CS 223 Computer Organization & Architecture

Lecture 25 [06.04.2020]

Pipeline Hazards

John Jose

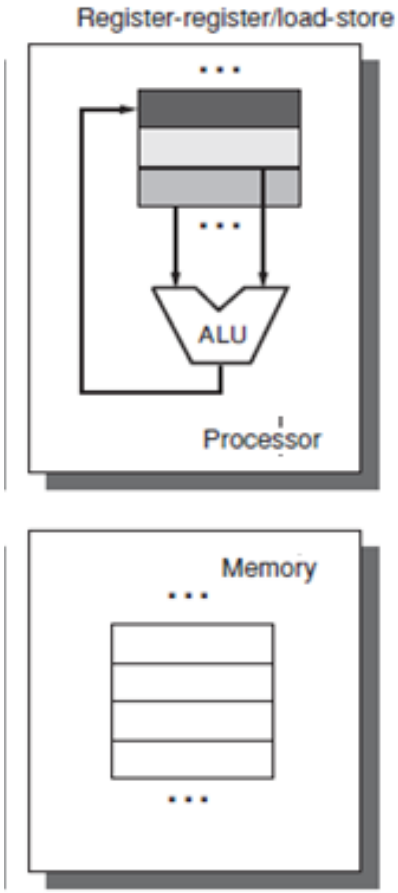
Assistant Professor

Department of Computer Science & Engineering

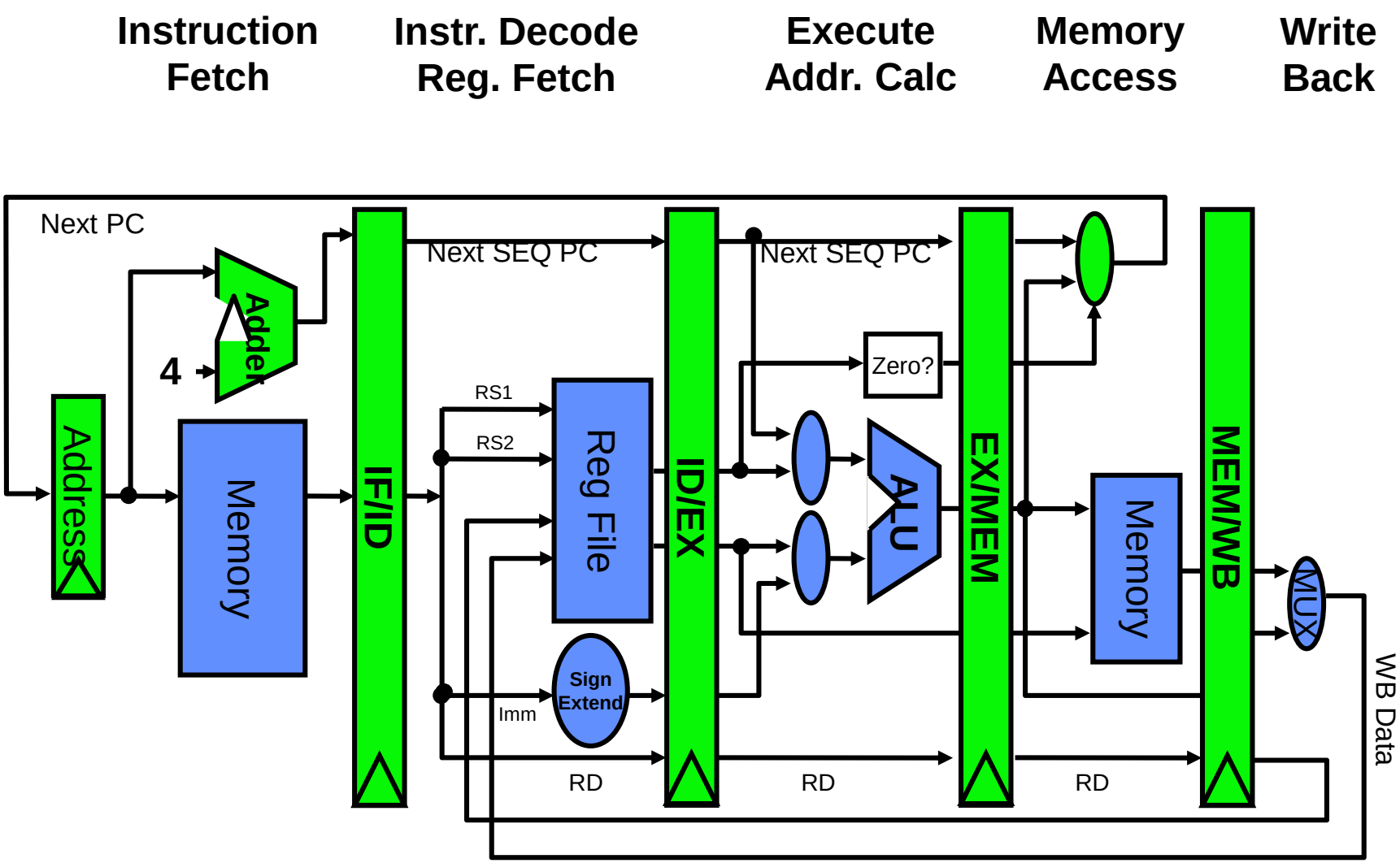
Indian Institute of Technology Guwahati, Assam.

Introduction to MIPS-RISC Architecture

- Microprocessor without Interlocked Pipelined Stages
- 32 registers (32 bit each)
- Uniform length instructions [4B]
- RISC- Load store architecture



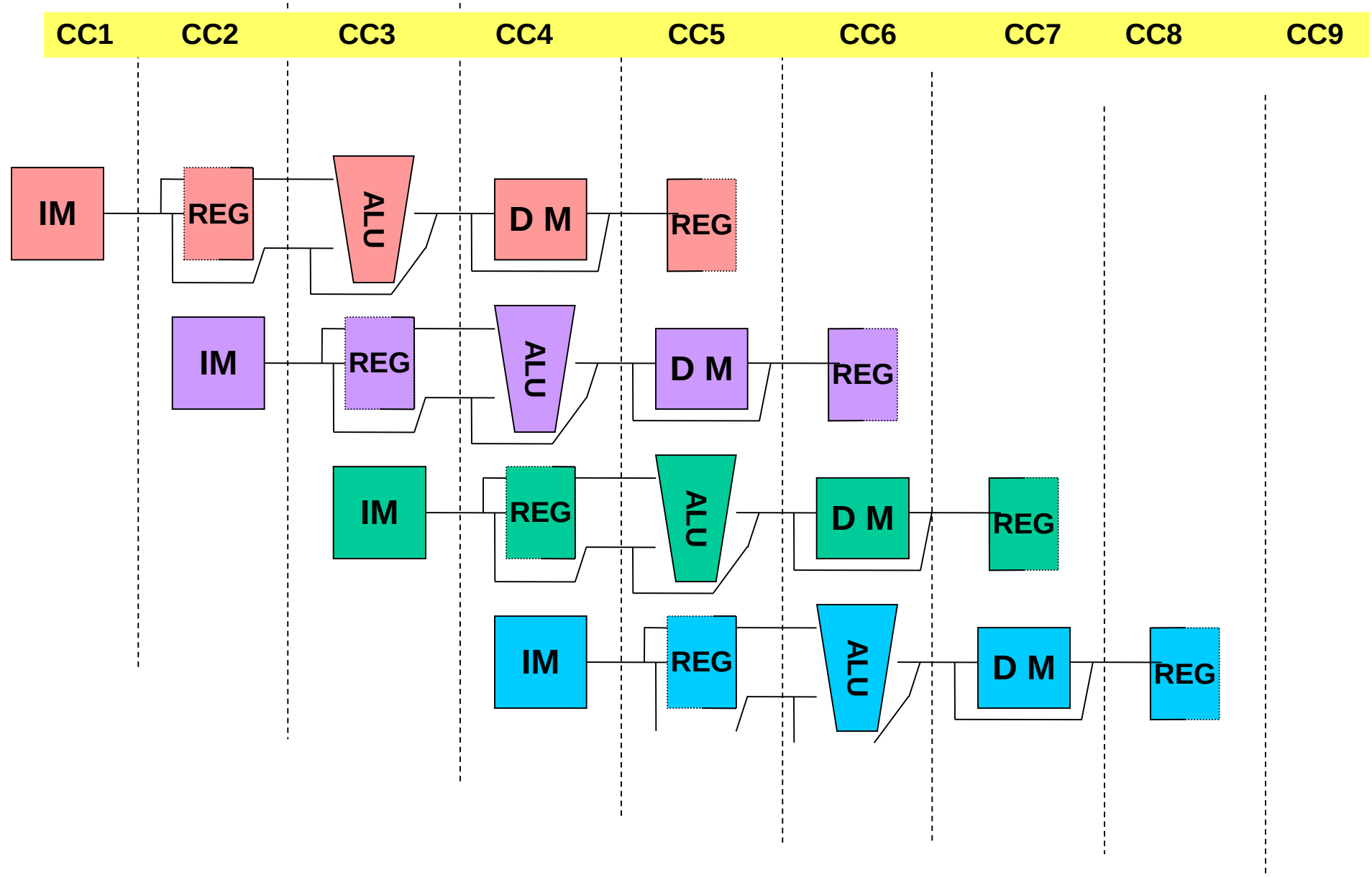
Pipelined RISC Data path



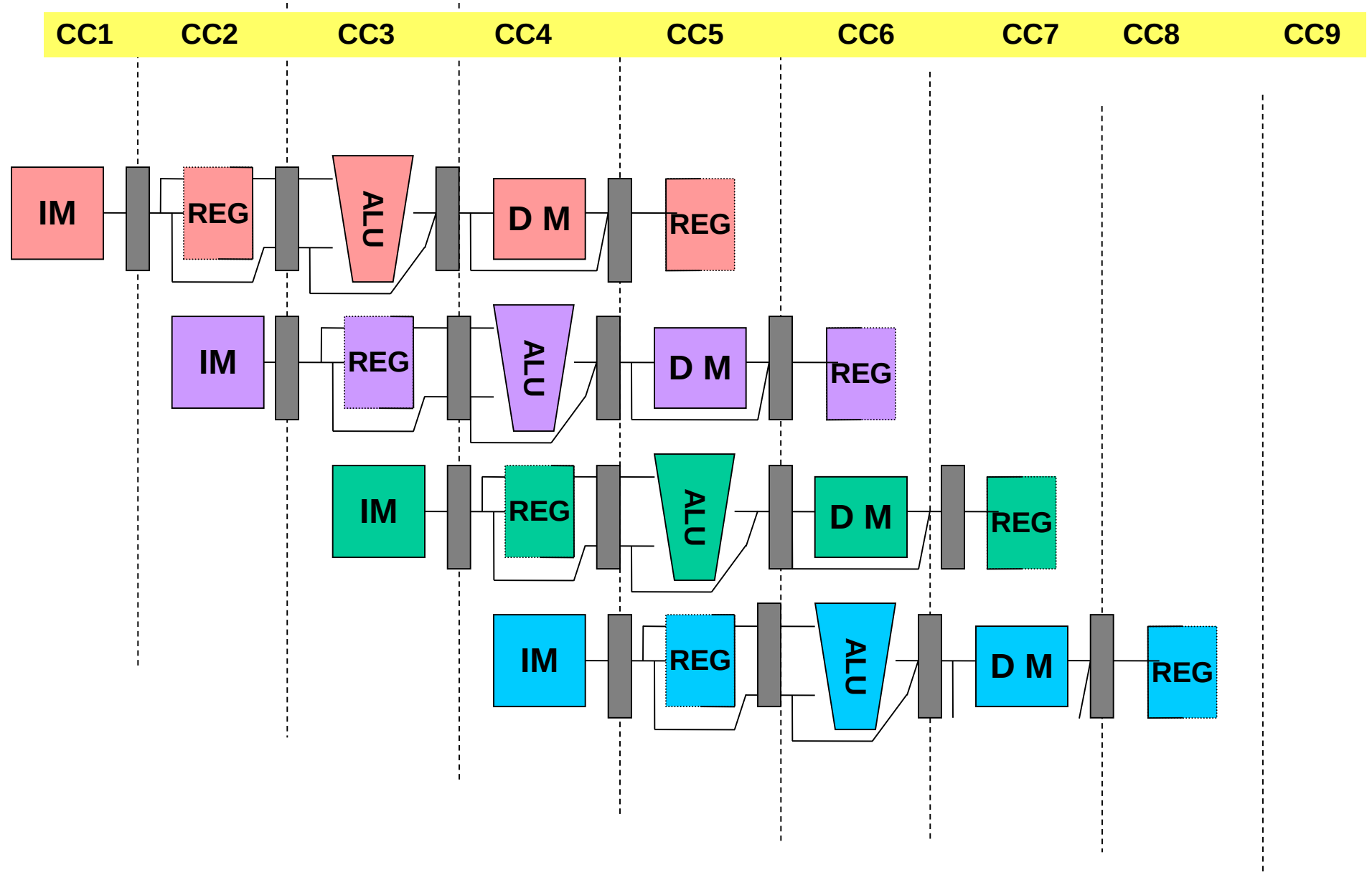
Visualizing Pipelining

	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

Visualizing Pipelining

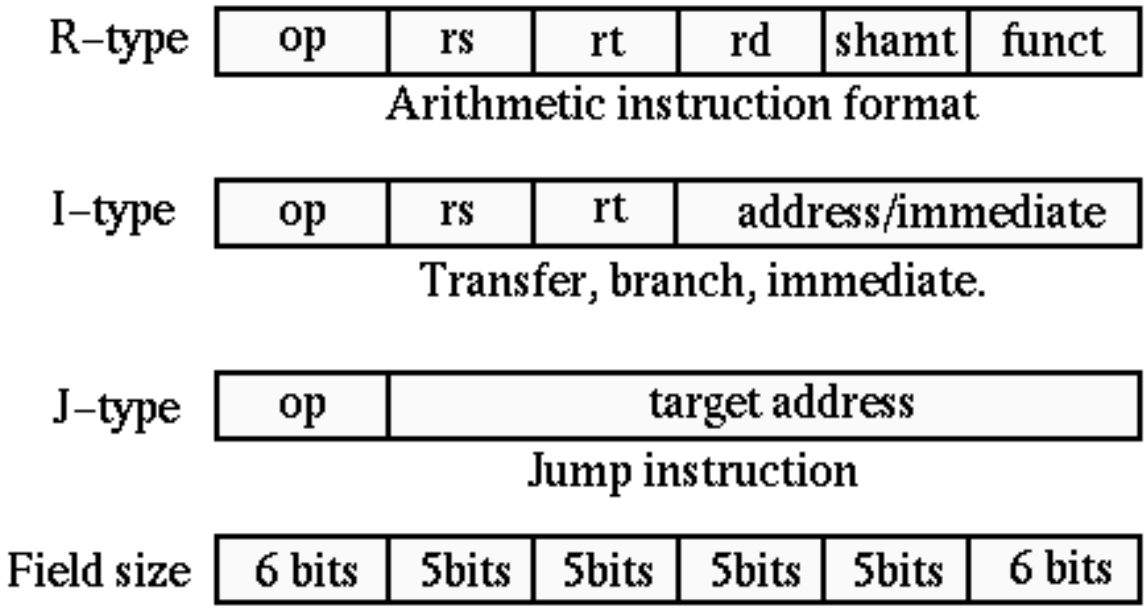


Visualizing Pipelining



Introduction to MIPS

- MIPS – Microprocessor without Interlocked Pipelined Stages
- 32 registers (32 bit each)
- Uniform length instructions
- RISC- Load store architecture



Visualizing Pipelining

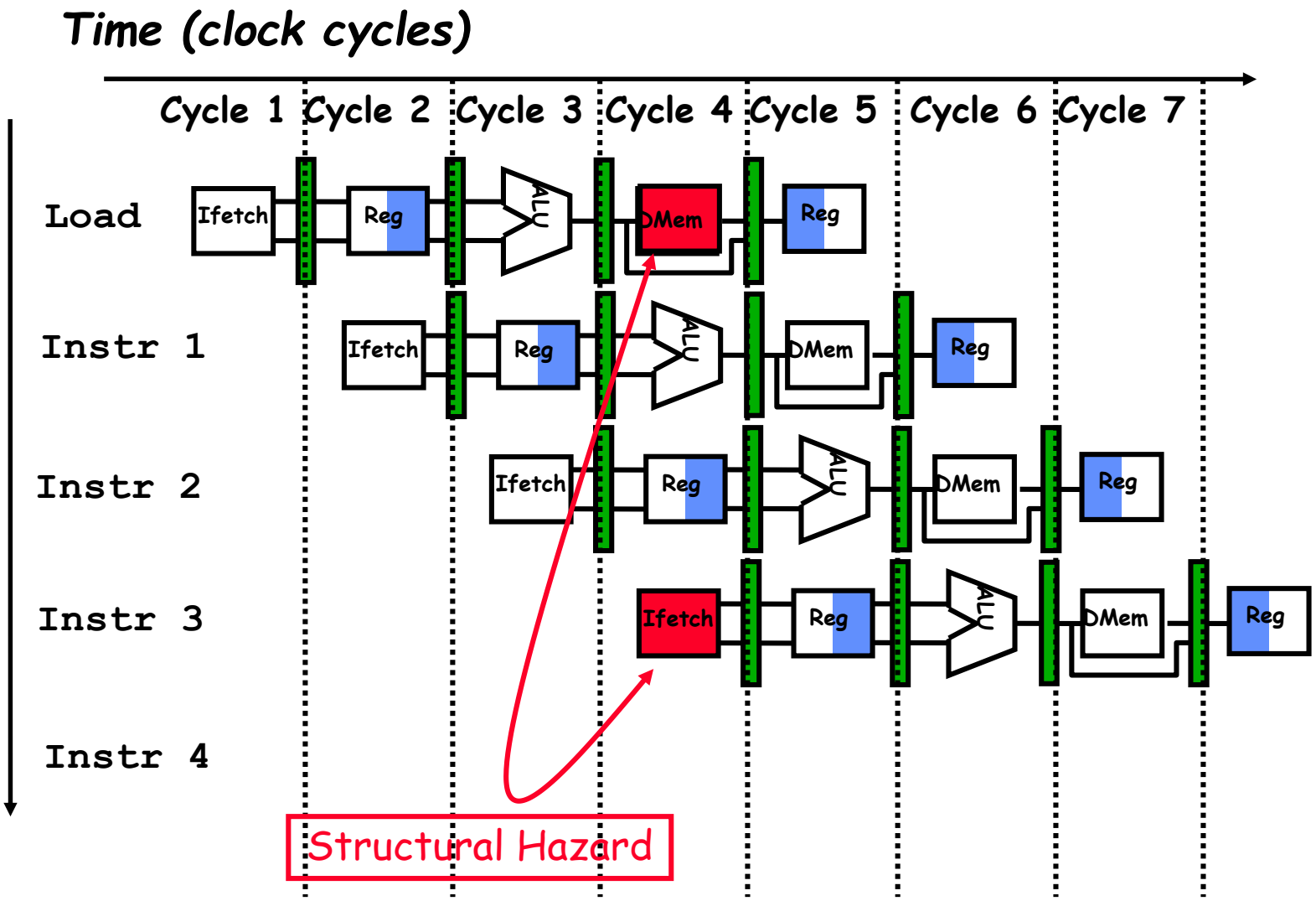
	Clock number							
Instruction number	1	2	3	4	5	6	7	8
i	IF	ID	EX	MEM	WB			
i+1		IF	ID	EX	MEM	WB		
i+2			IF	ID	EX	MEM	WB	
i+3				IF	ID	EX	MEM	WB
i+4					IF	ID	EX	MEM

Limits to pipelining

- **Hazards:** circumstances that would cause incorrect execution if next instruction is fetched and executed
  - **Structural hazards:** Attempting to use the same hardware to do two different things at the same time
  - **Data hazards:** Instruction depends on result of prior instruction still in the pipeline
  - **Control hazards:** Caused by delay between the fetching of instructions and decisions about changes in control flow (branches)

Structural Hazard

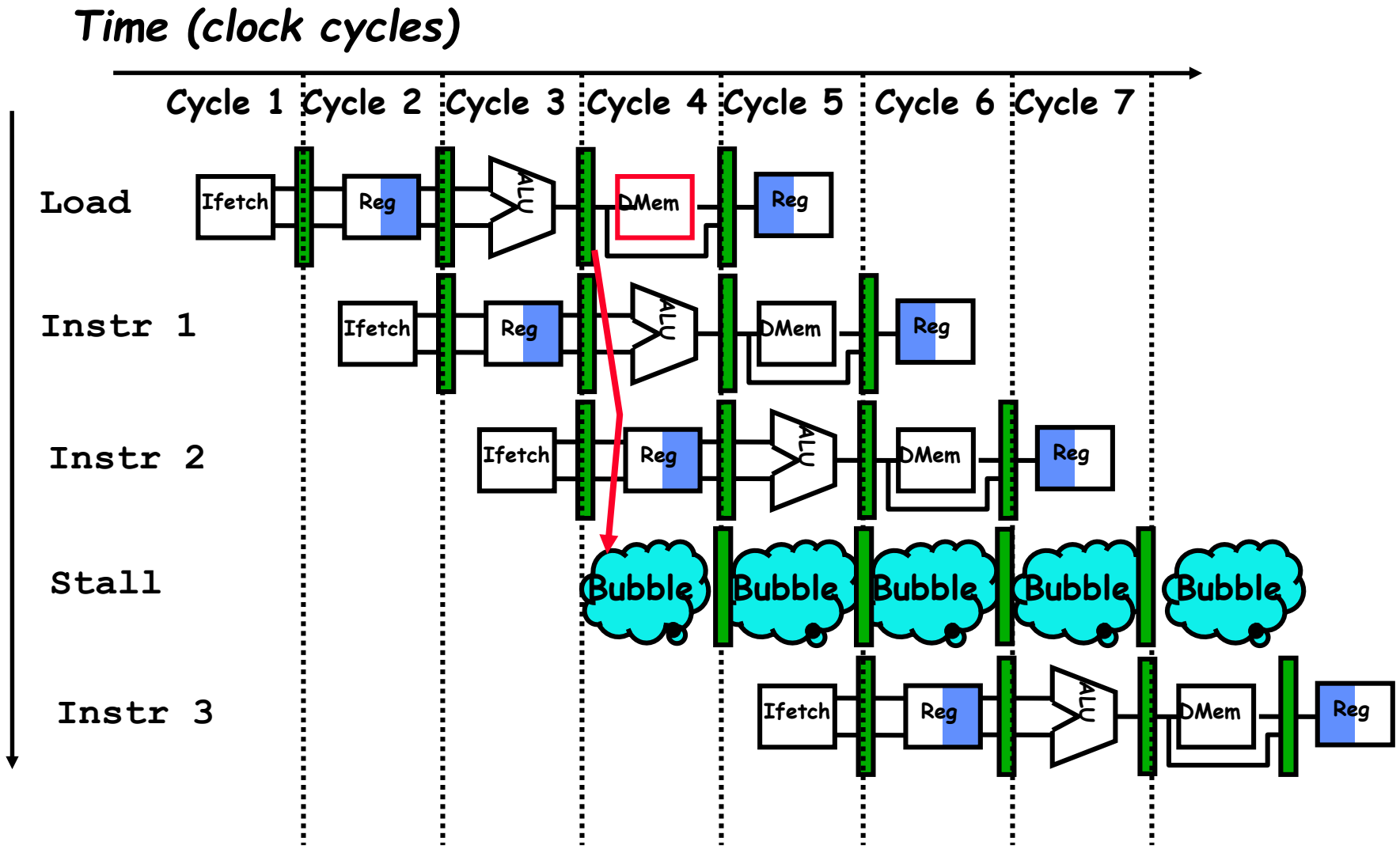
Eg: Uniport Memory



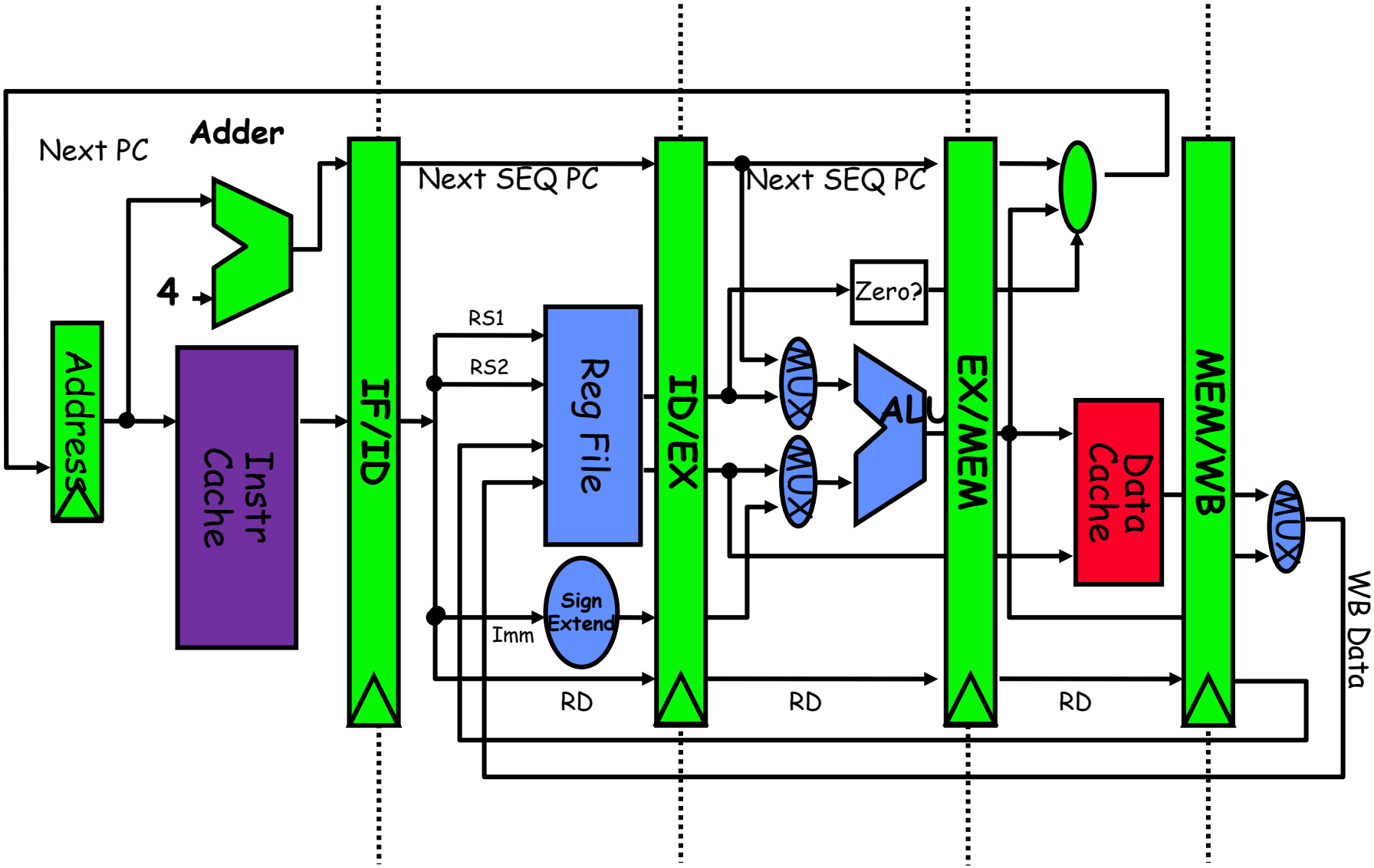
Resolving Structural Hazard

- 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

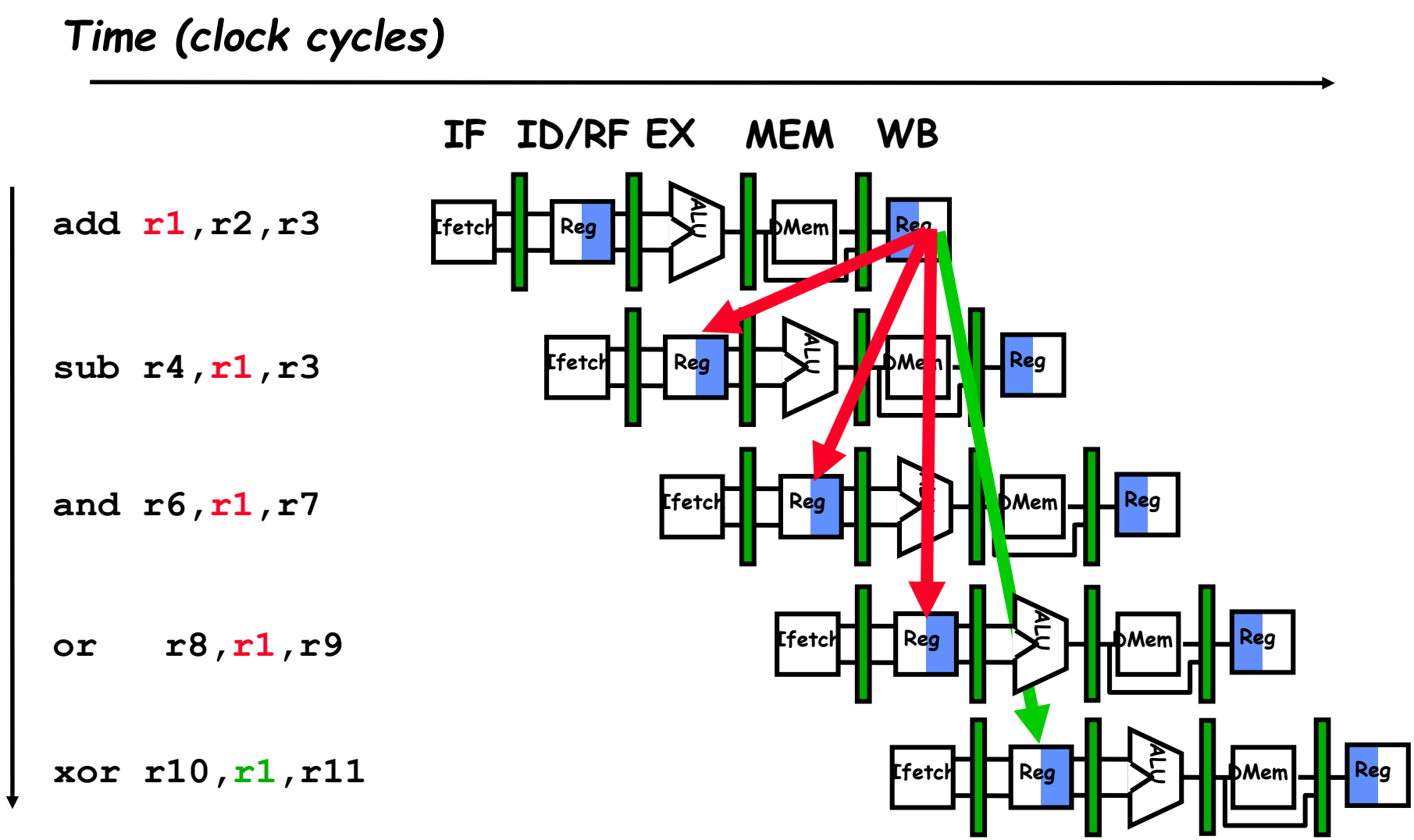
Detecting & Resolving Structural Hazard



Eliminating Structural Hazards at Design

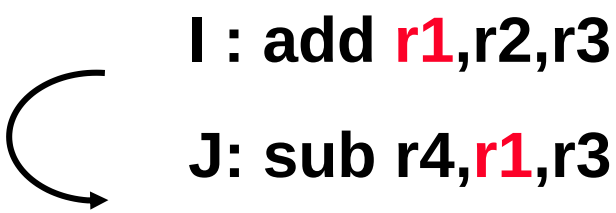


Data Hazard


Time (clock cycles)
IF ID/RF EX MEM WB
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
xor r10,r1,r11
Ifetch
Reg
ALU
DMem
Reg

Three Generic Data Hazards

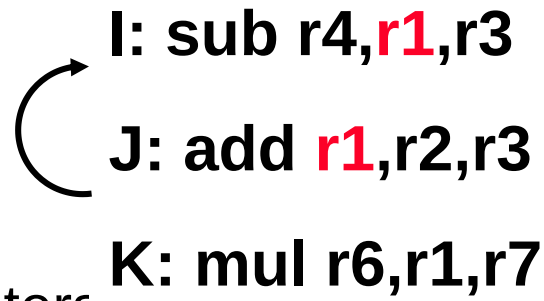
- **Read After Write (RAW)**
  $Instr_J$ tries to read operand before $Instr_I$ writes it



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

Three Generic Data Hazards

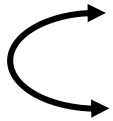
- **Write After Read (WAR)**
  Instr$_J$ writes operand before Instr$_I$ reads it



- 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.



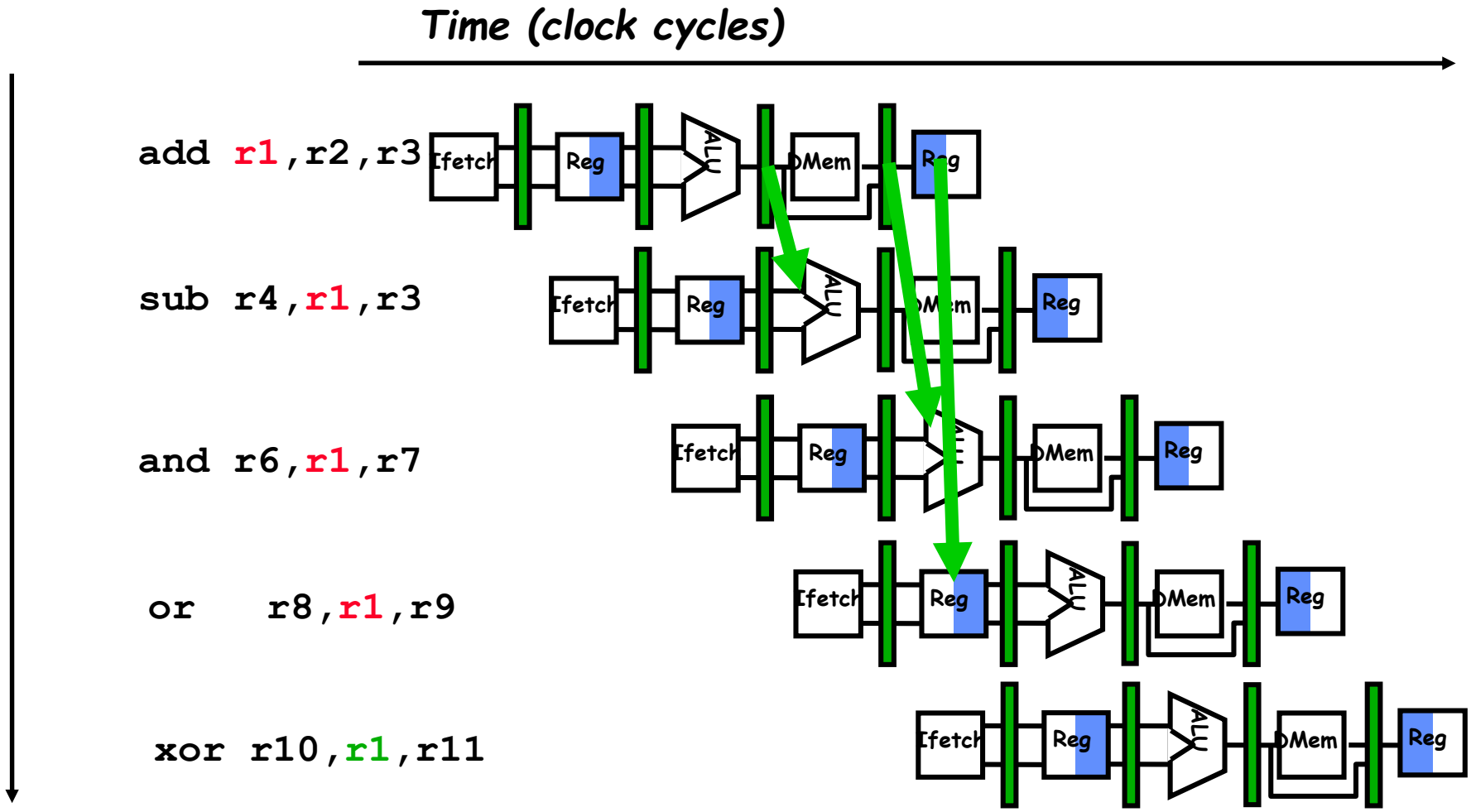
I: sub r1,r4,r3

J: add r1,r2,r3

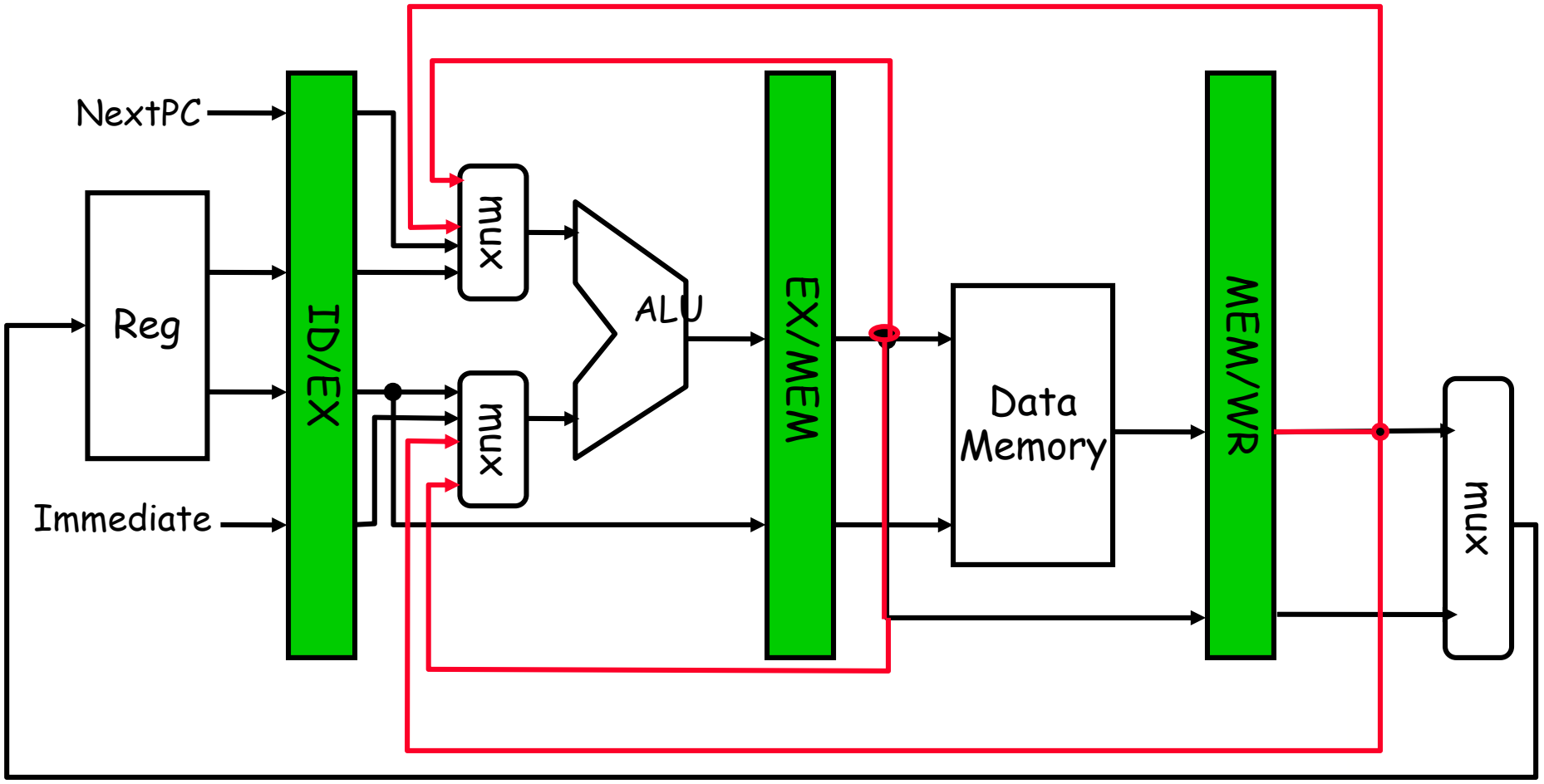
K: mul r6,r1,r7

- 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

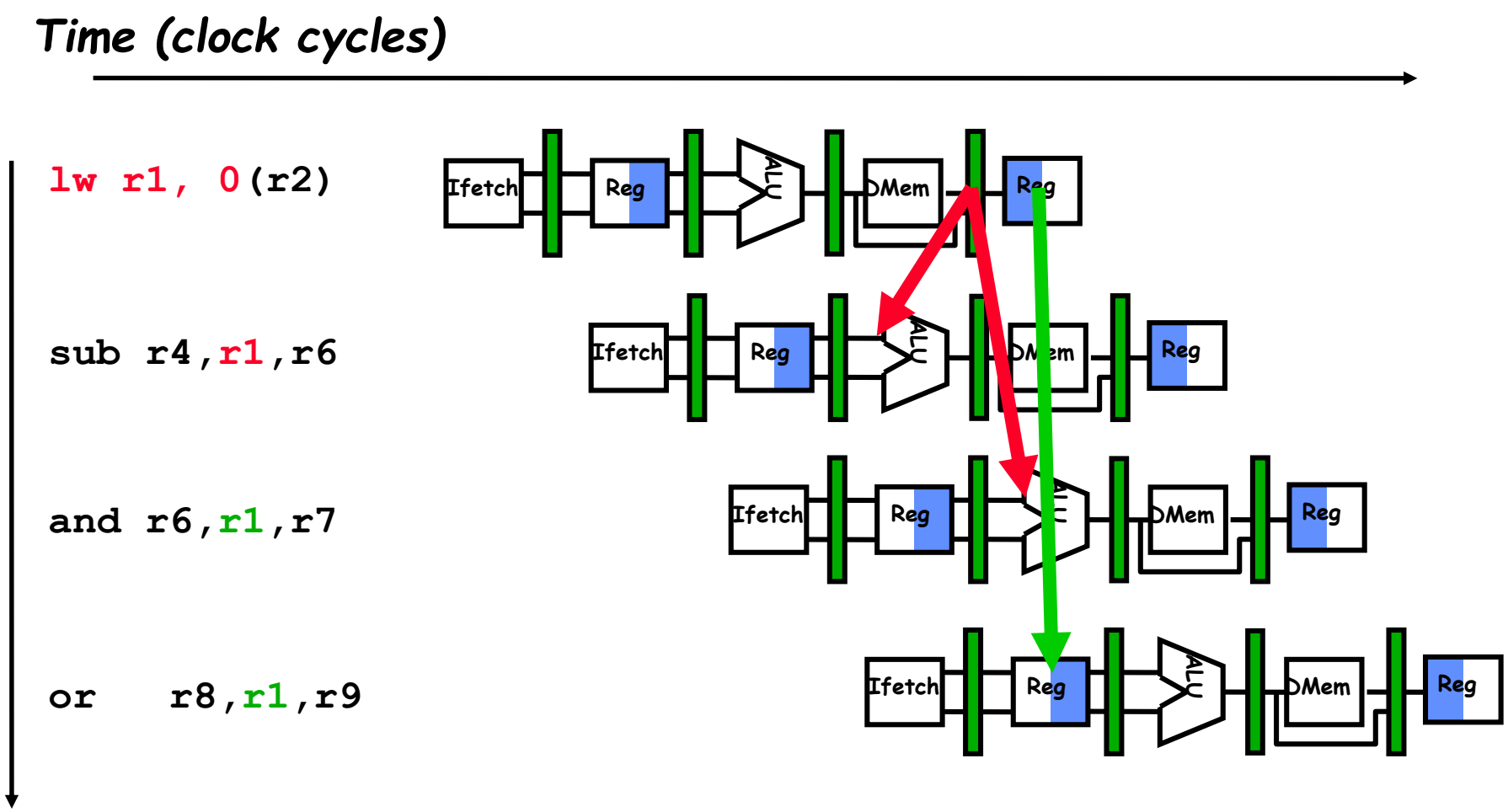
Operand Forwarding to Avoid Data Hazard



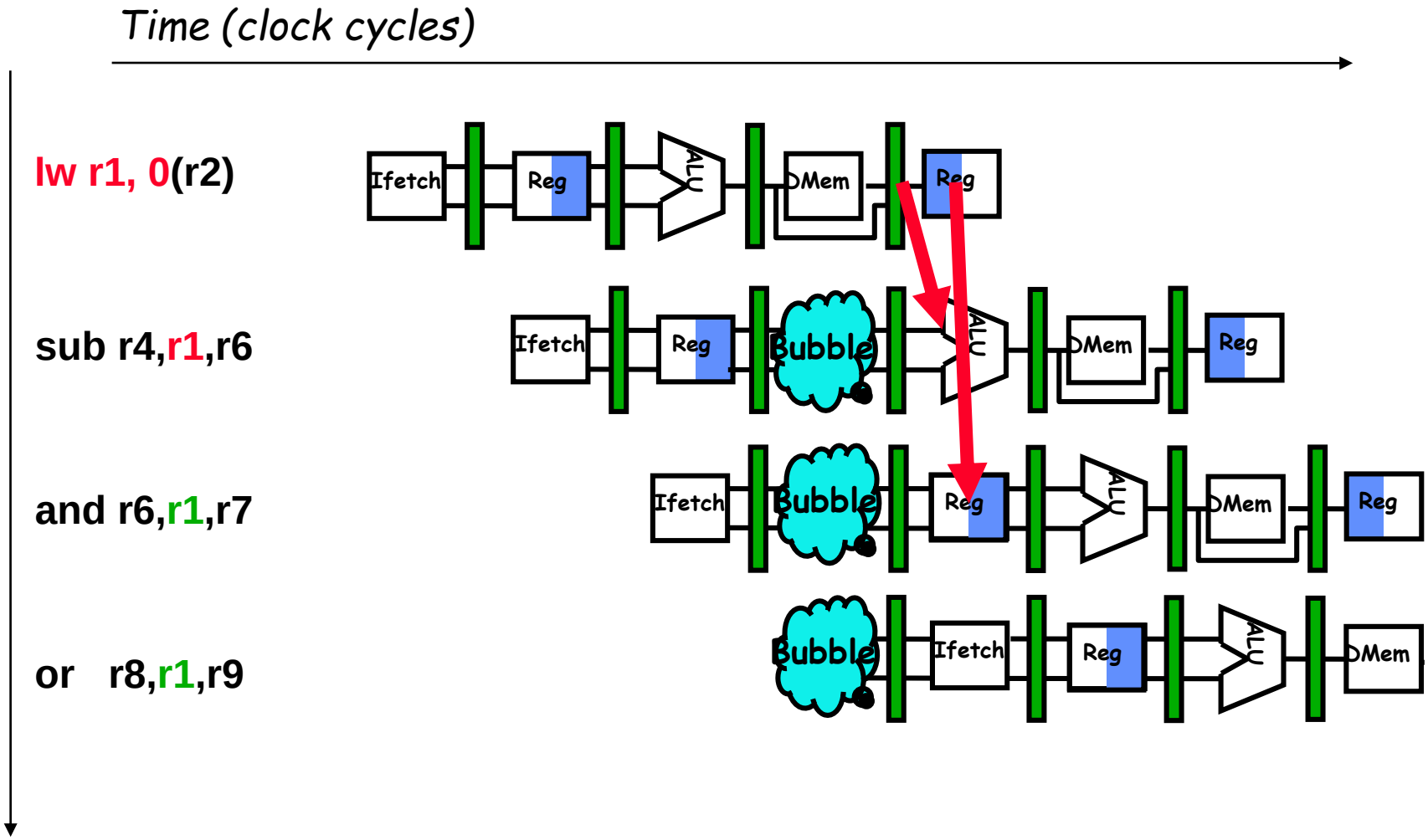
Hardware Change for Forwarding



Data Hazard even with Operand Forwarding



Resolving the Load-ALU Hazard



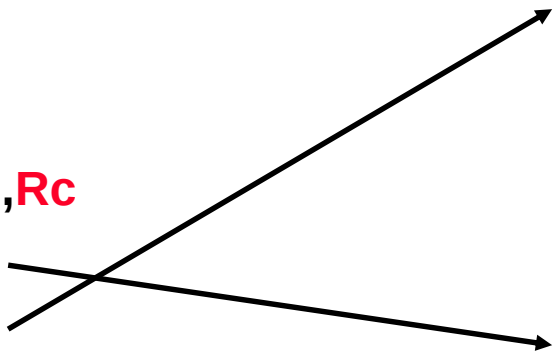
Software Scheduling for Load Hazards

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
```

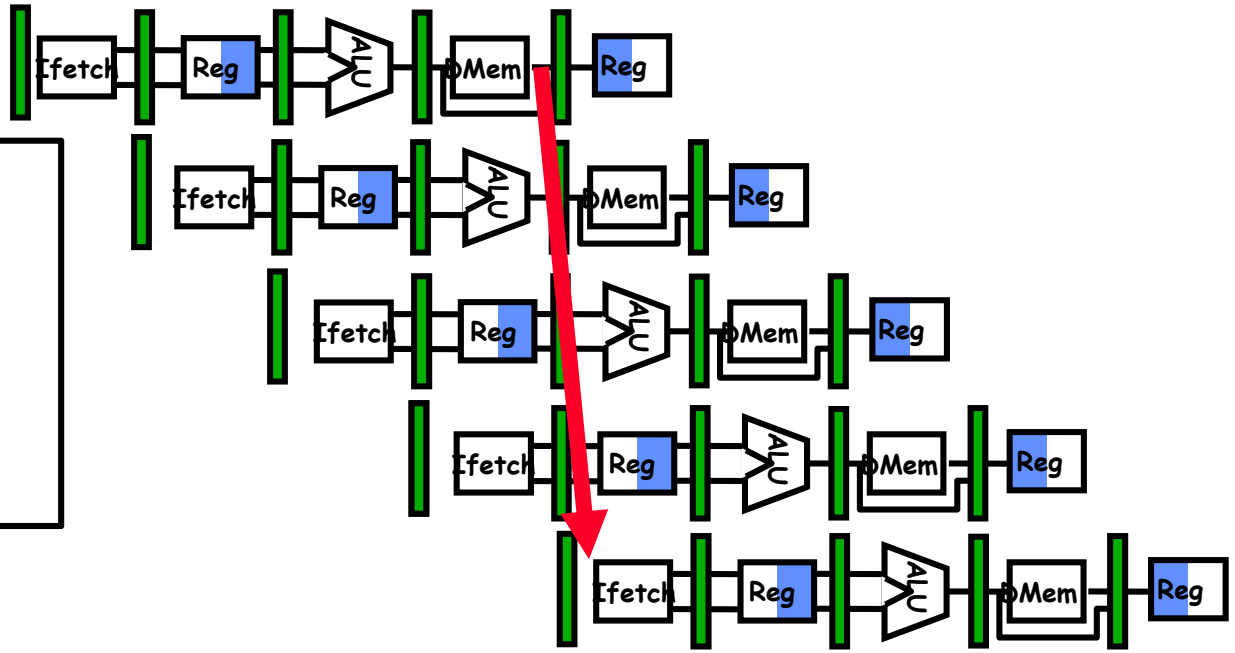


```
LW    Rb,b
LW    Rc,c
LW    Re,e
ADD   Ra,Rb,Rc
LW    Rf,f
SW    a,Ra
SUB   Rd,Re,Rf
SW    d,Rd
```

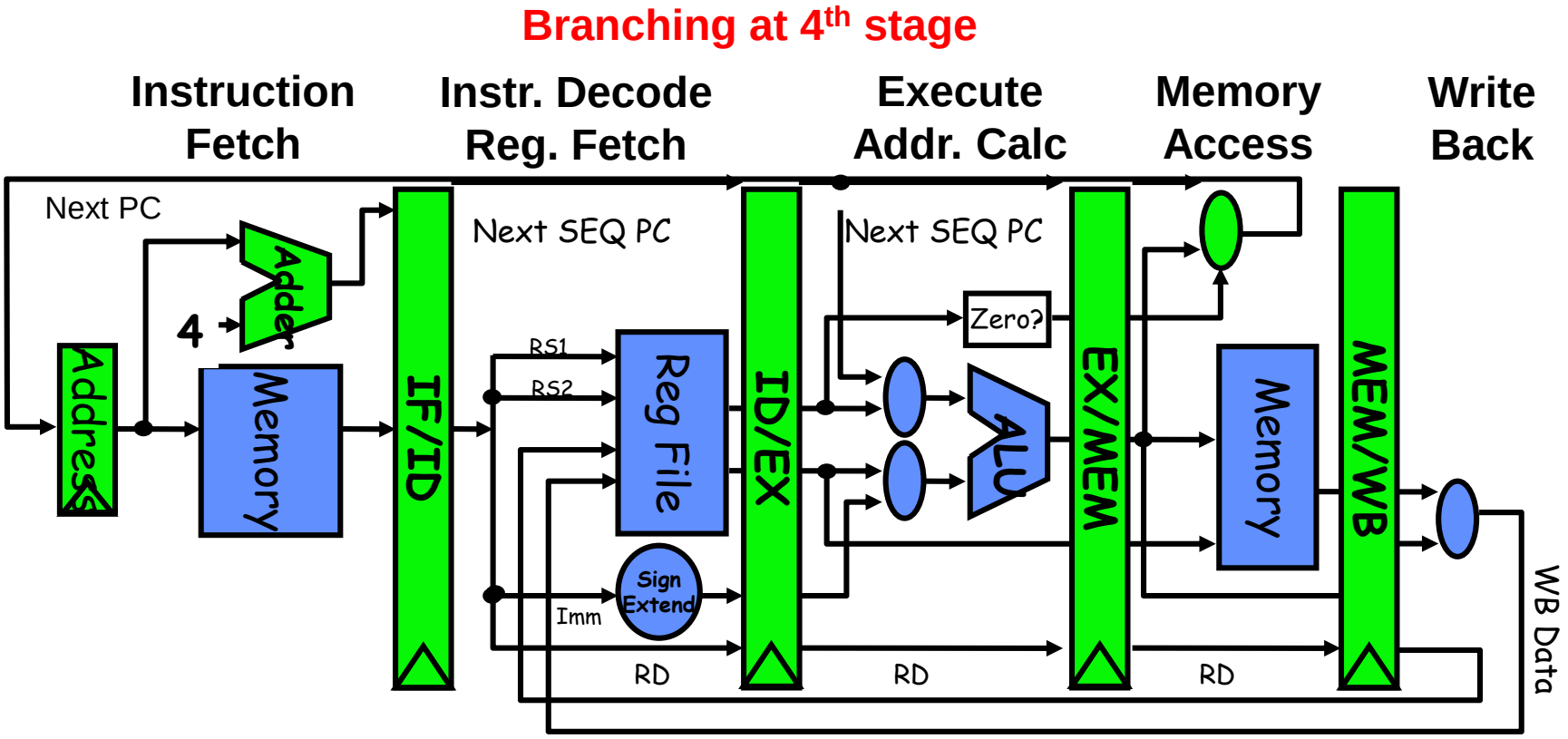
Control Hazard on Branches

=> Three Stage Stall

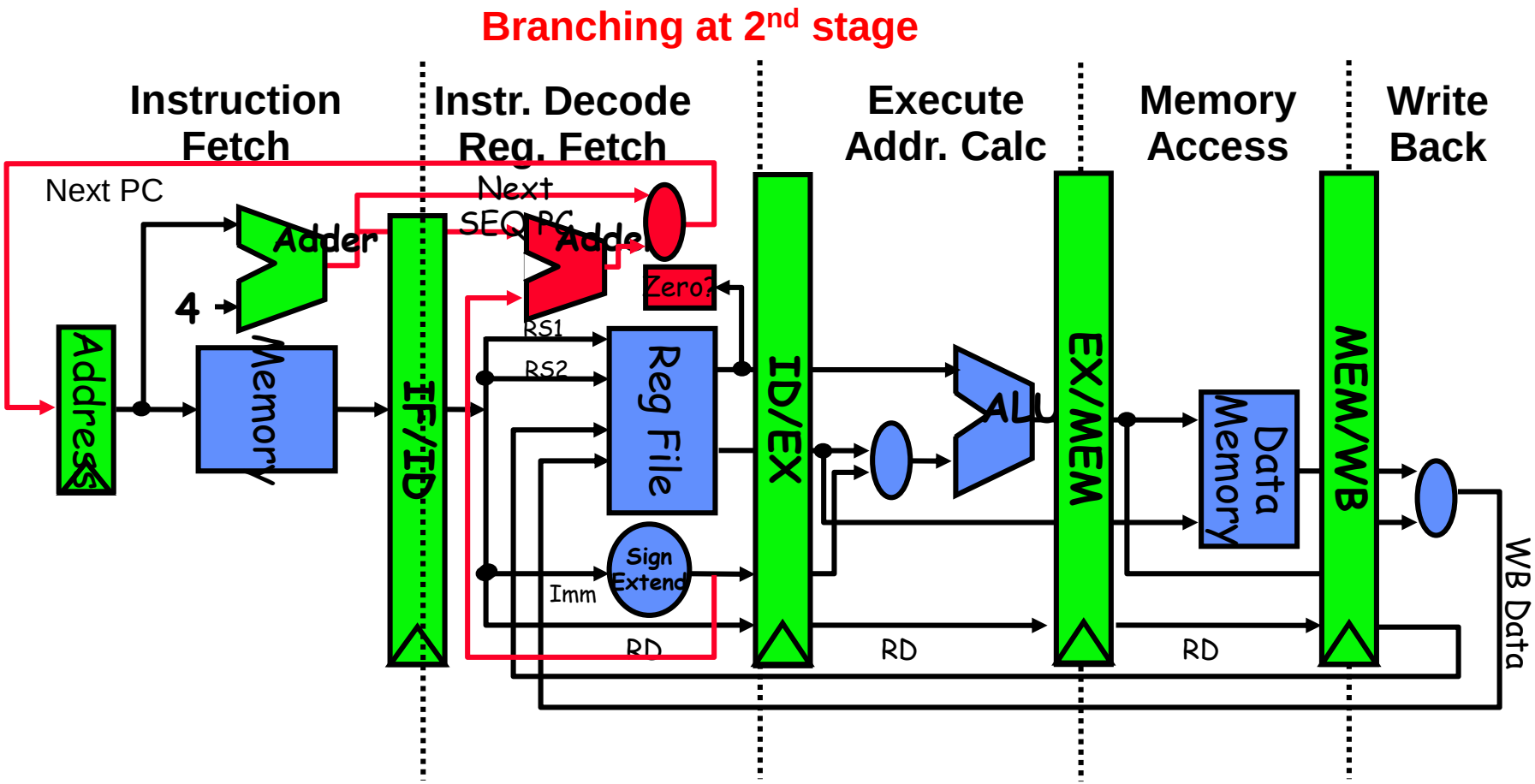
```
10: beq r1,r3,36
14: and r2,r3,r5
18: or  r6,r1,r7
22: add r8,r1,r9
36: xor r10,r1,r11
```



Conventional MIPS Pipeline



Branch Optimized MIPS Pipeline



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