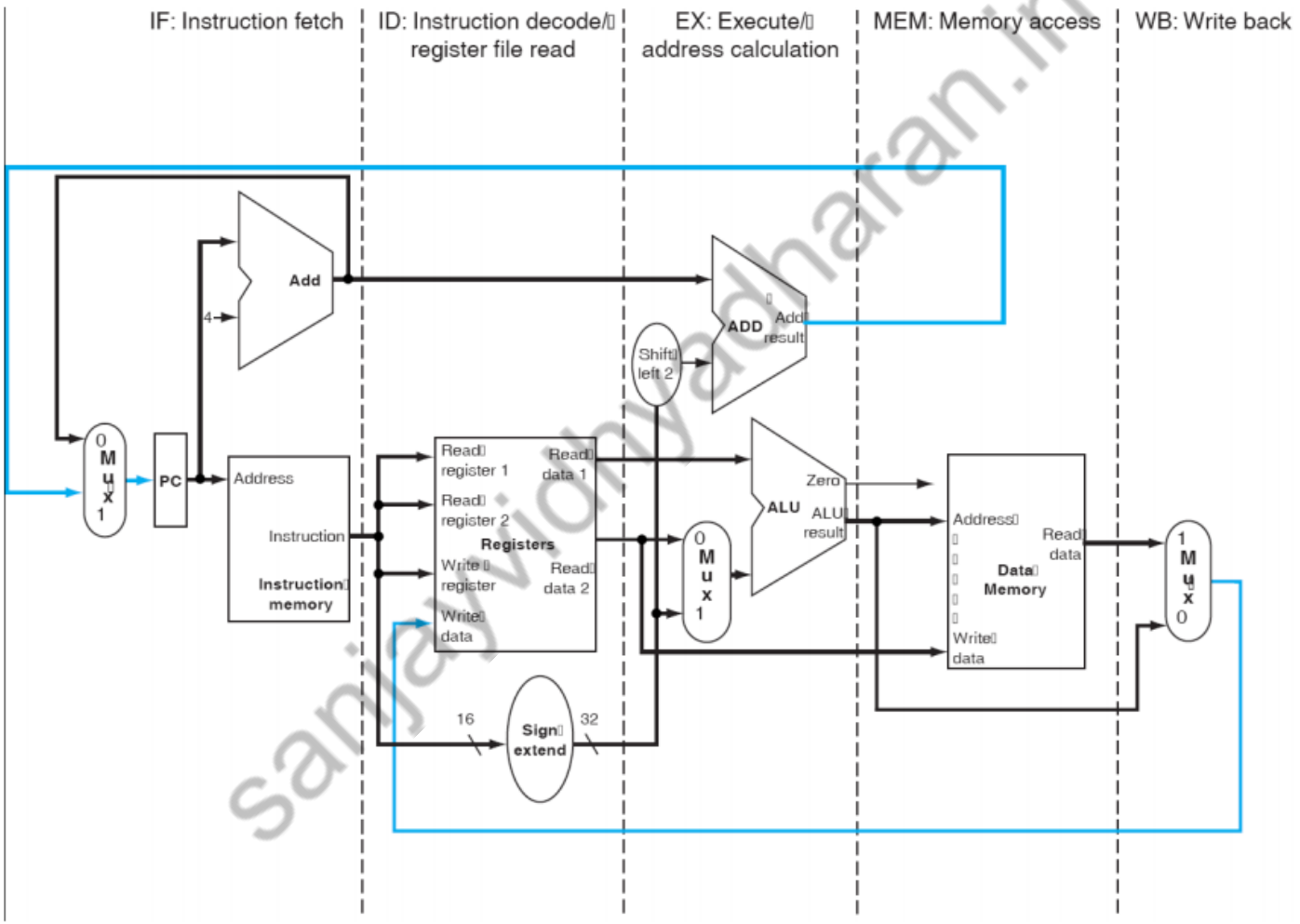


Pipeline Concept

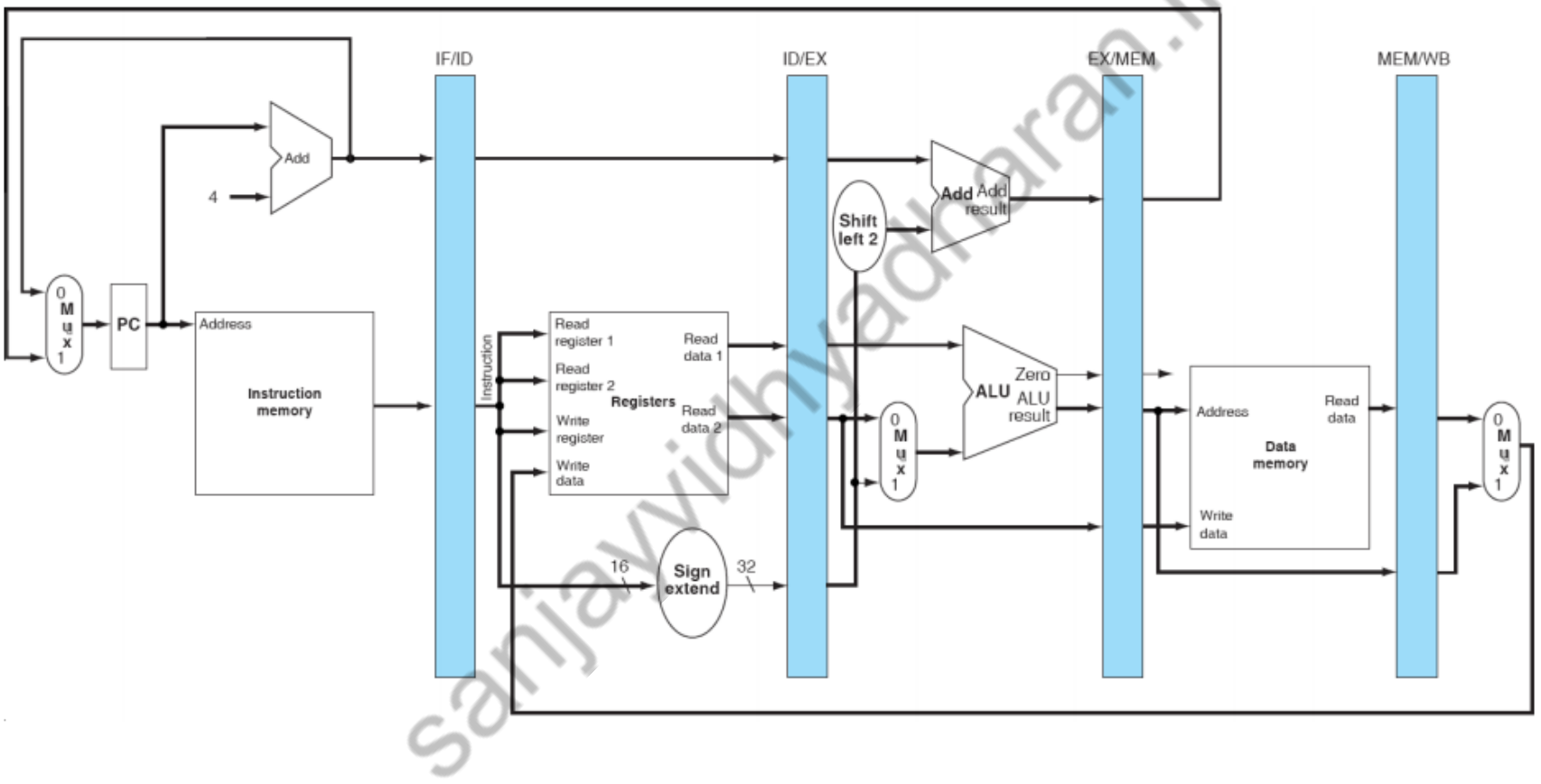
Why RISC is better for Pipeling



Review - Single-Cycle Processor



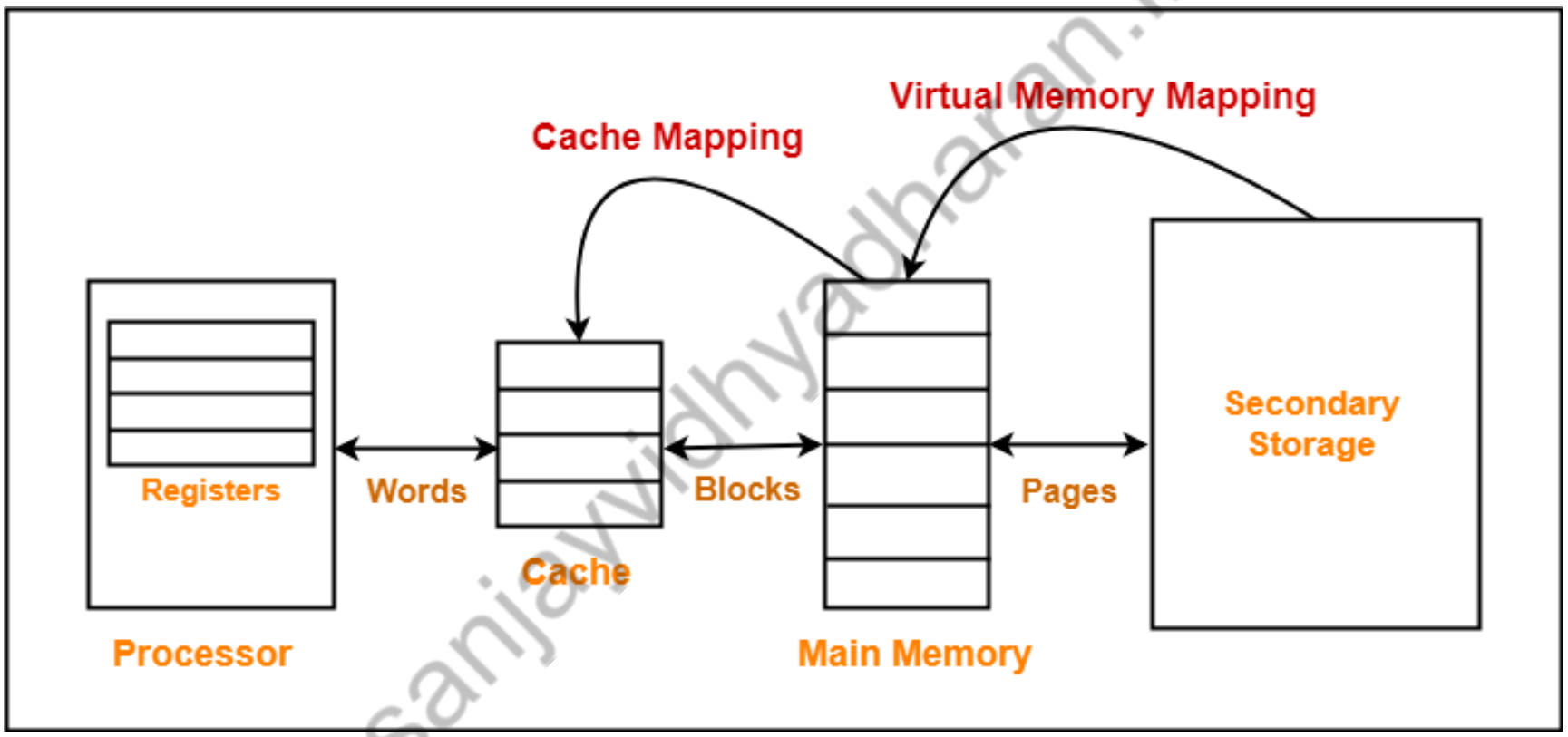
Basic Pipelined Processor



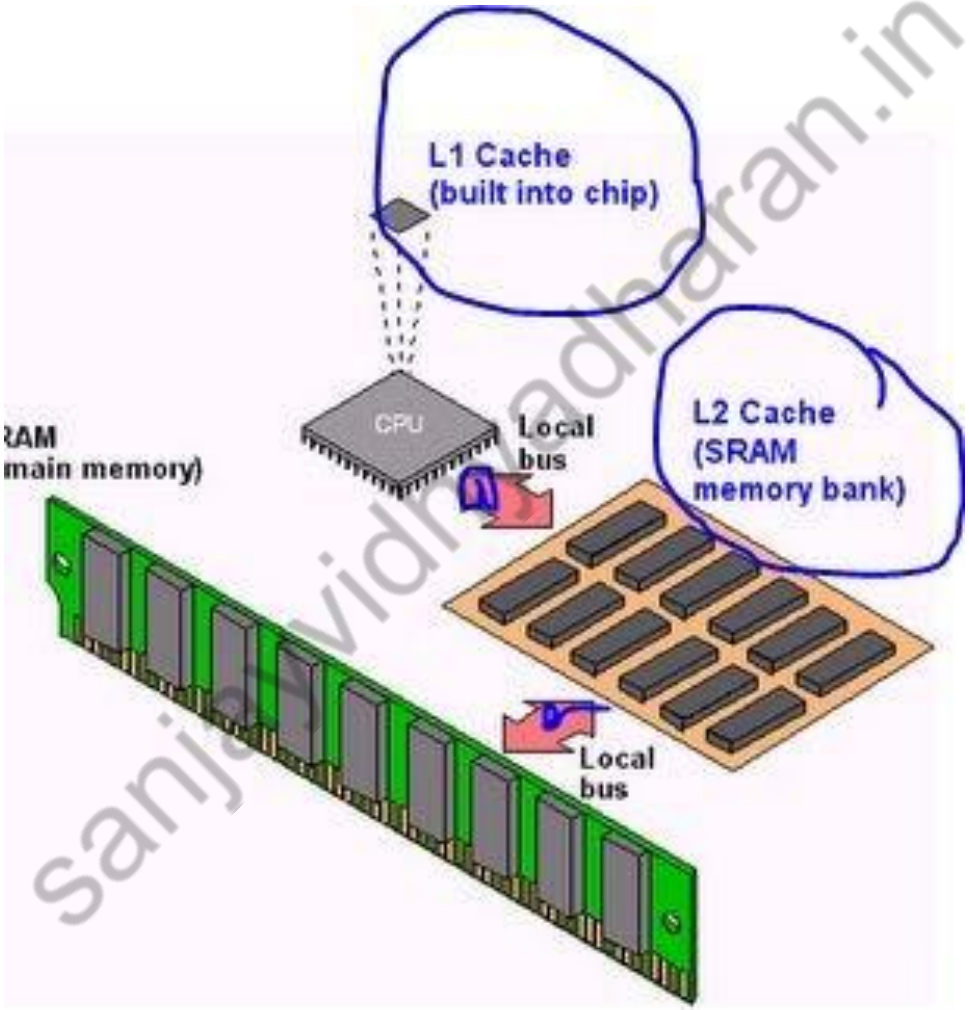
Role of Cache Memory

- Each pipeline stage is expected to complete in one clock cycle.
- The clock period should be long enough to let the slowest pipeline stage to complete.
- Faster stages can only wait for the slowest one to complete.
- Since main memory is very slow compared to the execution, if each instruction needs to be fetched from main memory, pipeline is almost useless.
- Fortunately, we have cache.

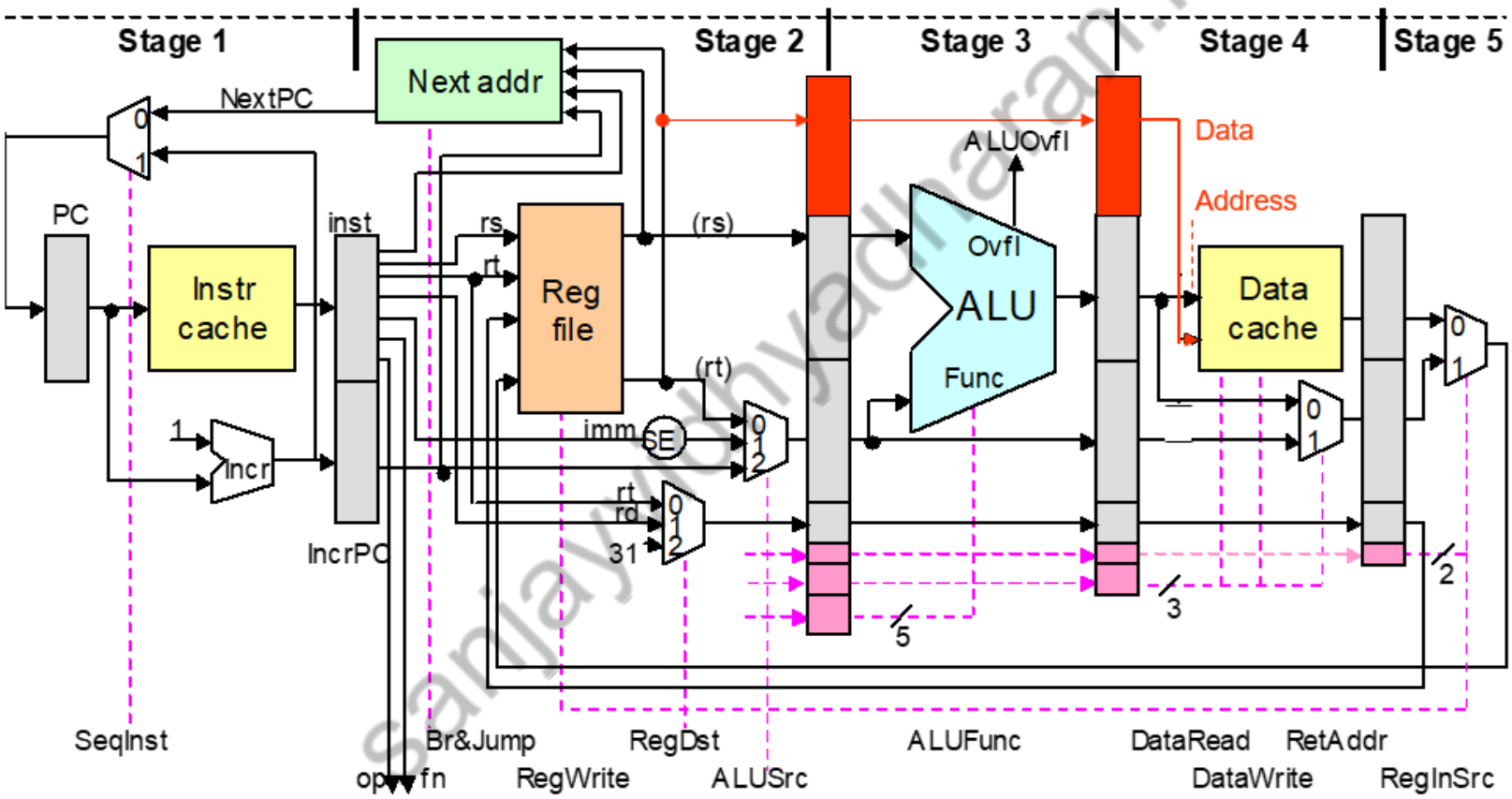
Cache Memory



Cache Memory



Cache Memory

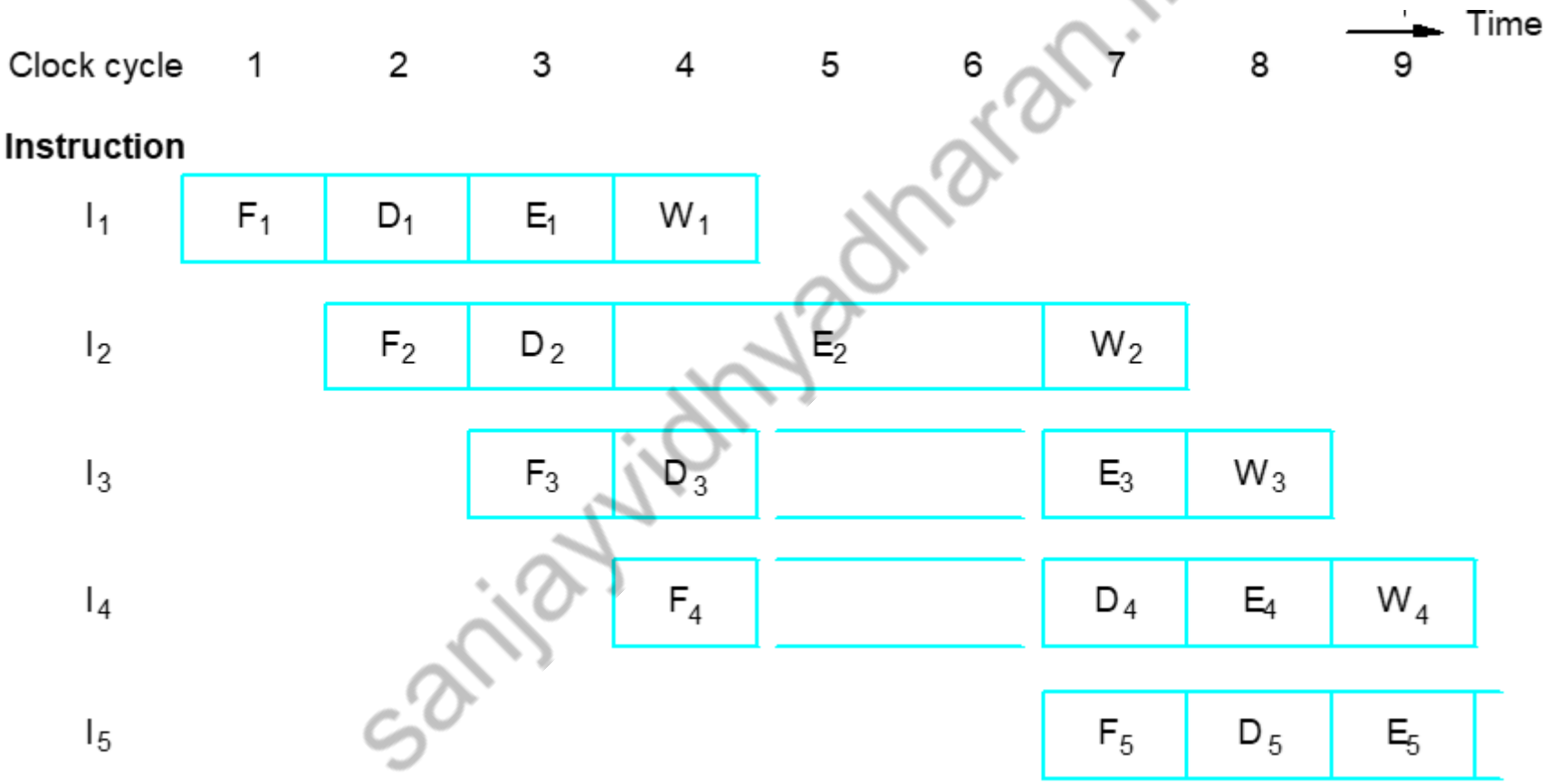


Pipeline Performance

- The potential increase in performance resulting from pipelining is proportional to the number of pipeline stages.
- However, this increase would be achieved only if all pipeline stages require the same time to complete, and there is no interruption throughout program execution.
- Unfortunately, this is not true.

Pipeline Performance

Effect of an Execution Operation taking more than one Clock Cycle

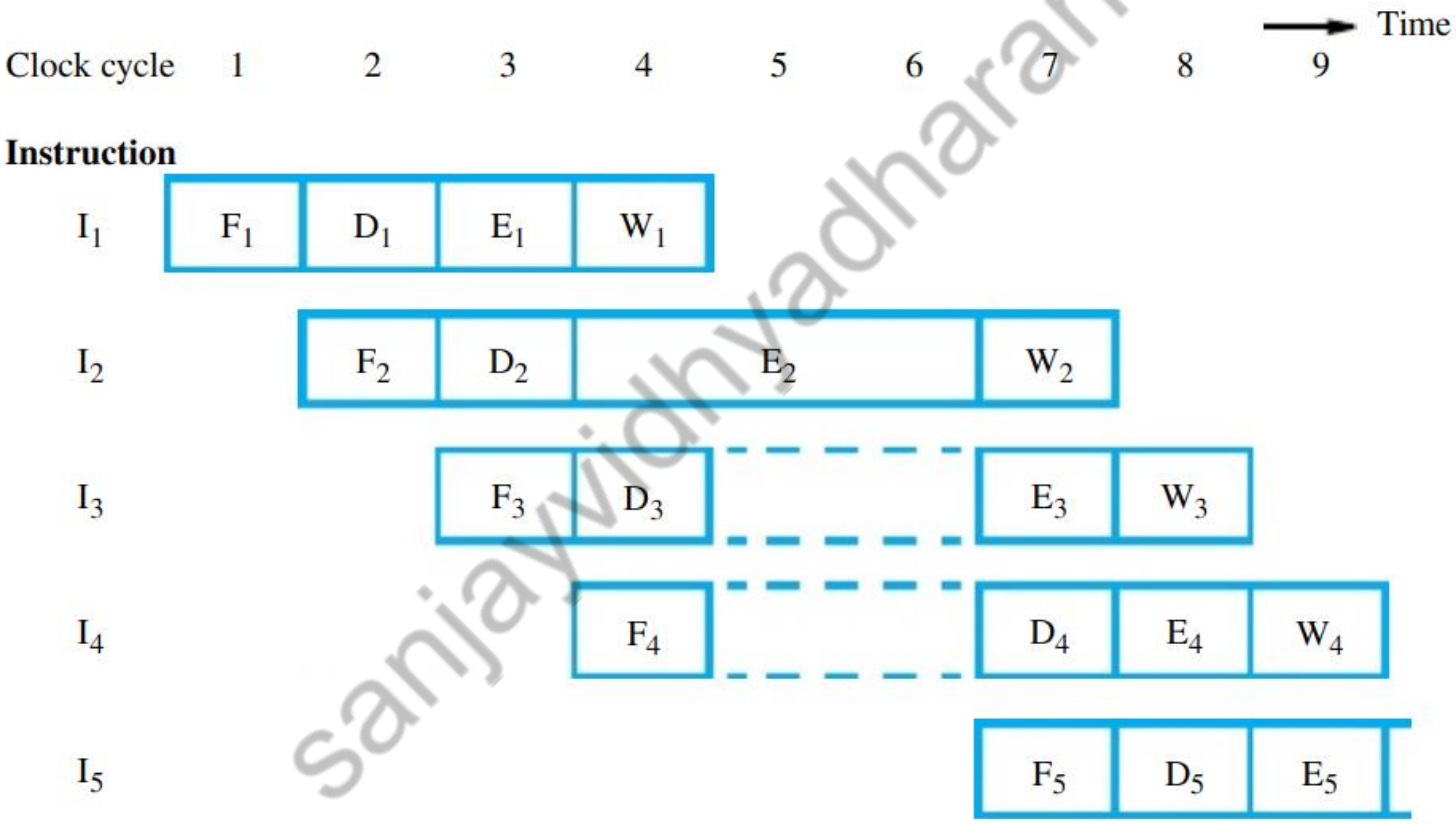


Pipeline Hazard

- The previous pipeline is said to have been stalled for two clock cycles.
- Any condition that causes a pipeline to stall is called a **hazard**.
- **Data hazard** – any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. So some operation has to be delayed, and the pipeline stalls.
- **Instruction (control) hazard** – a delay in the availability of an instruction causes the pipeline to stall.
- **Structural hazard** – the situation when two instructions require the use of a given hardware resource at the same time.

Data Hazard

Certain ALU operations like multiplication, division required more than one clock cycles



Data Hazard

- We must ensure that the results obtained when instructions are executed in a pipelined processor are identical to those obtained when the same instructions are executed sequentially.
- Hazard occurs

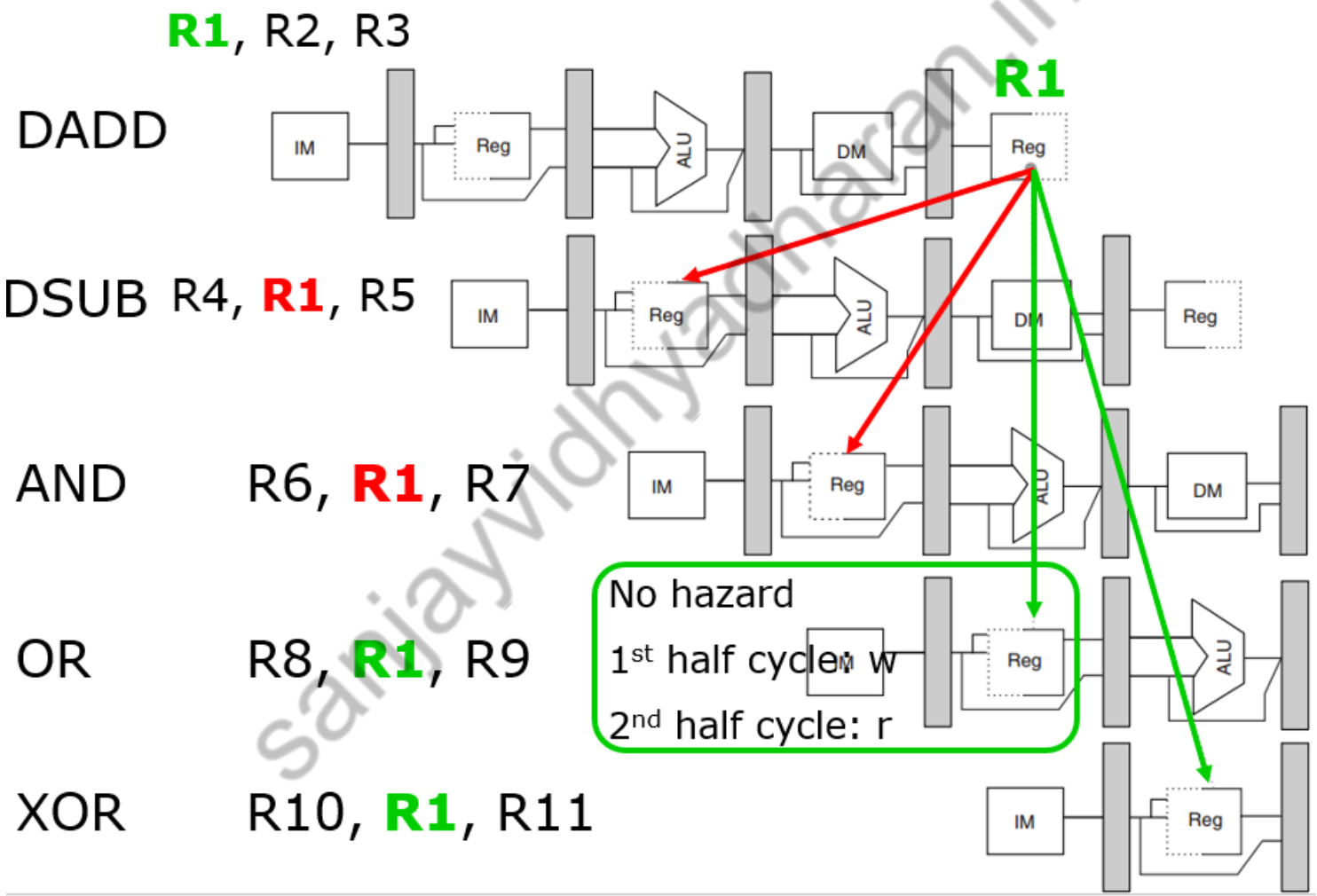
  $A \leftarrow 3 + A$

  $B \leftarrow 4 \times A$
- No hazard

  $A \leftarrow 5 \times C$

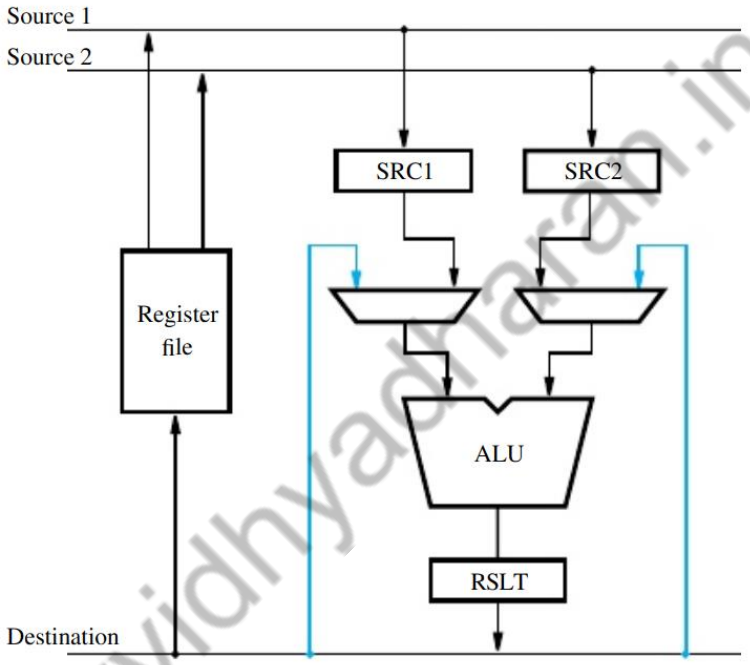
  $B \leftarrow 20 + C$
- When two operations depend on each other, they must be executed sequentially in the correct order.

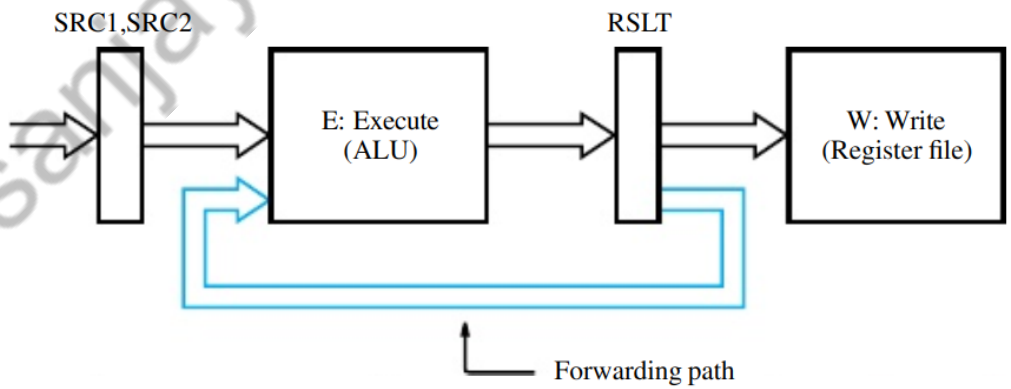
Data Hazard



OPERAND FORWARDING

$A \leftarrow 3 + A$
$B \leftarrow 4 \times A$



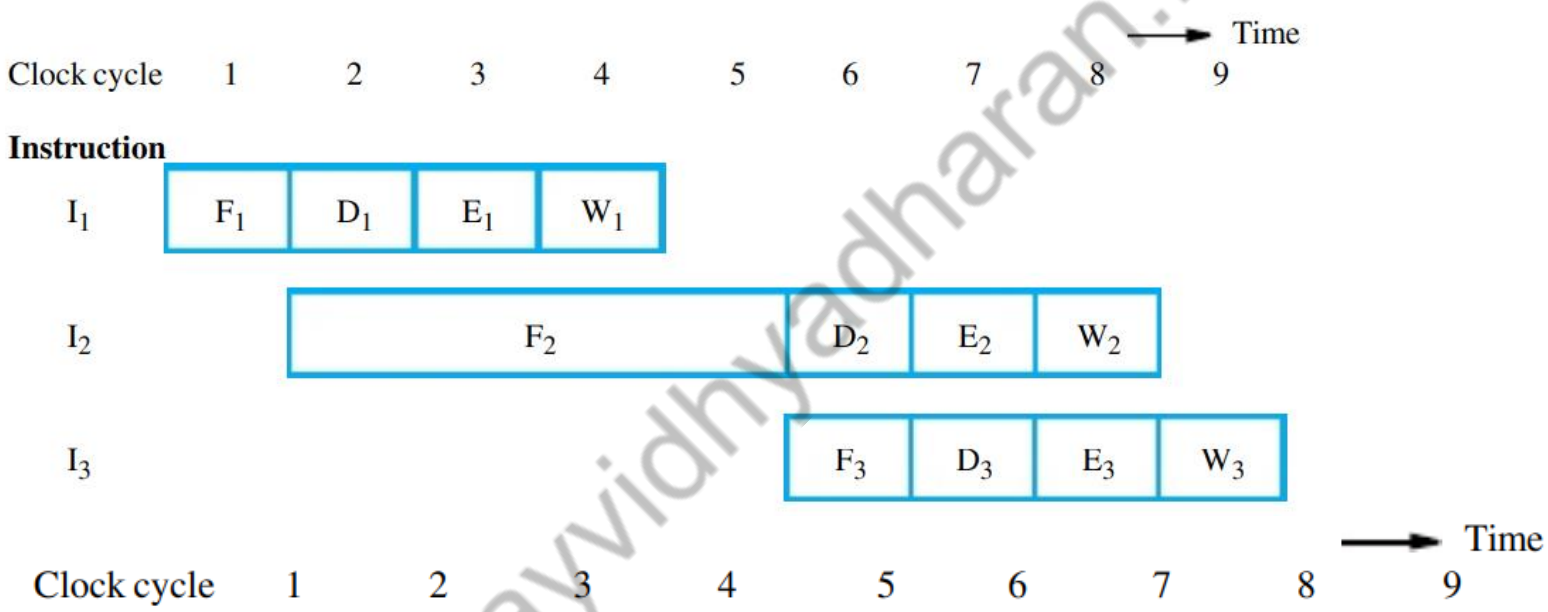


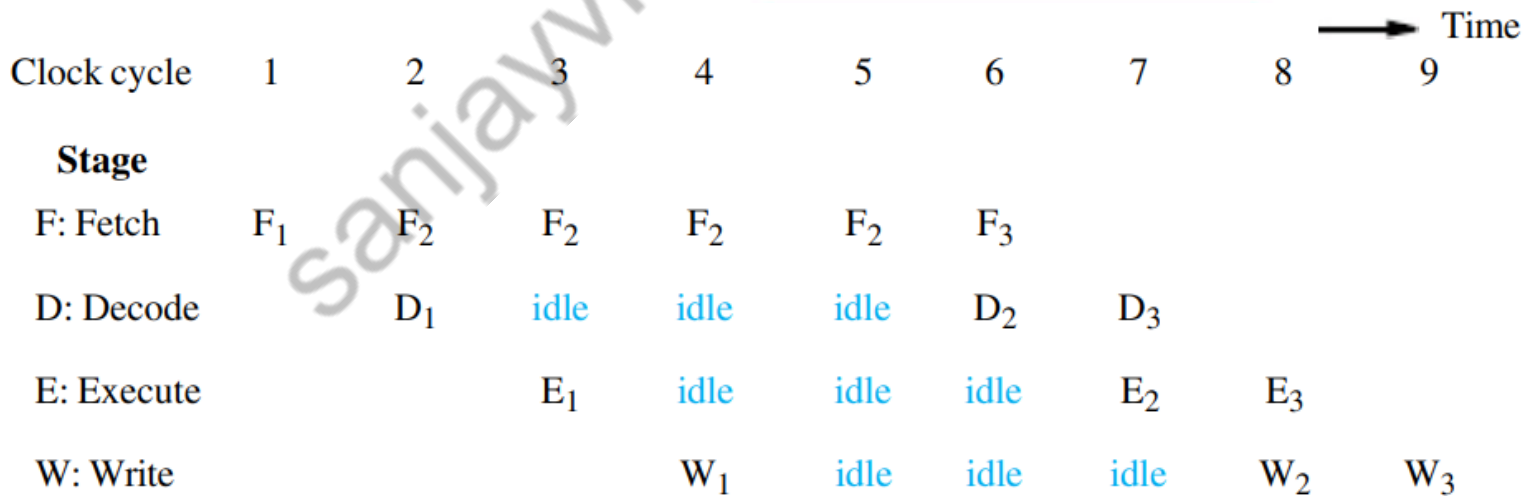
HANDLING DATA HAZARDS IN SOFTWARE

```
I1:   Mul    R2,R3,R4
      NOP
      NOP
I2:   Add    R5,R4,R6
```

Instruction (Control) Hazard

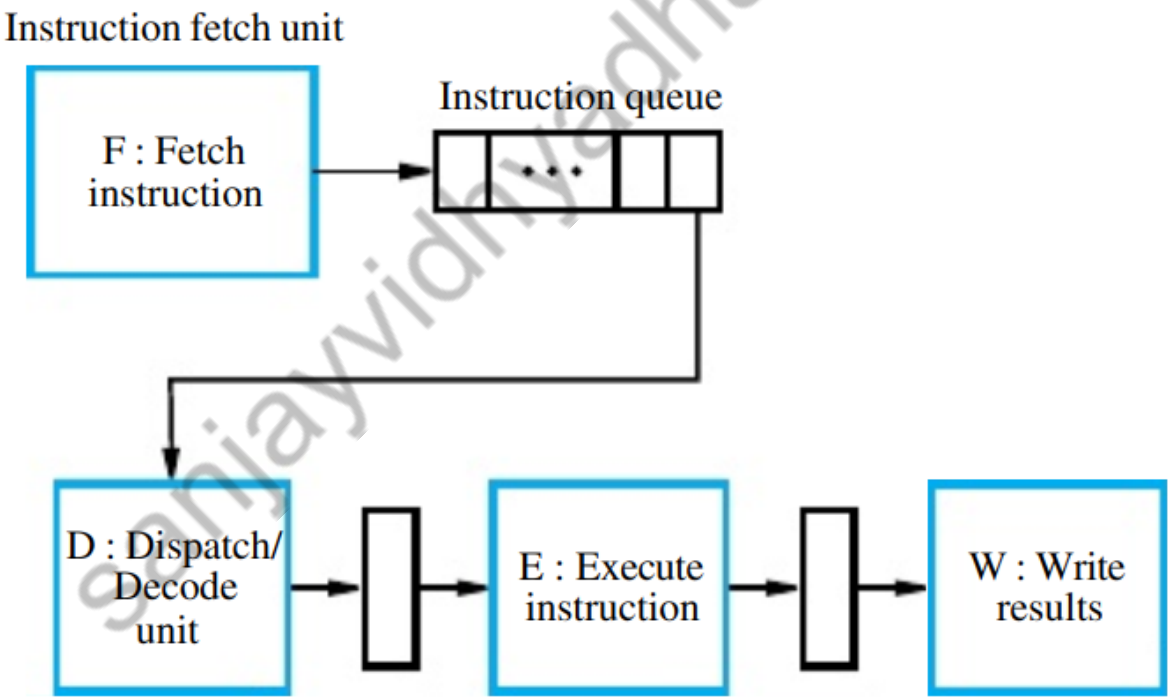
This may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. (E.g. Unconditional Branches)





Instruction Queue and Prefetching

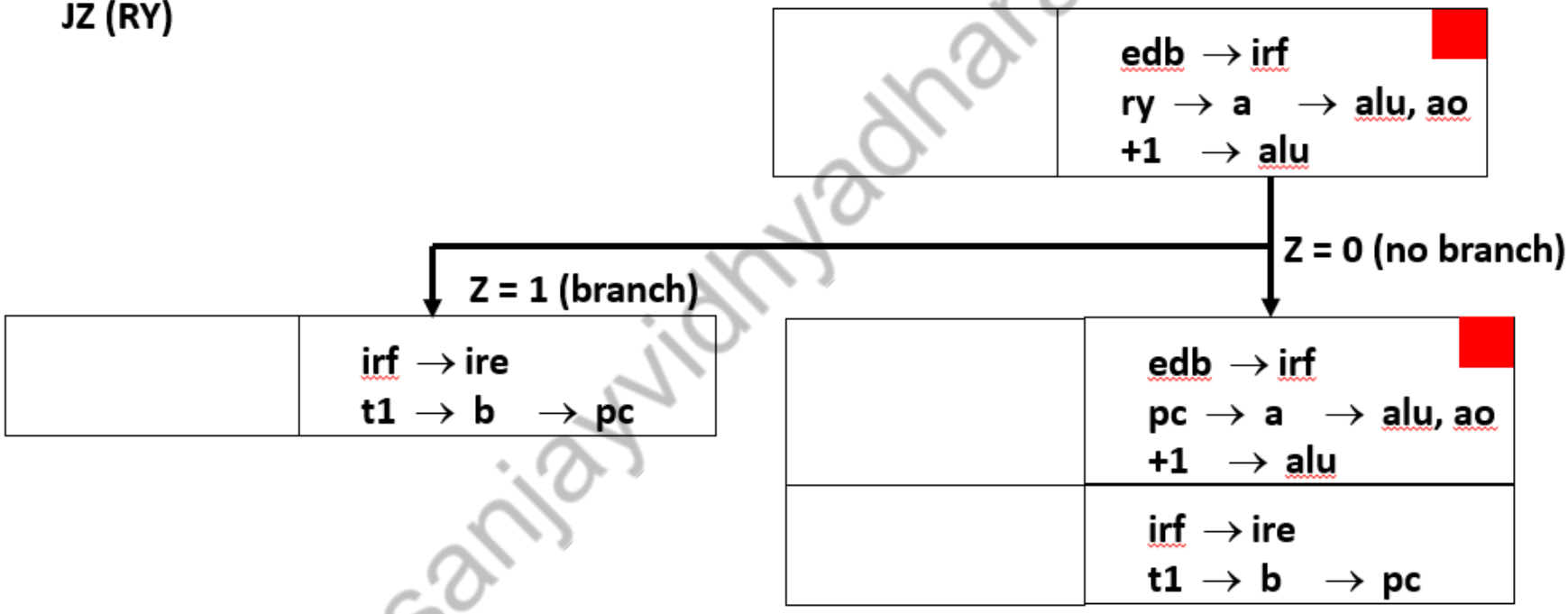
Either a cache miss or a branch instruction stalls the pipeline for one or more clock cycles. To reduce the effect of these interruptions, many processors employ sophisticated fetch units that can fetch instructions before they are needed and put them in a queue. Typically, the instruction queue can store several instructions.



Branch Prediction

Z : Zero Flag

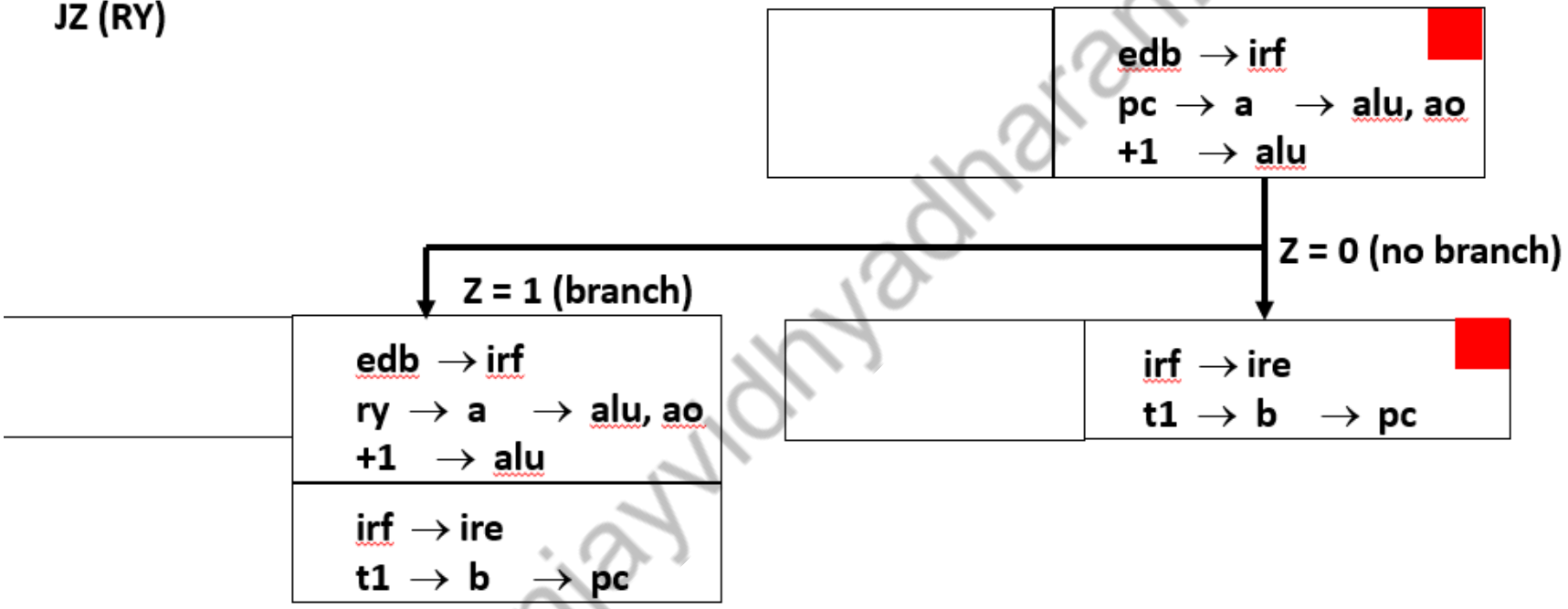
JZ (RY)



Branch Prediction

Z : Zero Flag

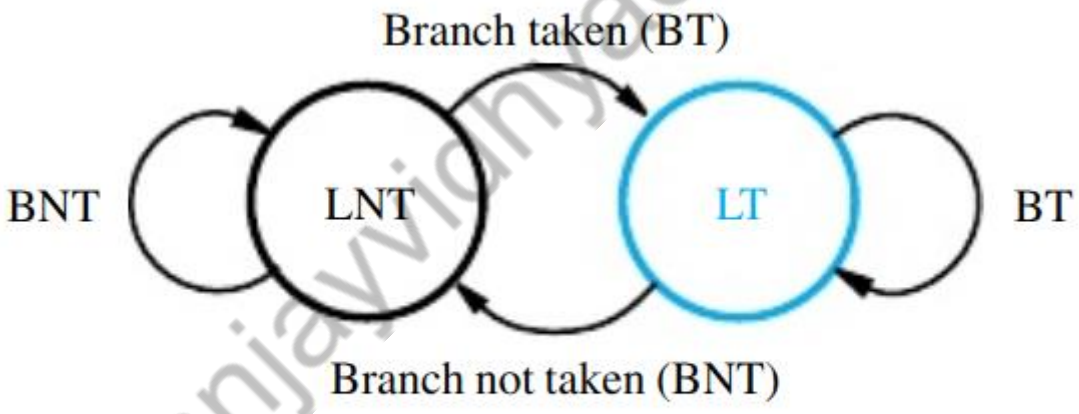
JZ (RY)



Dynamic Branch Prediction

LT: Branch is likely to be taken

LNT: Branch is likely not to be taken



(a) A 2-state algorithm

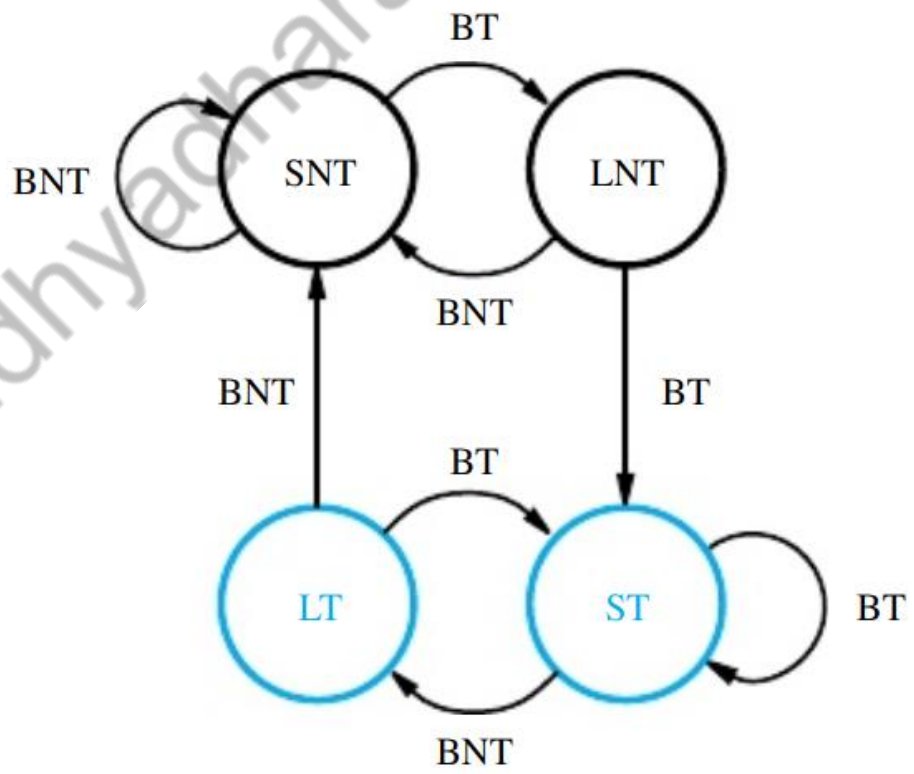
Dynamic Branch Prediction

ST: Strongly likely to be taken

LT: Likely to be taken

LNT: Likely not to be taken

SNT: Strongly likely not to be taken



Addressing Modes

Useful addressing modes include
index,
indirect,
autoincrement,
and autodecrement.

Many processors provide various combinations of these modes to increase the flexibility of their instruction sets. Complex addressing modes, such as those involving double indexing, are often encountered.

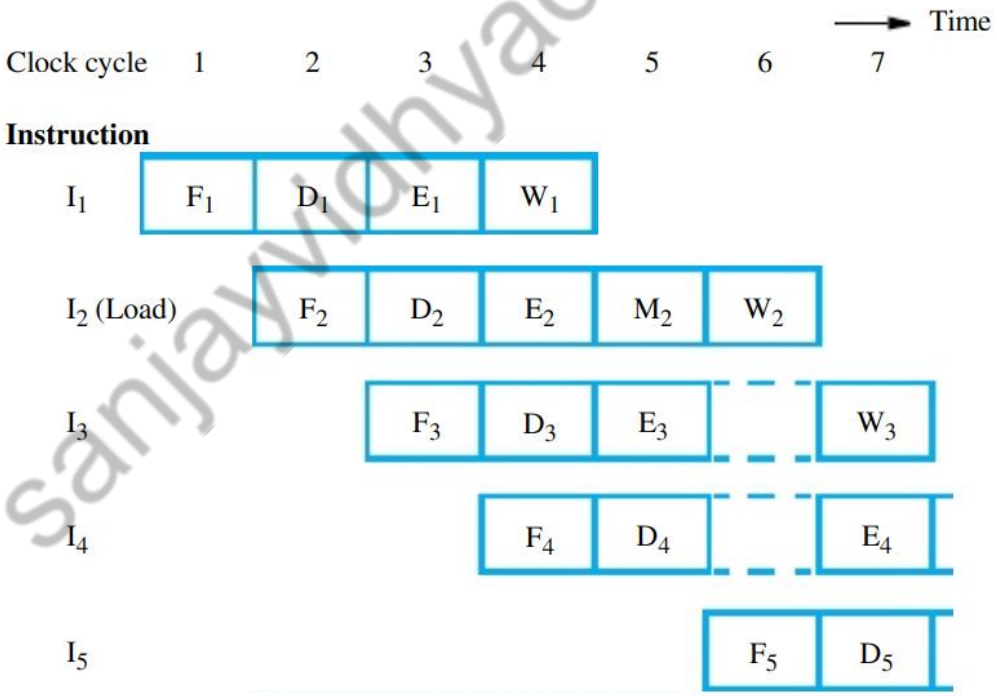
In choosing the addressing modes to be implemented in a pipelined processor, we must consider the effect of each addressing mode on instruction flow in the pipeline:

- Side effects
- The extent to which complex addressing modes cause the pipeline to stall
- Whether a given mode is likely to be used by compilers

Addressing Modes

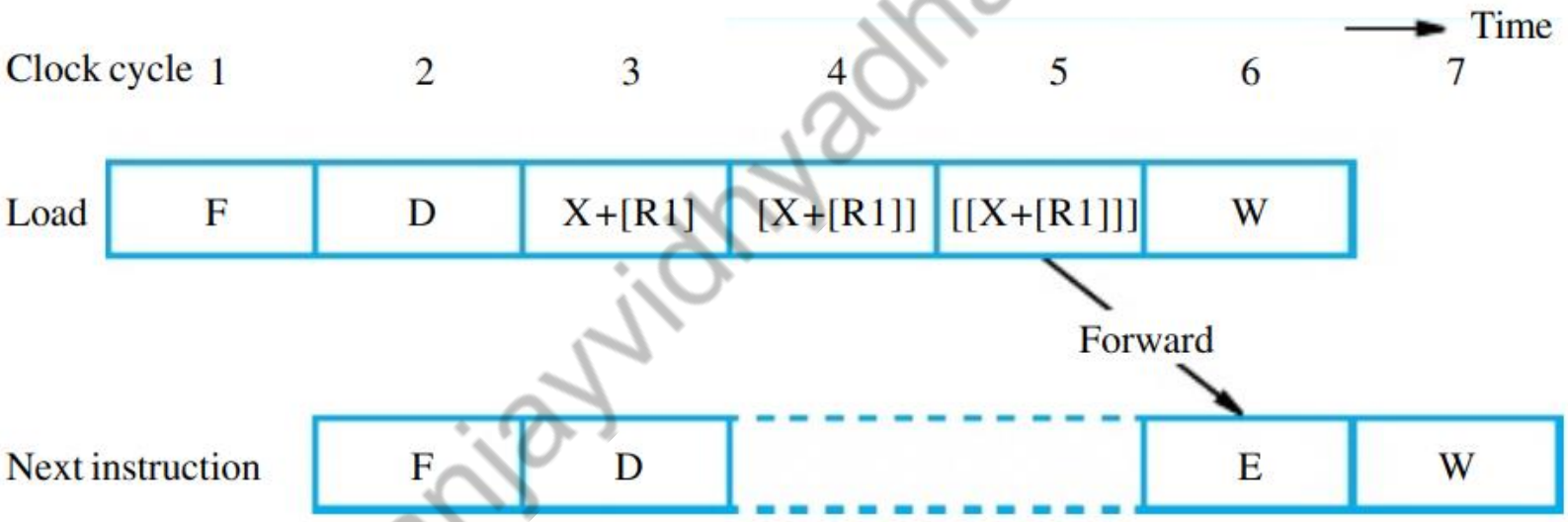
- Load (R1),R2 : No address computation hence can be done in single micro-cycle

- Load X(R1),R2 : Read the displacement X (X Available in the Instruction)
  Add X and (R1) (Additional Clock: Total 5 Clocks)
  Load data to R2



Addressing Modes

- Load (X(R1)),R2 : Read the displacement X (X Available in the Instruction)
  Add X and (R1) (Additional Clock)
  Fetch data from X + (R1) (Additional Clock)
  Get data again from memory (Total 6 Clock Cycles)



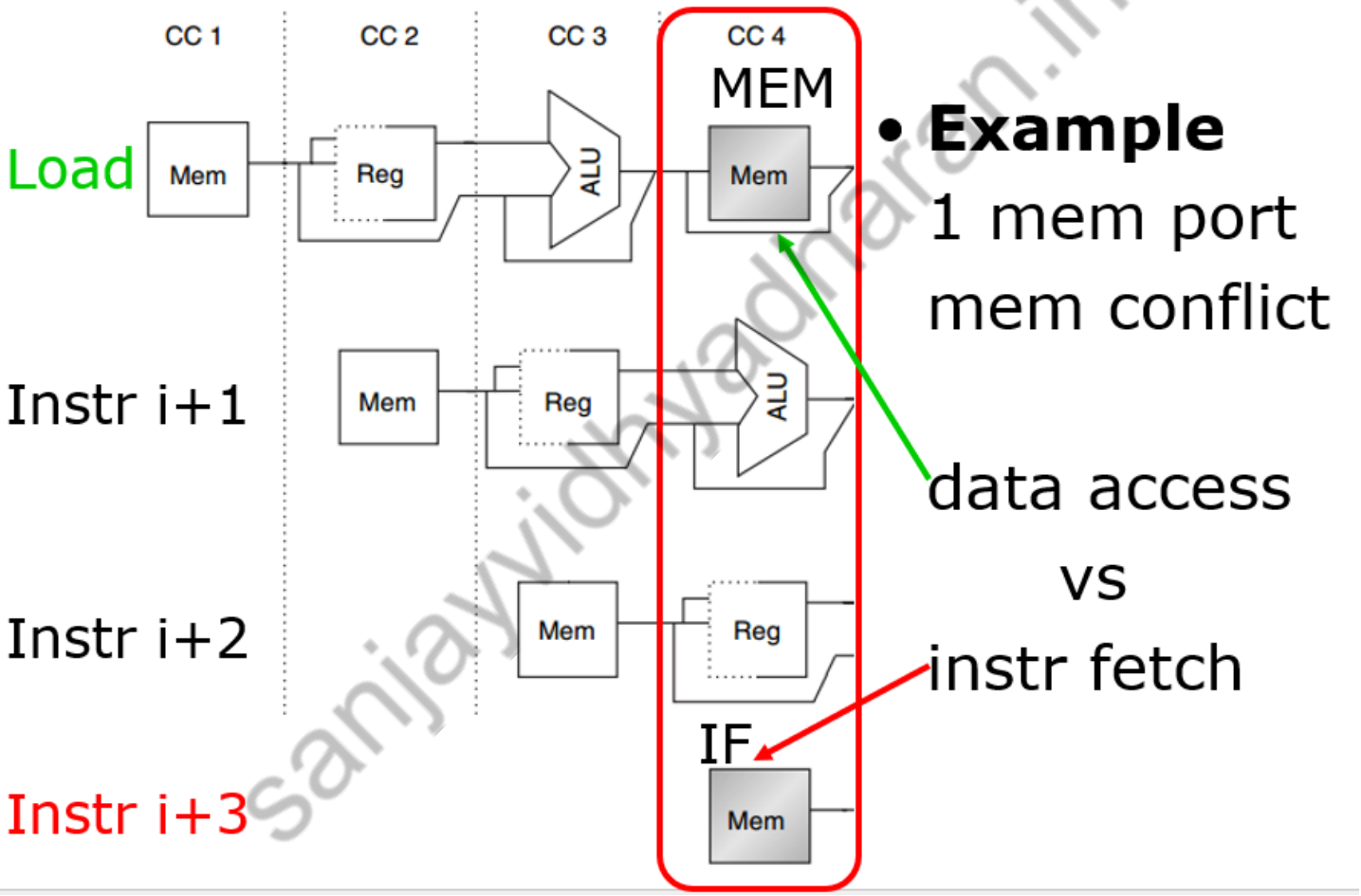
Addressing Modes

Complex addressing modes are not suitable for pipelined execution and should be avoided.

The addressing modes used in modern processors often have the following features:

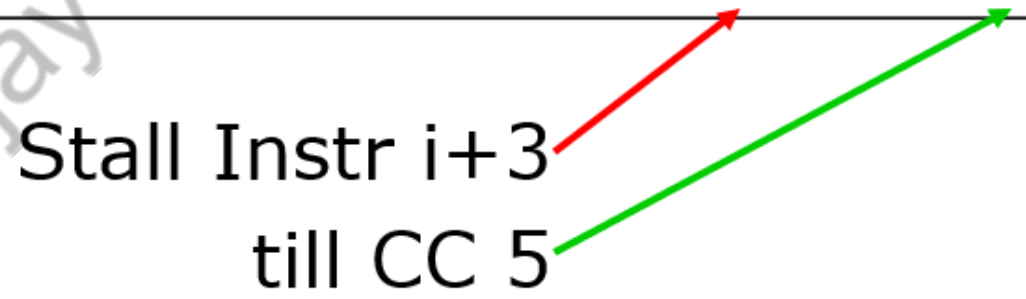
- Access to an operand does not require more than one access to the memory.
- Only load and store instructions access memory operands.
- The addressing modes used do not have side effects

Structural Hazards



Structural Hazards

Instruction	1	2	3	4	5
Load instruction	IF	ID	EX	MEM	WB
Instruction $i + 1$		IF	ID	EX	MEM
Instruction $i + 2$			IF	ID	EX
Instruction $i + 3$				stall	IF



Thank you