CS223 : Computer Architecture & Organization

Lecture 26 [07.04.2022]

Pipeline Hazards

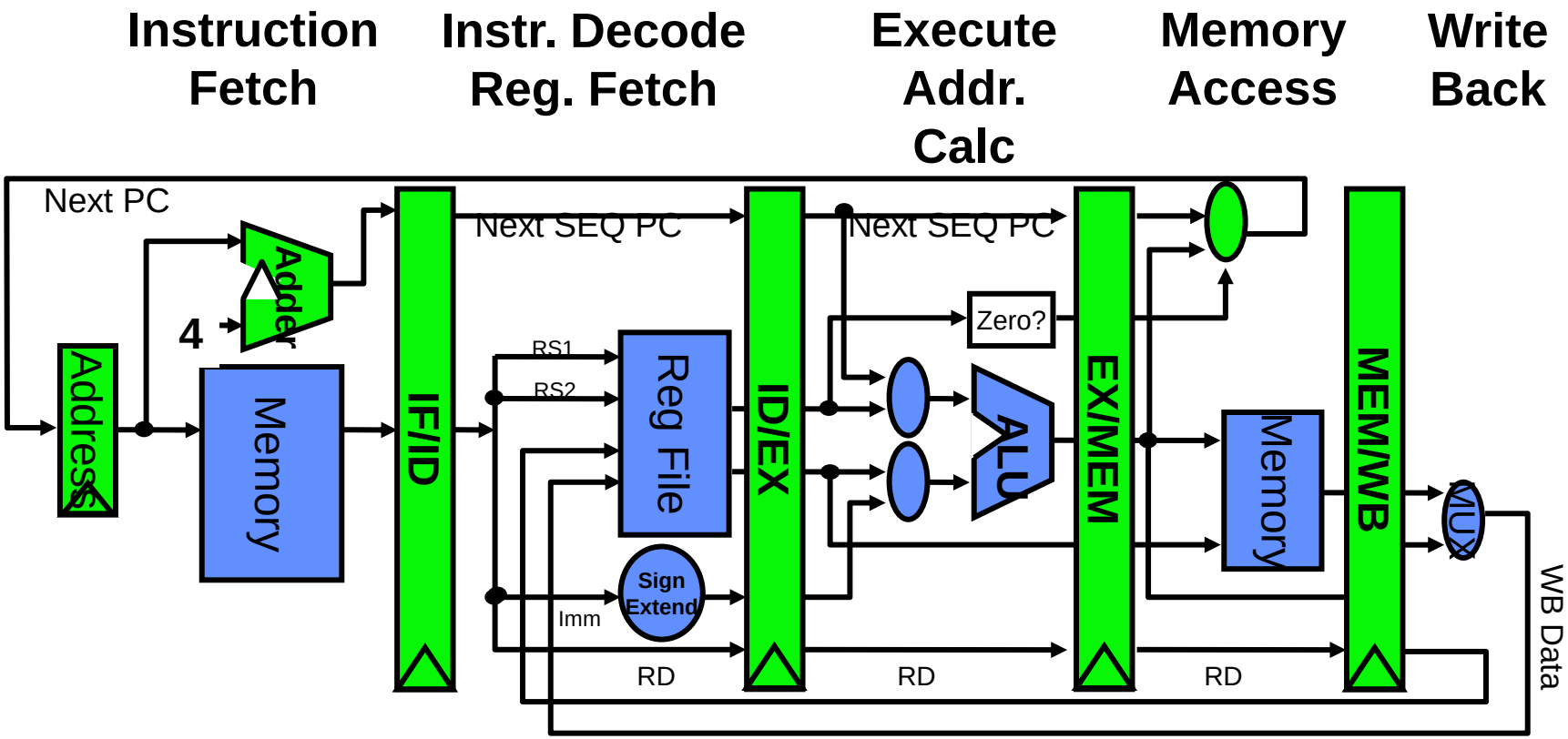
Dr. John Jose

Associate Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Pipelined RISC Data path



Visualizing Pipelining

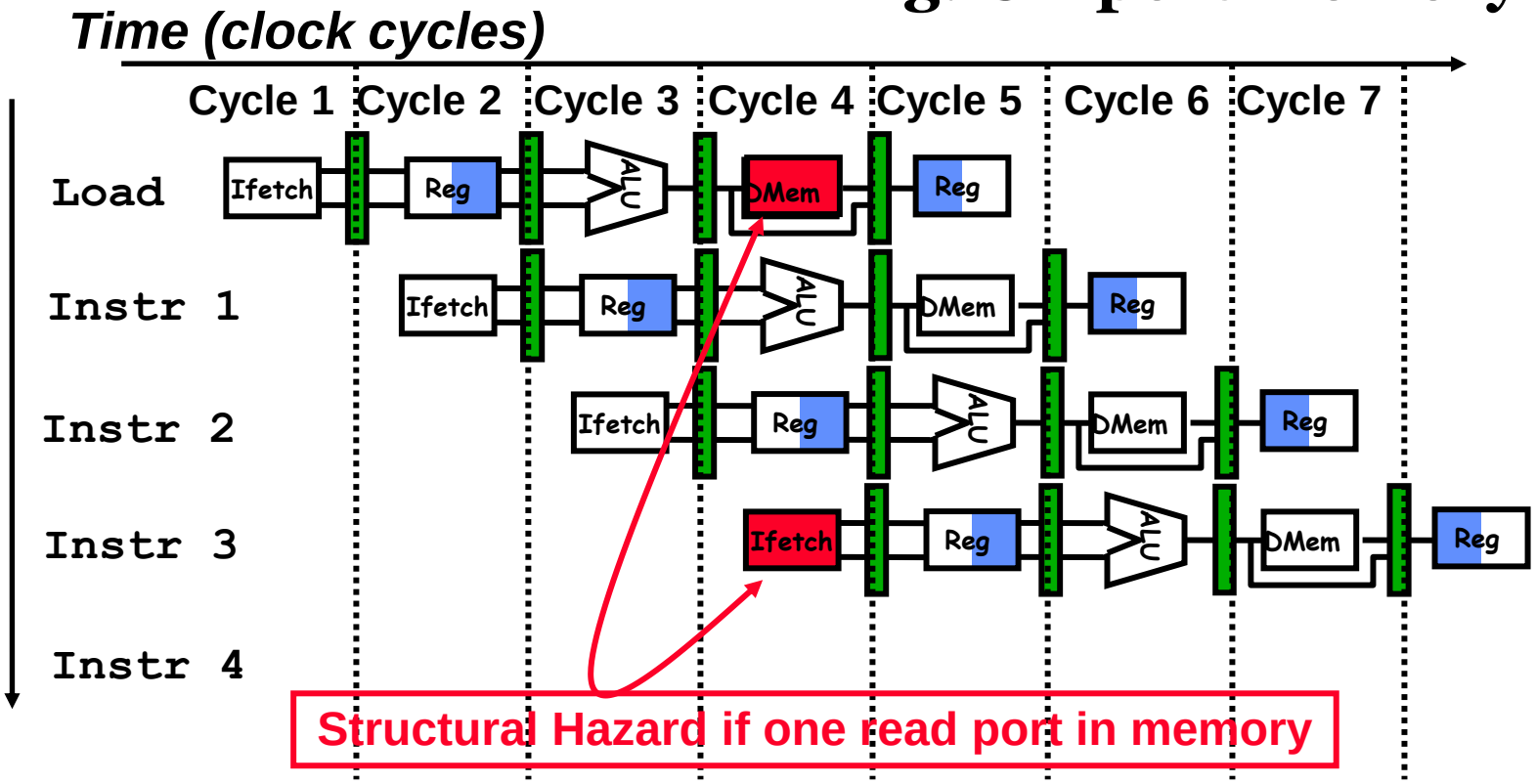
	Clock number							
Instruction number	1	2	3	4	5	6	7	8
i	IF	ID	EX	MEM	WB			
i+1		IF	ID	EX	MEM	WB		
i+2			IF	ID	EX	MEM	WB	
i+3				IF	ID	EX	MEM	WB
i+4					IF	ID	EX	MEM

Limits to pipelining

- **Hazards:** circumstances that would cause incorrect execution if next instruction is fetched and executed
  - **Structural hazards:** Different instructions, at different stages, in the pipeline want to use the same hardware resource
  - **Data hazards:** An instruction in the pipeline requires data to be computed by a previous instruction still in the pipeline
  - **Control hazards:** Succeeding instruction, to put into pipeline, depends on the outcome of a previous branch instruction, already in pipeline

Structural Hazard

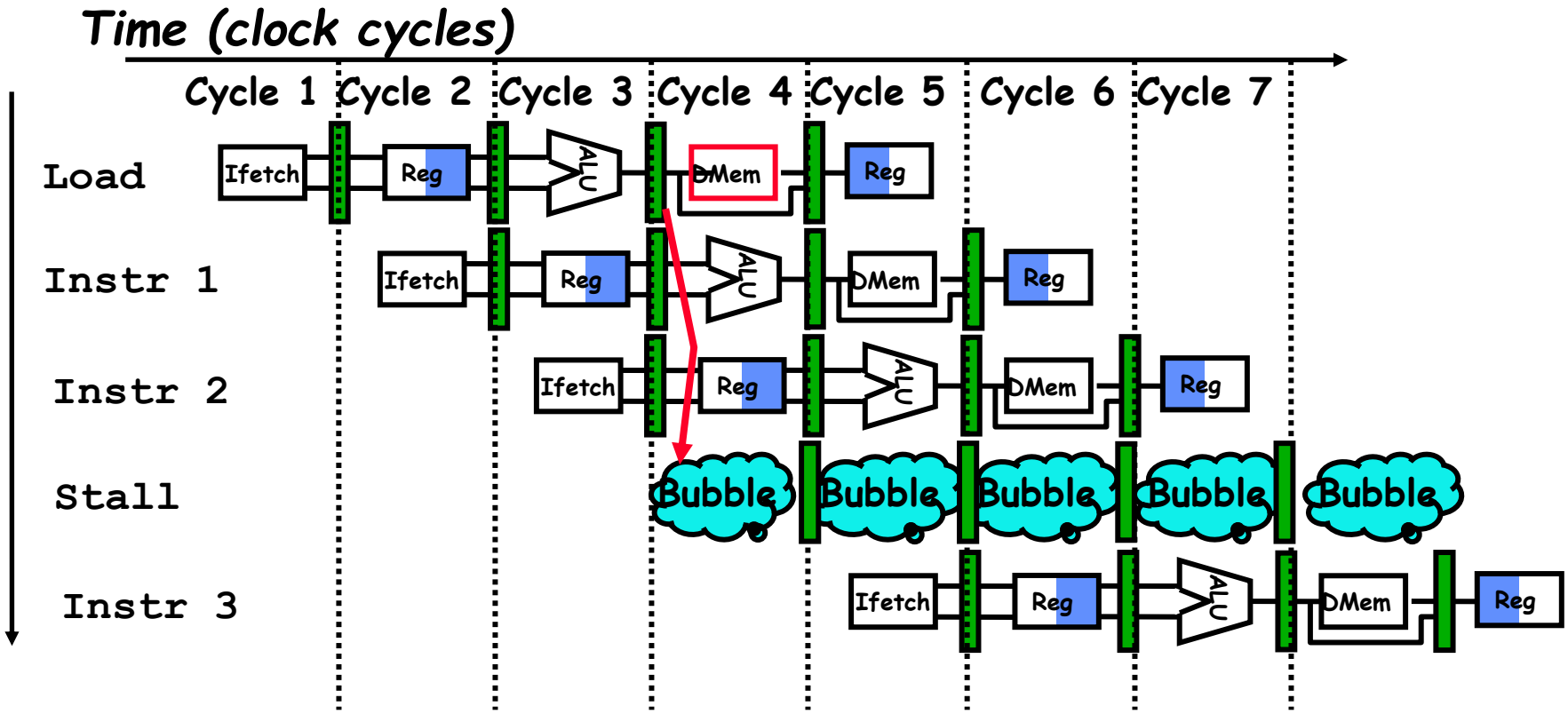
Eg: Uniport Memory



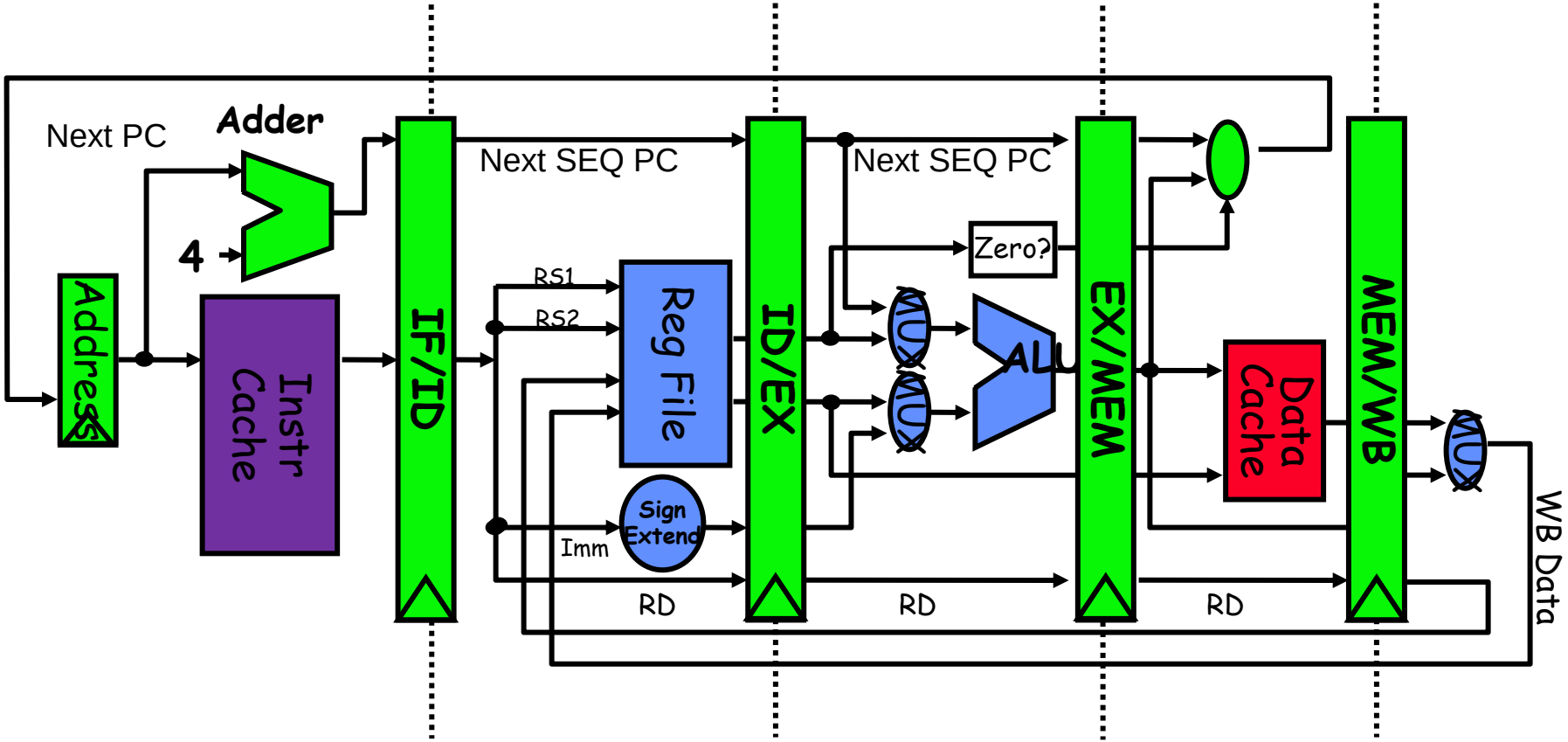
Resolving Structural Hazard

- Eliminate the use same hardware for two different things at the same time
- **Solution 1: Wait**
  - must detect the hazard
  - must have mechanism to stall
- **Solution 2: Duplicate hardware**
  - Multiple such units will help both instruction to progress

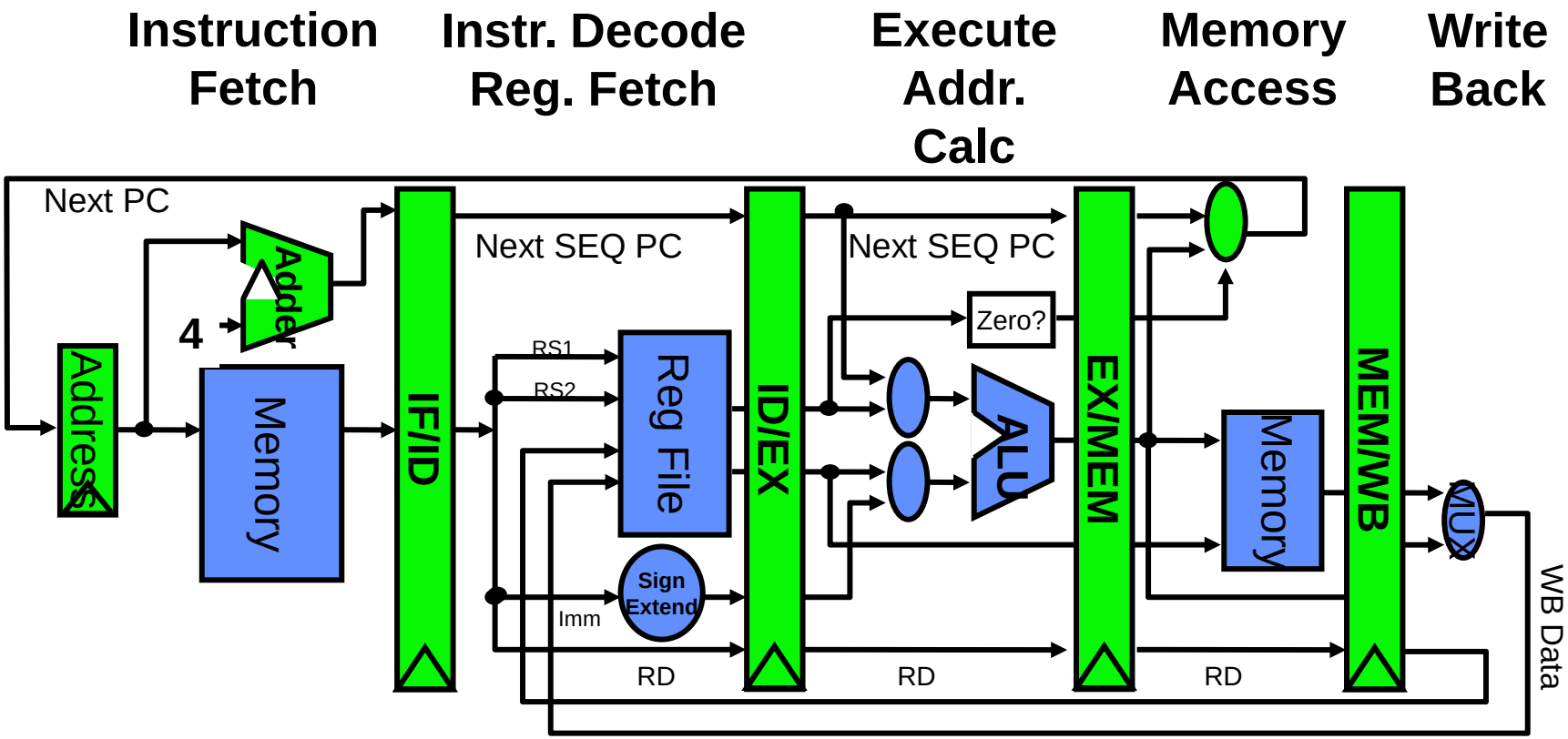
Detecting & Resolving Structural Hazard



Eliminating Structural Hazards at Design

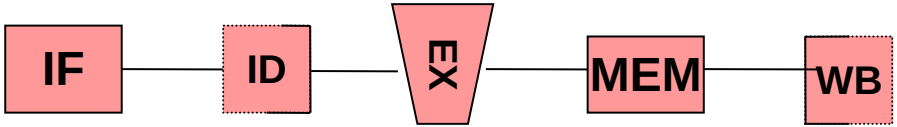


Pipelined RISC Data path



5 Steps of RISC Data path

- Each instruction can take at most 5 clock cycles
- **Instruction fetch cycle (IF)**
- **Instruction decode/register fetch cycle (ID)**
- **Execution/Effective address cycle (EX)**
- **Memory access cycle (MEM)**
- **Write-back cycle (WB)**



Visualizing Pipelining

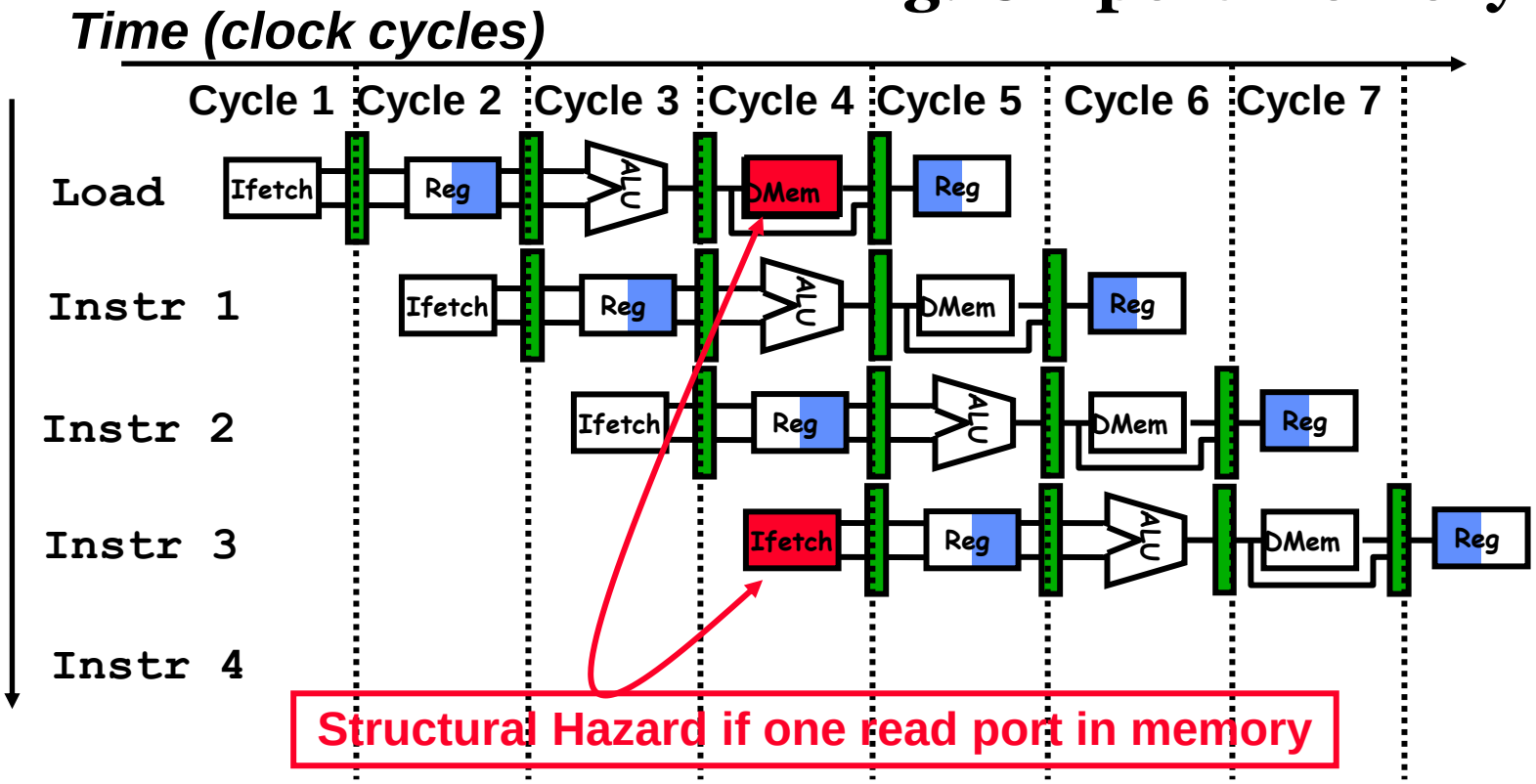
	Clock number							
Instruction number	1	2	3	4	5	6	7	8
i	IF	ID	EX	MEM	WB			
i+1		IF	ID	EX	MEM	WB		
i+2			IF	ID	EX	MEM	WB	
i+3				IF	ID	EX	MEM	WB
i+4					IF	ID	EX	MEM

Limits to pipelining

- **Hazards:** circumstances that would cause incorrect execution if next instruction is fetched and executed
  - **Structural hazards:** Different instructions, at different stages, in the pipeline want to use the same hardware resource
  - **Data hazards:** An instruction in the pipeline requires data to be computed by a previous instruction still in the pipeline
  - **Control hazards:** Succeeding instruction, to put into pipeline, depends on the outcome of a previous branch instruction, already in pipeline

Structural Hazard

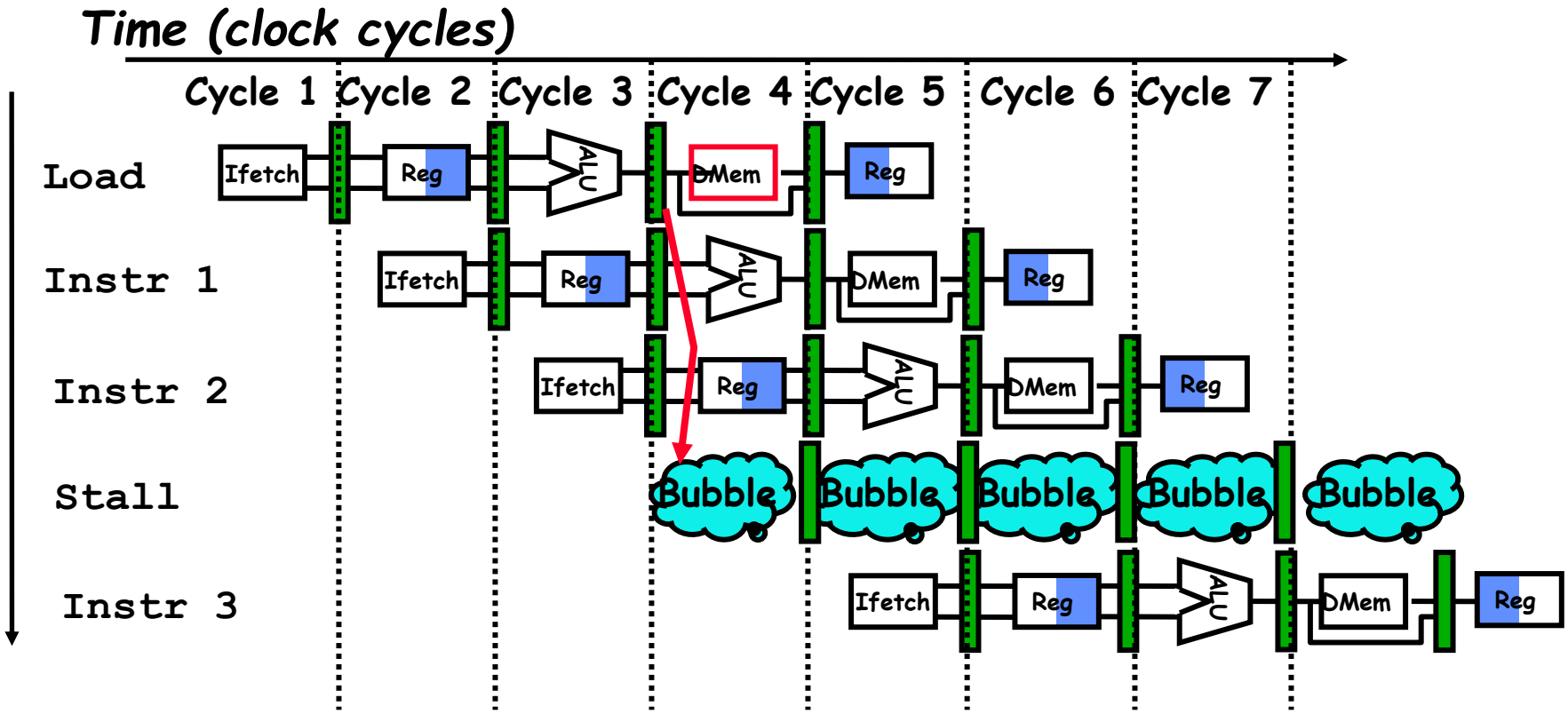
Eg: Uniport Memory



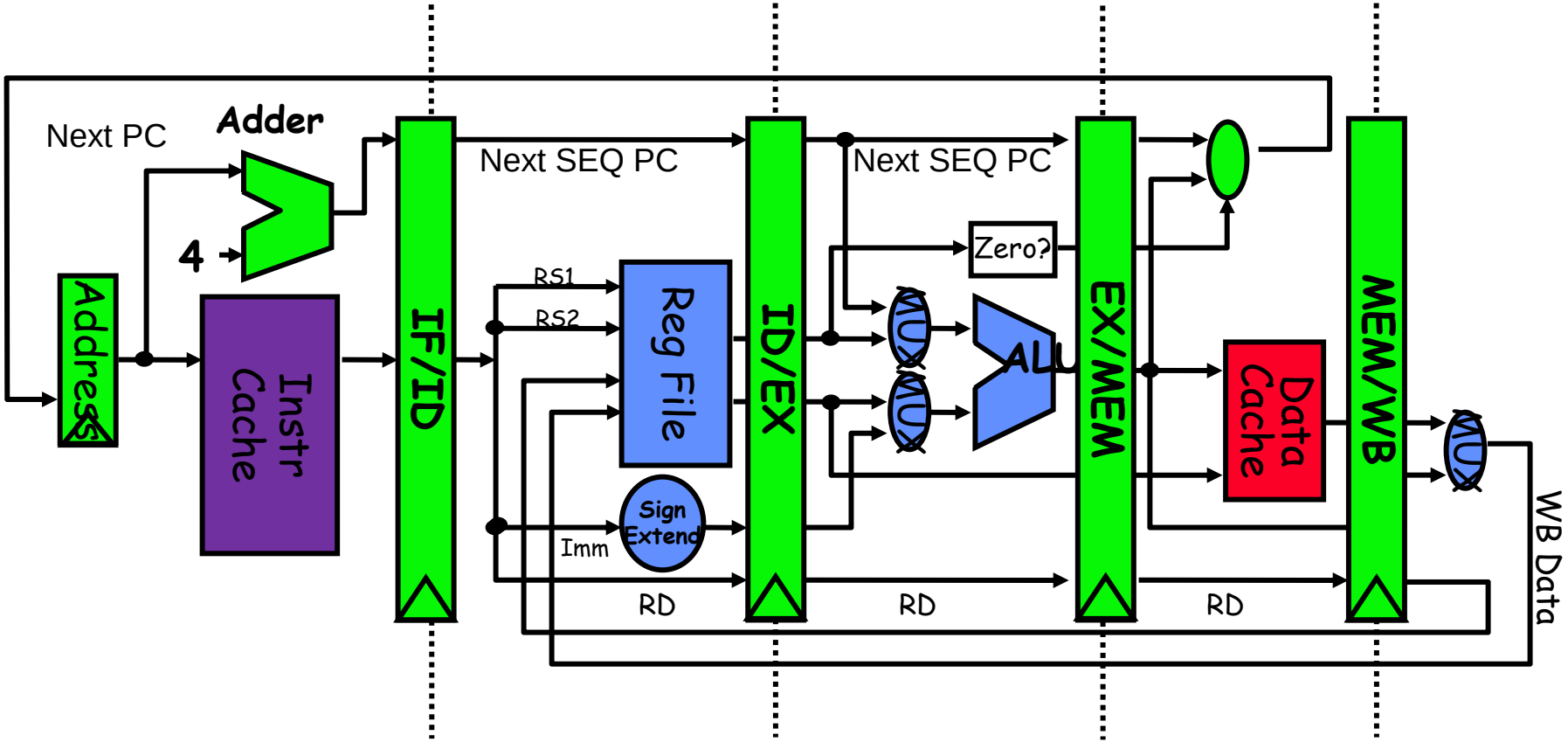
Resolving Structural Hazard

- Eliminate the use same hardware for two different things at the same time
- **Solution 1: Wait**
  - must detect the hazard
  - must have mechanism to stall
- **Solution 2: Duplicate hardware**
  - Multiple such units will help both instruction to progress

Detecting & Resolving Structural Hazard



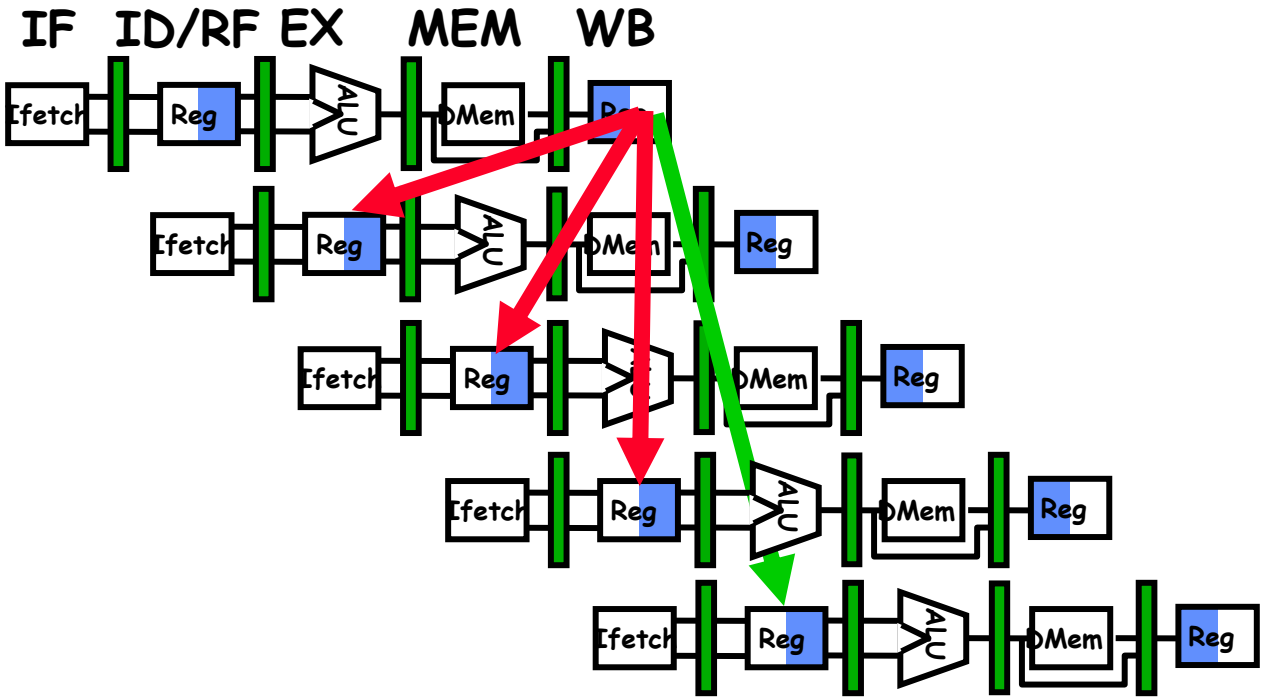
Eliminating Structural Hazards at Design



Data Hazard

Time (clock cycles)

```
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or   r8,r1,r9
xor r10,r1,r11
```



Three Generic Data Hazards

- **Read After Write (RAW)**
  $Instr_J$ tries to read operand before $Instr_I$ writes it

I : add r1,r2,r3

J: sub r4,r1,r3

- Caused by a data dependence
- This hazard results from an actual need for communication.

Three Generic Data Hazards

- **Write After Read (WAR)**
  $Instr_J$ writes operand before $Instr_I$ reads it

I: sub r4,r1,r3

J: add r1,r2,r3

K: mul r6,r1,r7

- Called an anti-dependence by compiler writers.
- This results from reuse of the name r1
- Can't happen in MIPS 5 stage pipeline because:
  - All instructions take 5 stages, and
  - Reads are always in stage 2, and
  - Writes are always in stage 5

Three Generic Data Hazards

- **Write After Write (WAW)**
  $Instr_J$ writes operand before $Instr_I$ writes it.

I: sub r1,r4,r3

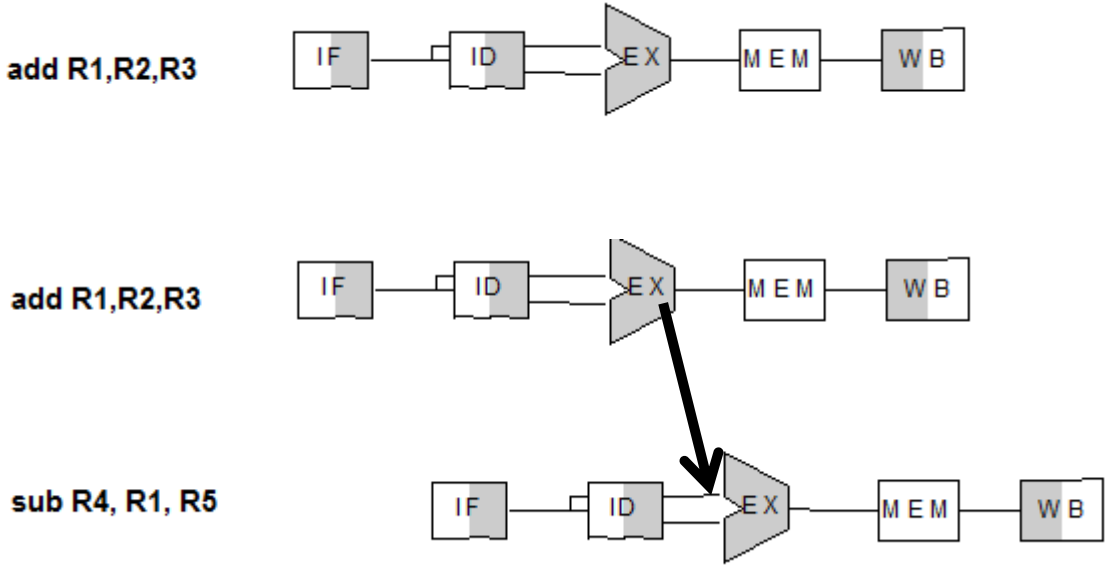
J: add r1,r2,r3

K: mul r6,r1,r7

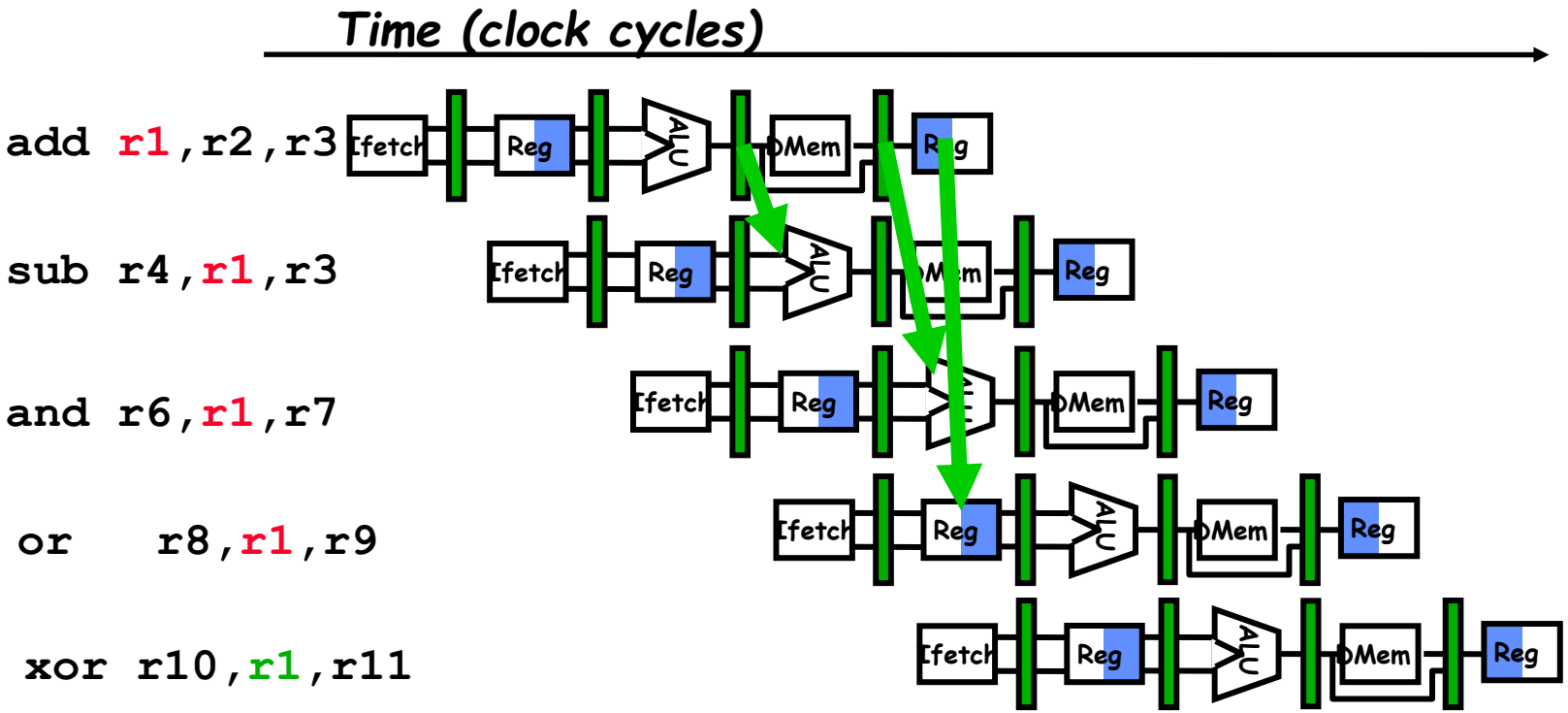
- Called an output dependence
- This also results from the reuse of name r1.
- Can't happen in MIPS 5 stage pipeline because:
  - All instructions take 5 stages, and
  - Writes are always in stage 5
- WAR and WAW happens in out of order pipes

How to Handle Data Hazard ?

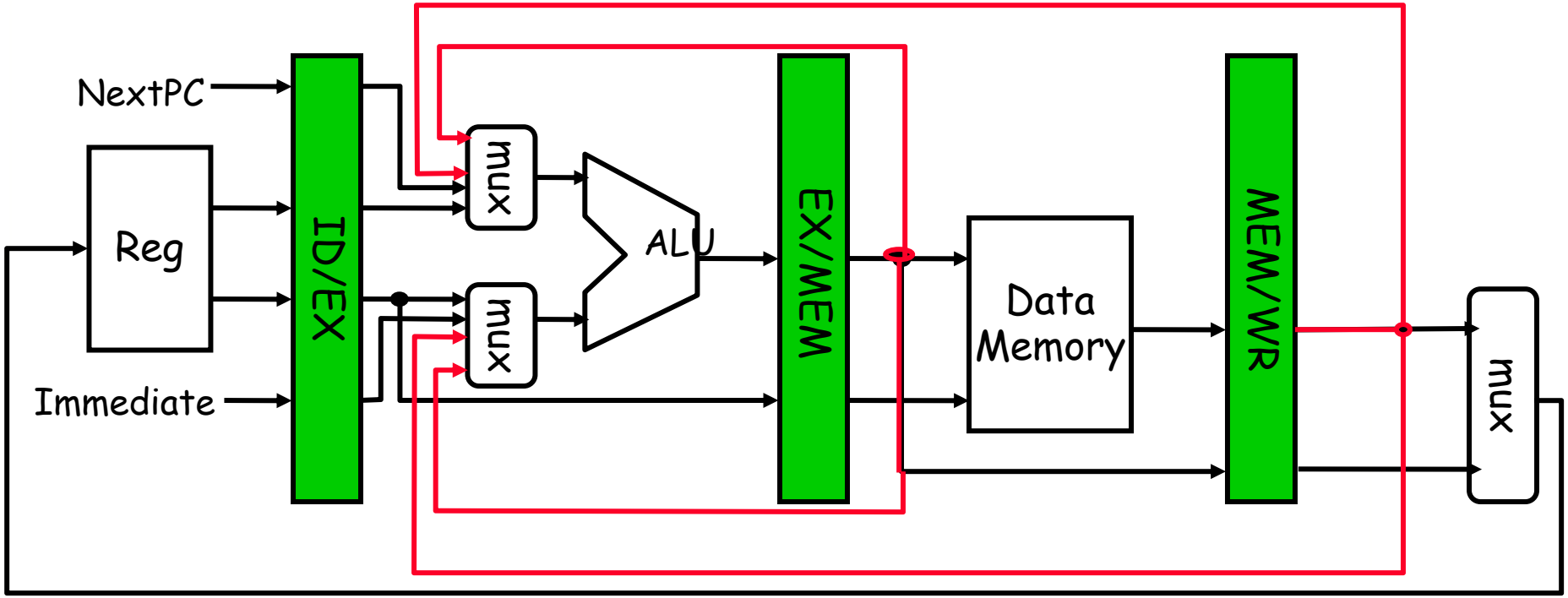
- Data hazard: instruction needs data from the result of a previous instruction still executing in pipeline
- **Solution:** Forward data if possible.



Operand Forwarding to Avoid Data Hazard



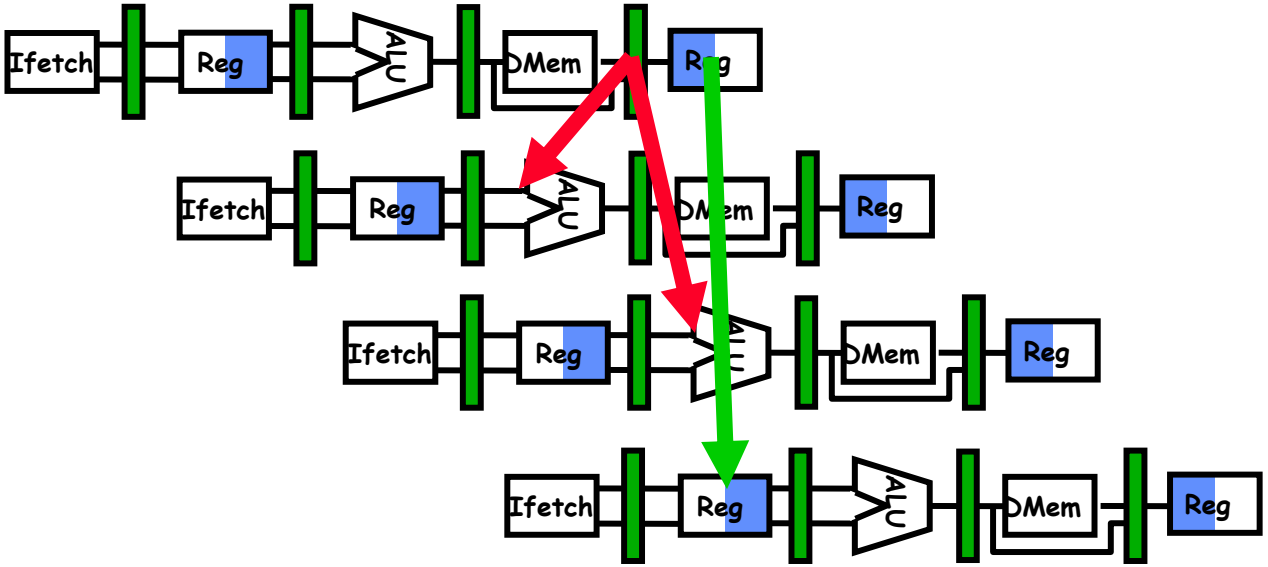
Hardware Change for Forwarding



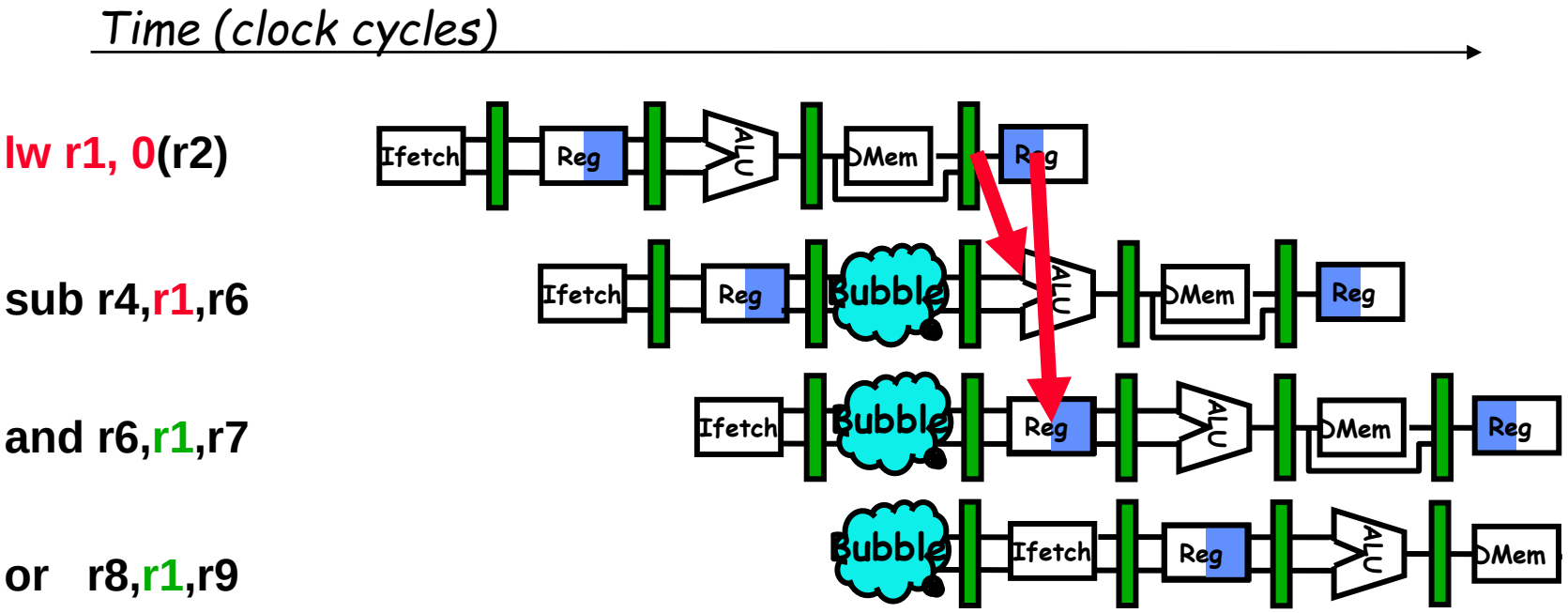
Data Hazard even with Operand Forwarding

Time (clock cycles)

```
lw r1, 0(r2)
sub r4,r1,r6
and r6,r1,r7
or   r8,r1,r9
```



Resolving the Load-ALU Hazard



Software Scheduling for Load Hazards

Assume a, b, c, d ,e, and f in memory.

a = b + c;

d = e – f;

```
LW    Rb,b
LW    Rc,c
ADD   Ra,Rb,Rc
SW    a,Ra
LW    Re,e
LW    Rf,f
SUB   Rd,Re,Rf
SW    d,Rd
```

```
LW    Rb,b
LW    Rc,c
LW    Re,e
ADD   Ra,Rb,Rc
LW    Rf,f
SW    a,Ra
SUB   Rd,Re,Rf
SW    d,Rd
```

Reference

- **Computer Architecture-A Quantitative Approach** (5th edition), John L. Hennessy, David A. Patterson, Morgan Kaufman.
- Appendix C: **Pipelining: Basic and Intermediate Concepts**
  - Section C2:
  - **The Major Hurdle of Pipelining—Pipeline Hazards**
  - **Structural hazards and data hazards**
- **NPTEL Video Link: https://tinyurl.com/yb4cbh2r**

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/