Lecture 16
Pipeline Hazards

Guest Lecture by

Mahesh

Today's Lecture

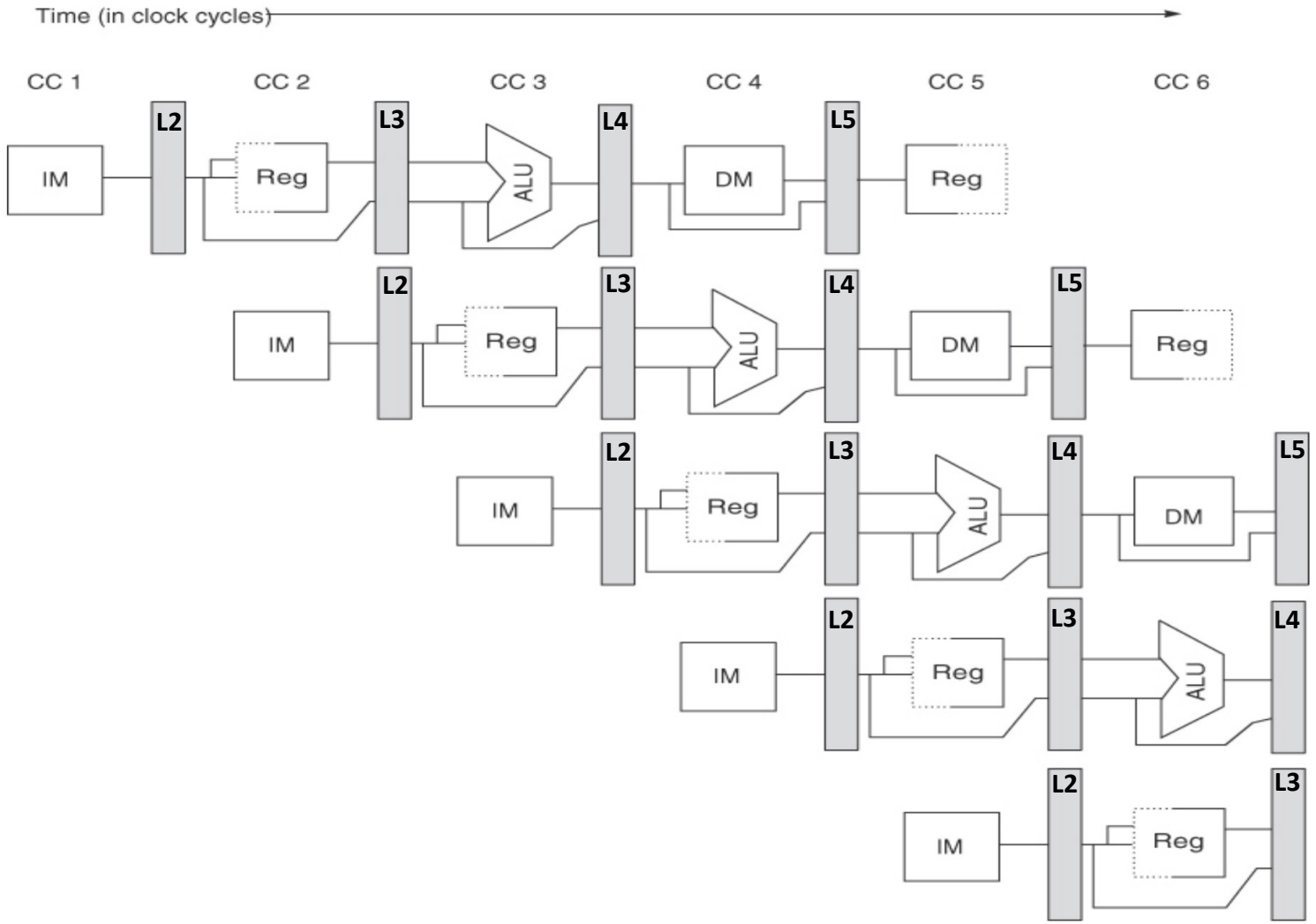
- Recap: 5-stage Pipeline Design
- Pipeline hazards
- Structural hazards
- Data hazards and data dependence analysis

Recap: 5-stage Pipeline Design

lw $1, 6($2)

add $3, $5, $0

sw $4, 8($6)



Issues/Conflicts with Pipelining

- Unified memory accessed for instruction and data in the same cycle
- Registers are read and written in the same cycle
- R-Type instructions can’t skip the DM stage, else conflict for WB
- Consuming instruction may have to wait for the producer
- Branch target is known only at the end of its execution – what do you do in the meantime?

Issues/Conflicts with Pipelining: Branching

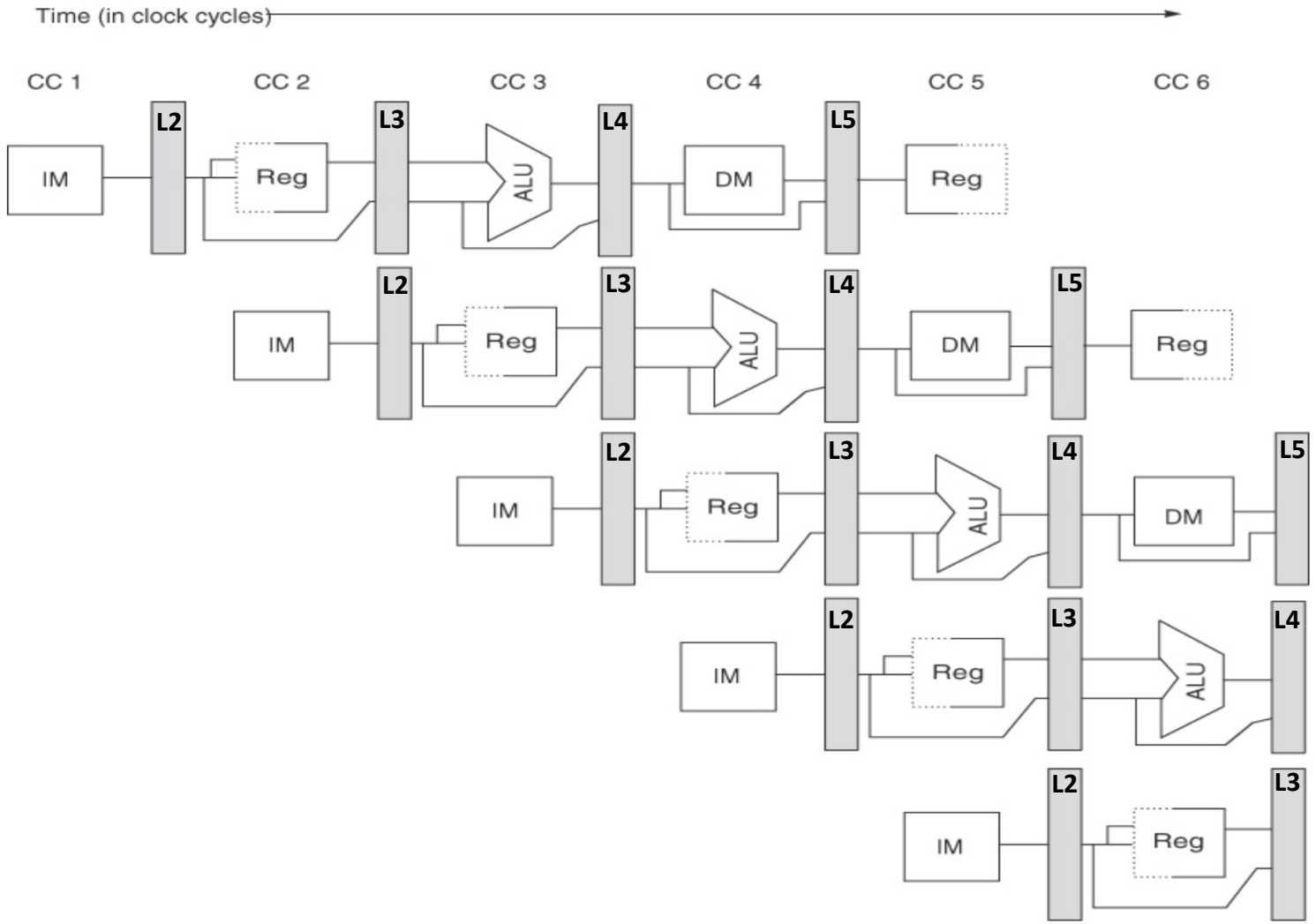
beq $1, $2, 1000

add $3, $5, $0

sub $4, $8, $6

and $7, $3, $5

sw $7, 2($3)



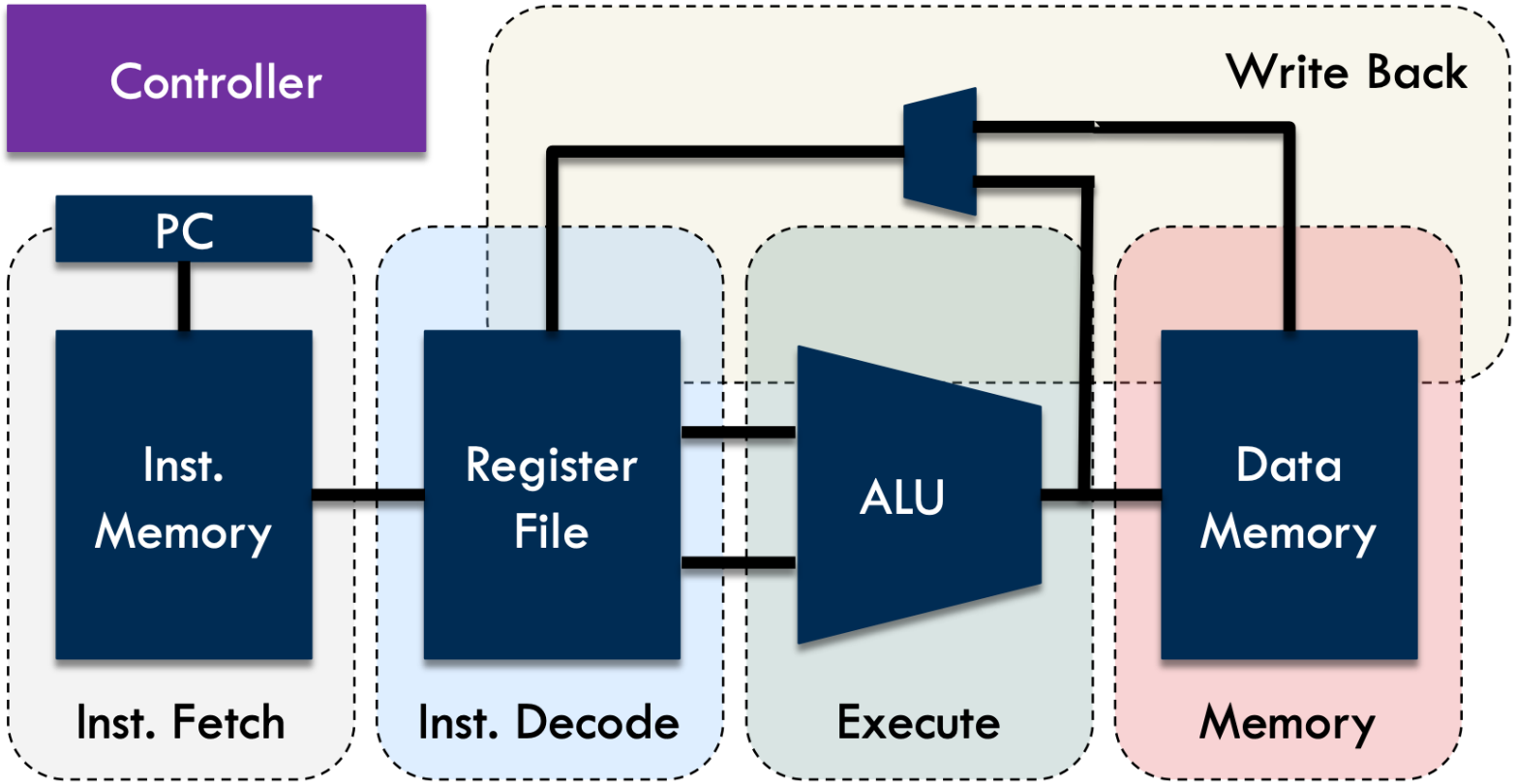
Pipeline Hazards

- **Structural Hazards:** Multiple instructions contend for the same resource.

- **Data Hazards:** A dependent instruction cannot proceed because it needs a value that hasn't been produced yet.

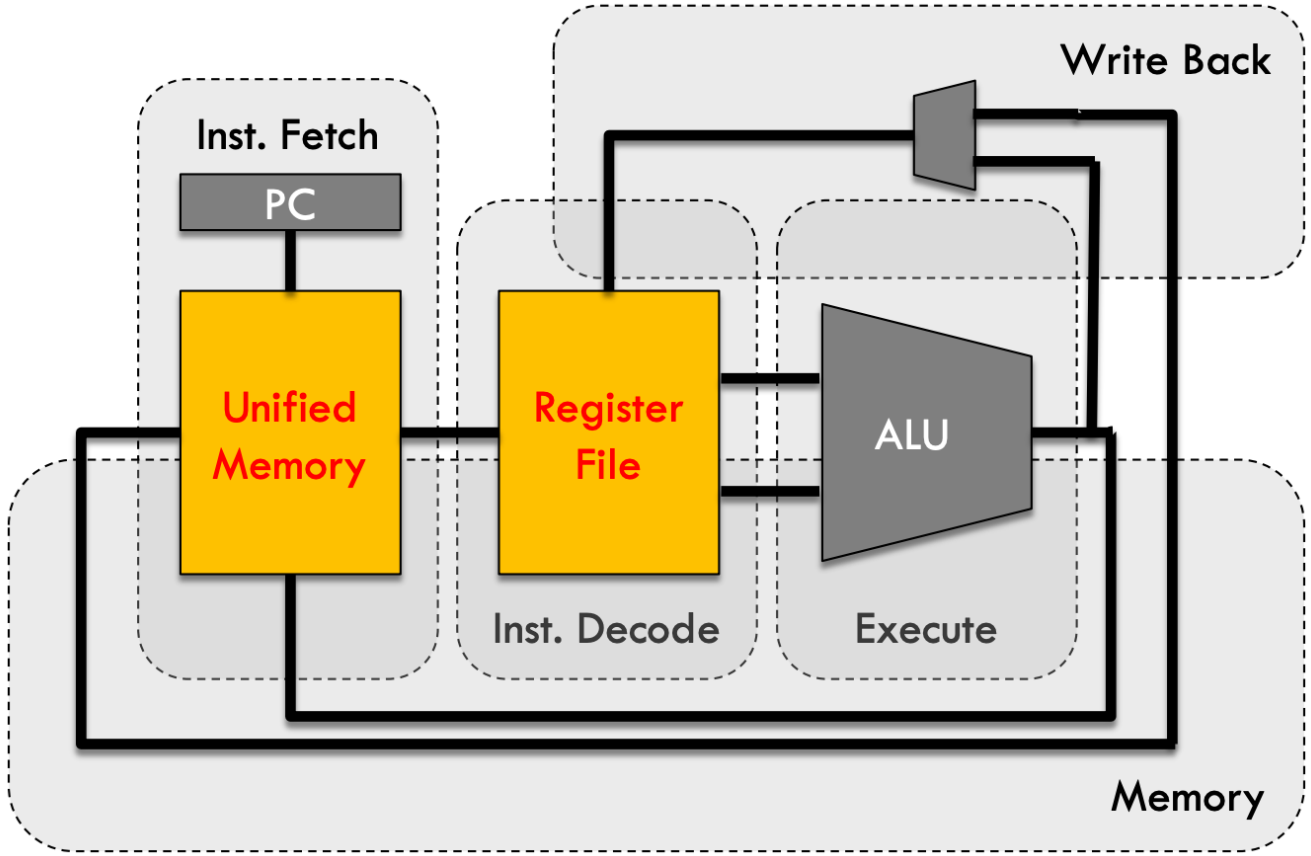
- **Control Hazards:** The next instruction cannot be fetched because the outcome of an earlier branch is unknown.

Structural Hazards



Structural Hazards

Unified memory and register file



Structural Hazards

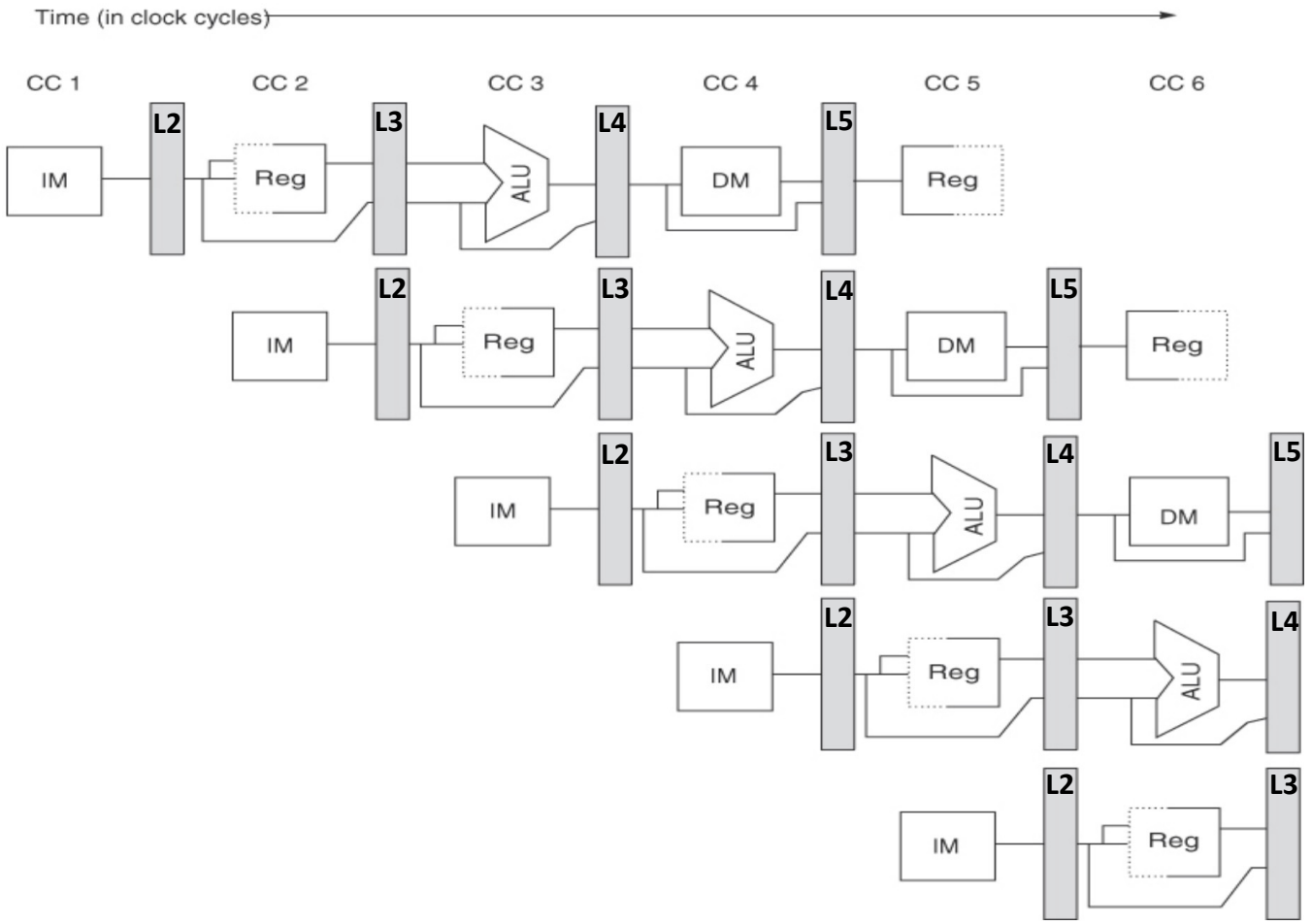
- Unified memory for instruction and data

lw $1, 6($2)

lw $3, 4($8)

sub $6, $5, $4

add $0, $7, $9



Structural Hazards

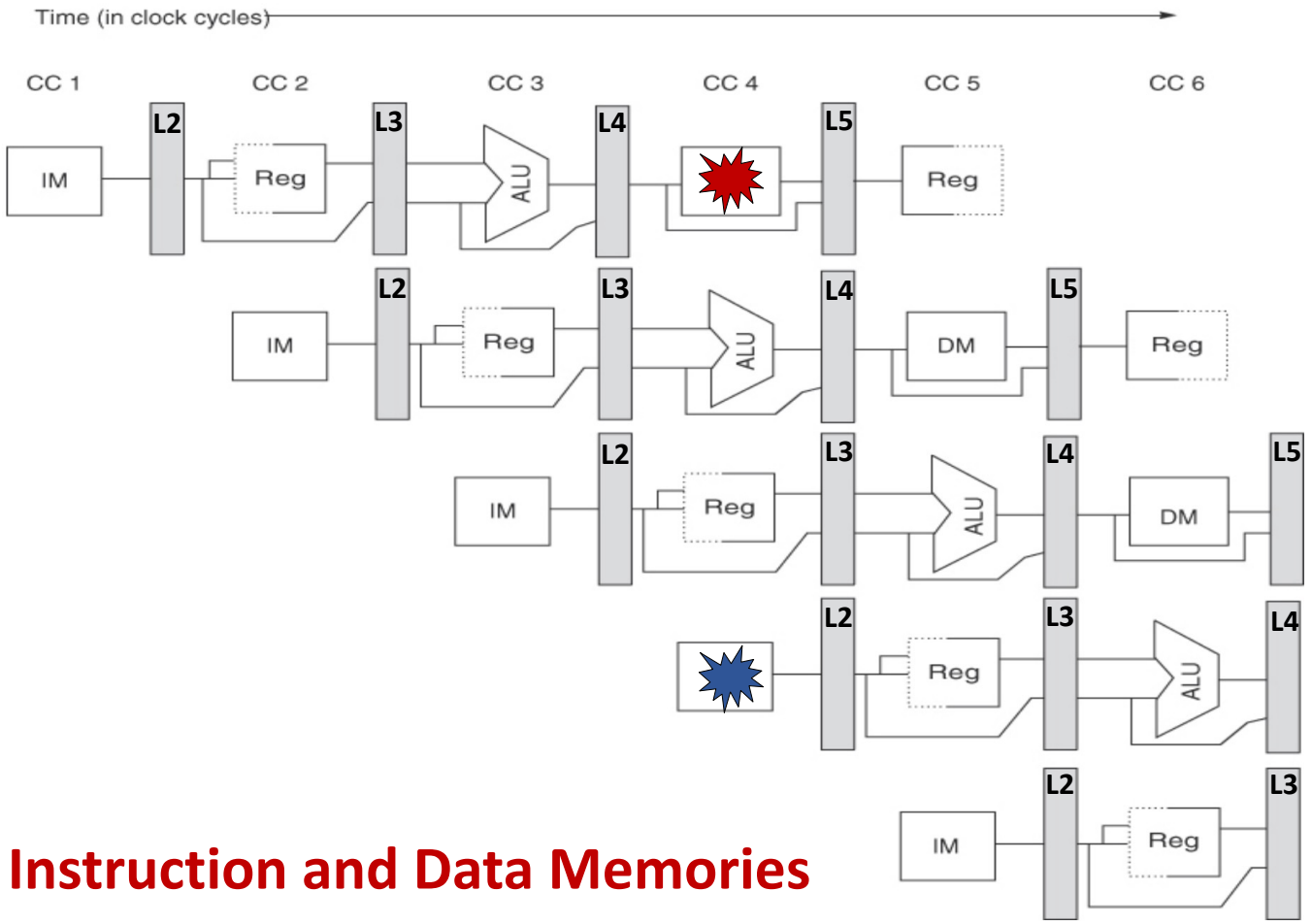
- Unified memory for instruction and data

lw $1, 6($2)

lw $3, 4($8)

sub $6, $5, $4

add $0, $7, $9



Separate Instruction and Data Memories

Structural Hazards

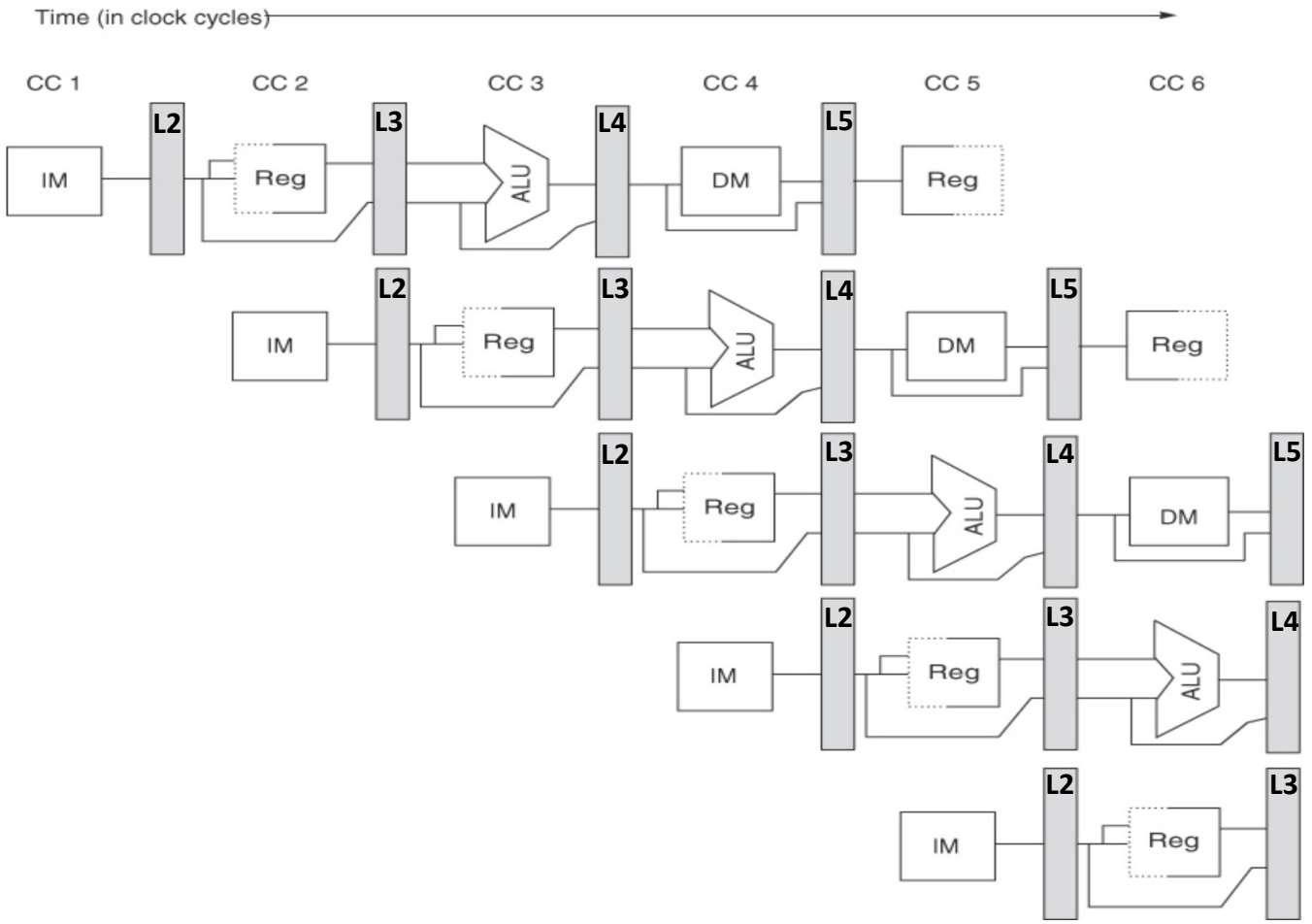
- Unified memory for instruction and data
- Register file with shared read/write access port

lw $1, 6($2)

lw $3, 4($8)

sub $6, $5, $4

add $0, $7, $9



Structural Hazards

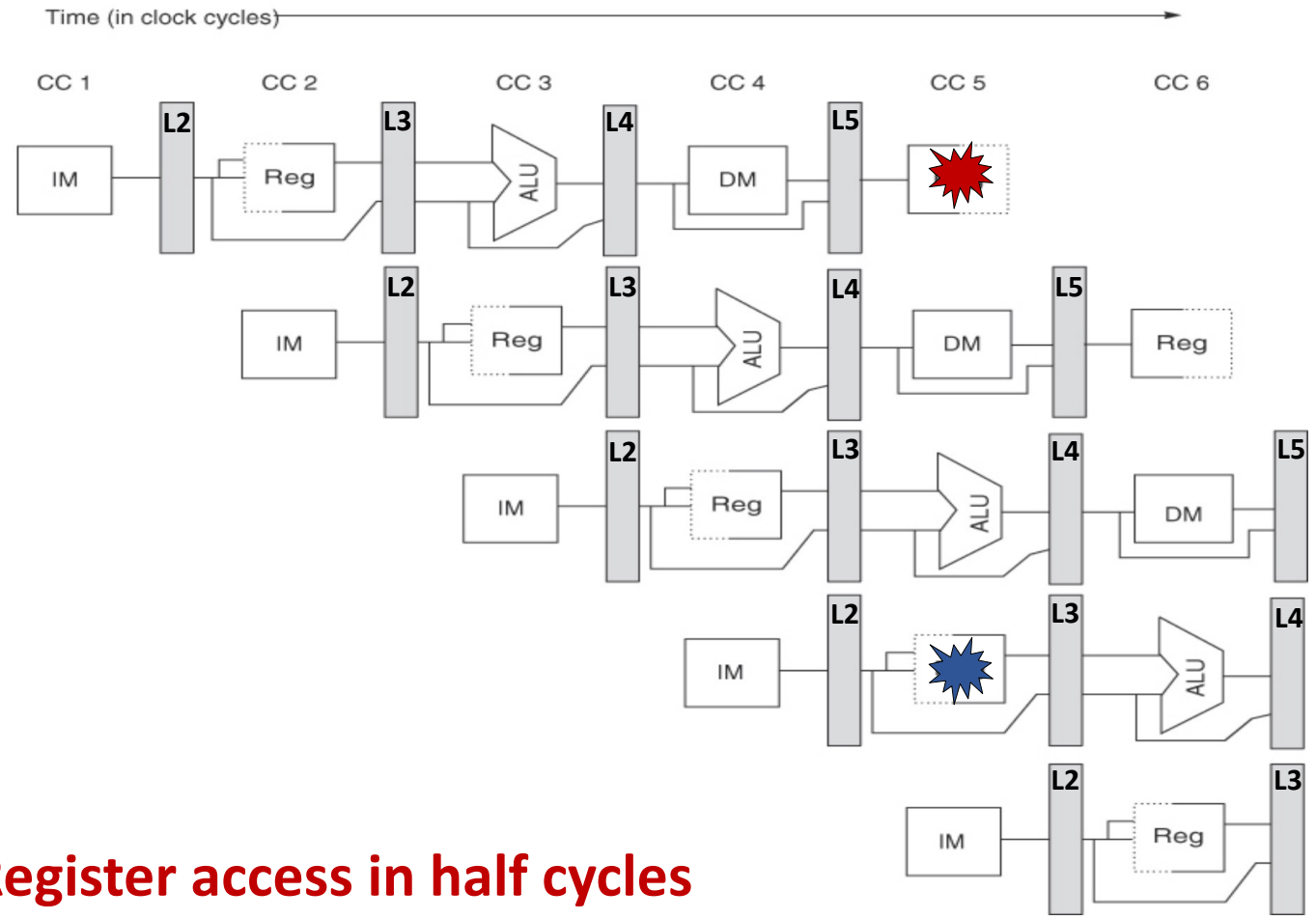
- Unified memory for instruction and data
- Register file with shared read/write access port

lw $1, 6($2)

lw $3, 4($8)

sub $6, $5, $4

add $0, $7, $9



Register access in half cycles

Data Hazards

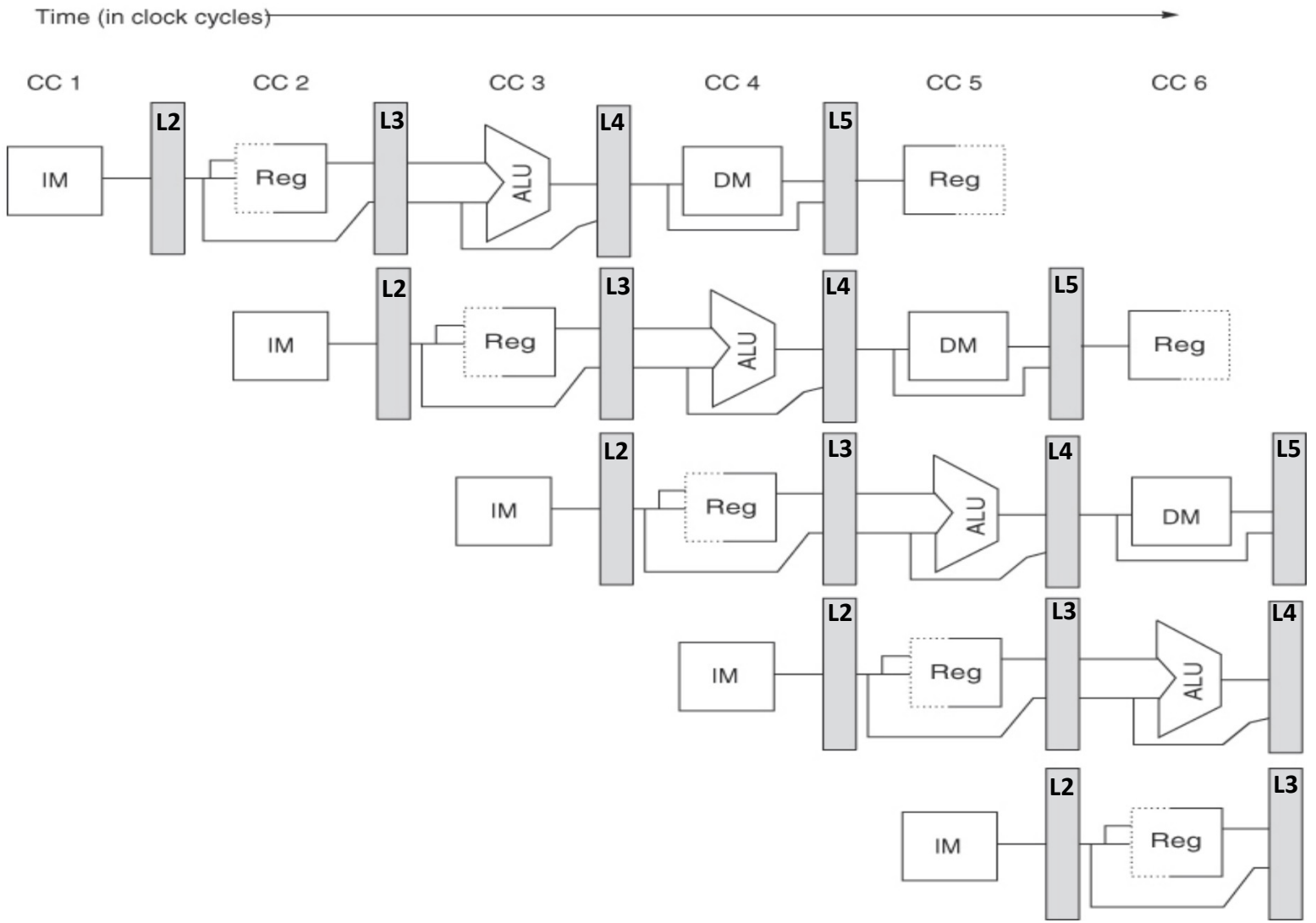
- An instruction *produces* a value in a given pipeline stage.
- A subsequent instruction *consumes* that value in a pipeline stage.
- The consumer may have to be delayed so that the time of consumption is later than the time of production.

Data Hazards

add $r3, $r1, $r2

add $r5, $r3, $r4

sub $r7, $r8, $r9

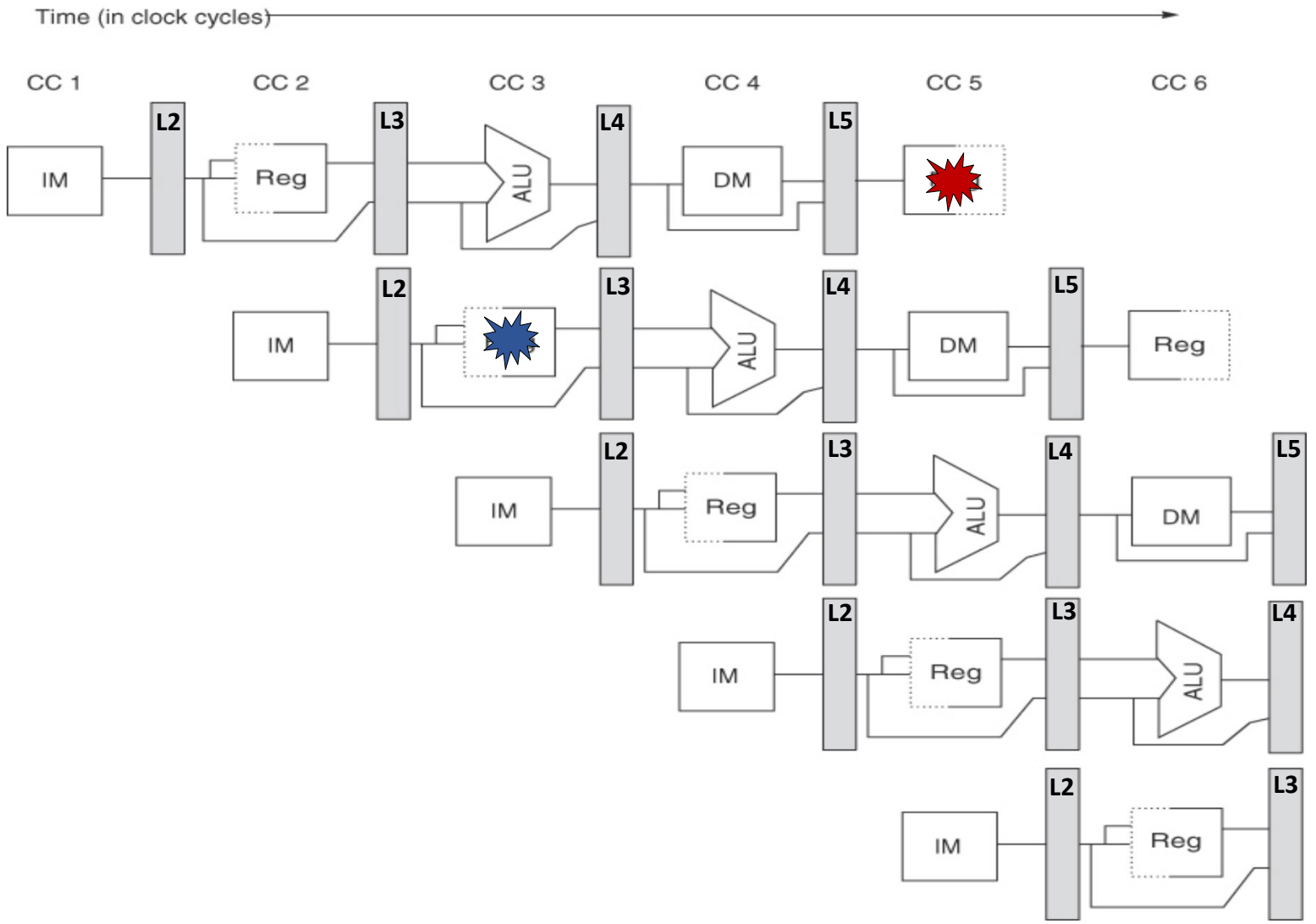


Data Hazards

add $r3, $r1, $r2

add $r5, $r3, $r4

sub $r7, $r8, $r9



Handling Data Hazards: Stalling

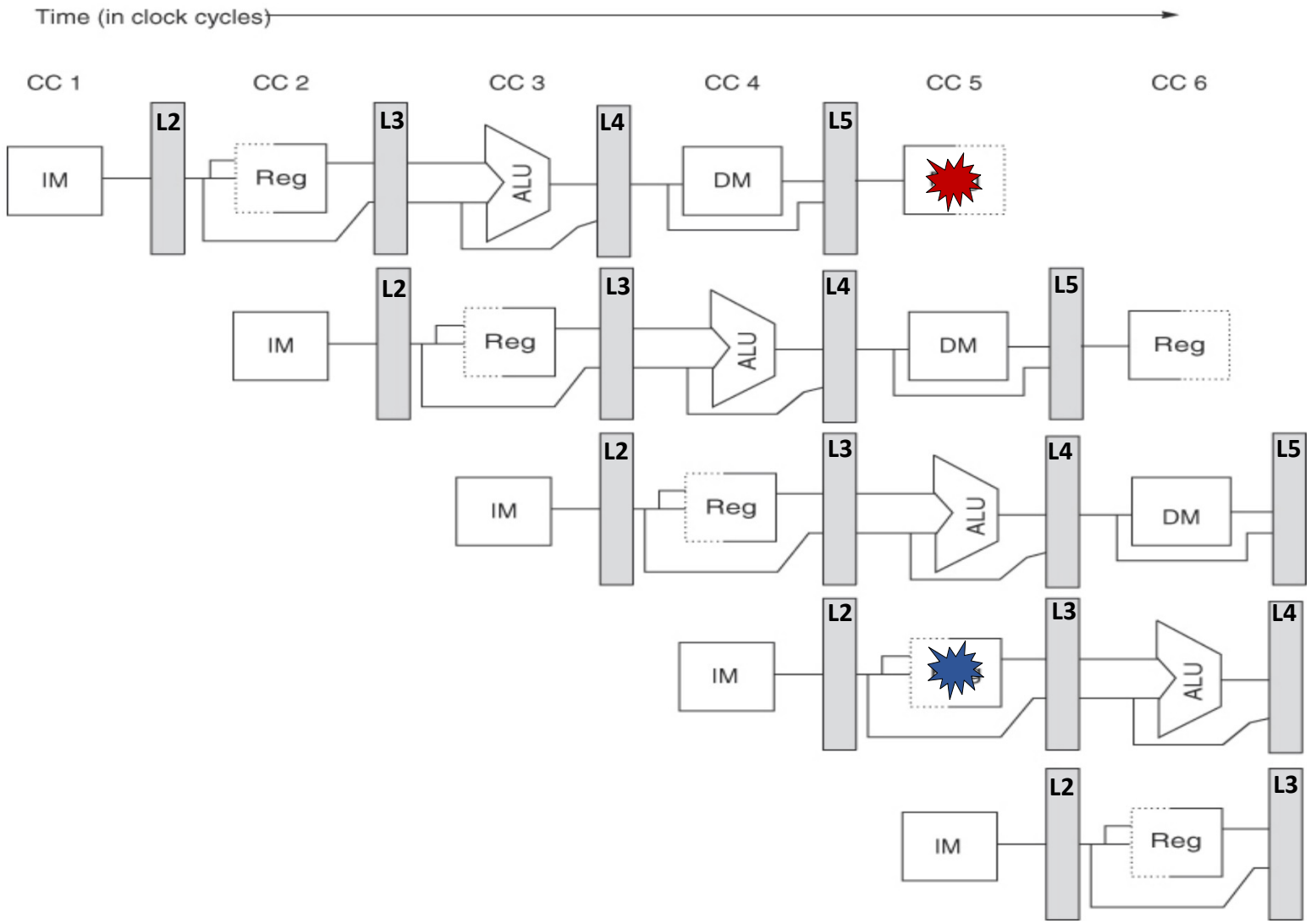
add $r3, $r1, $r2

nop

nop

add $r5, $r3, $r4

sub $r7, $r8, $r9

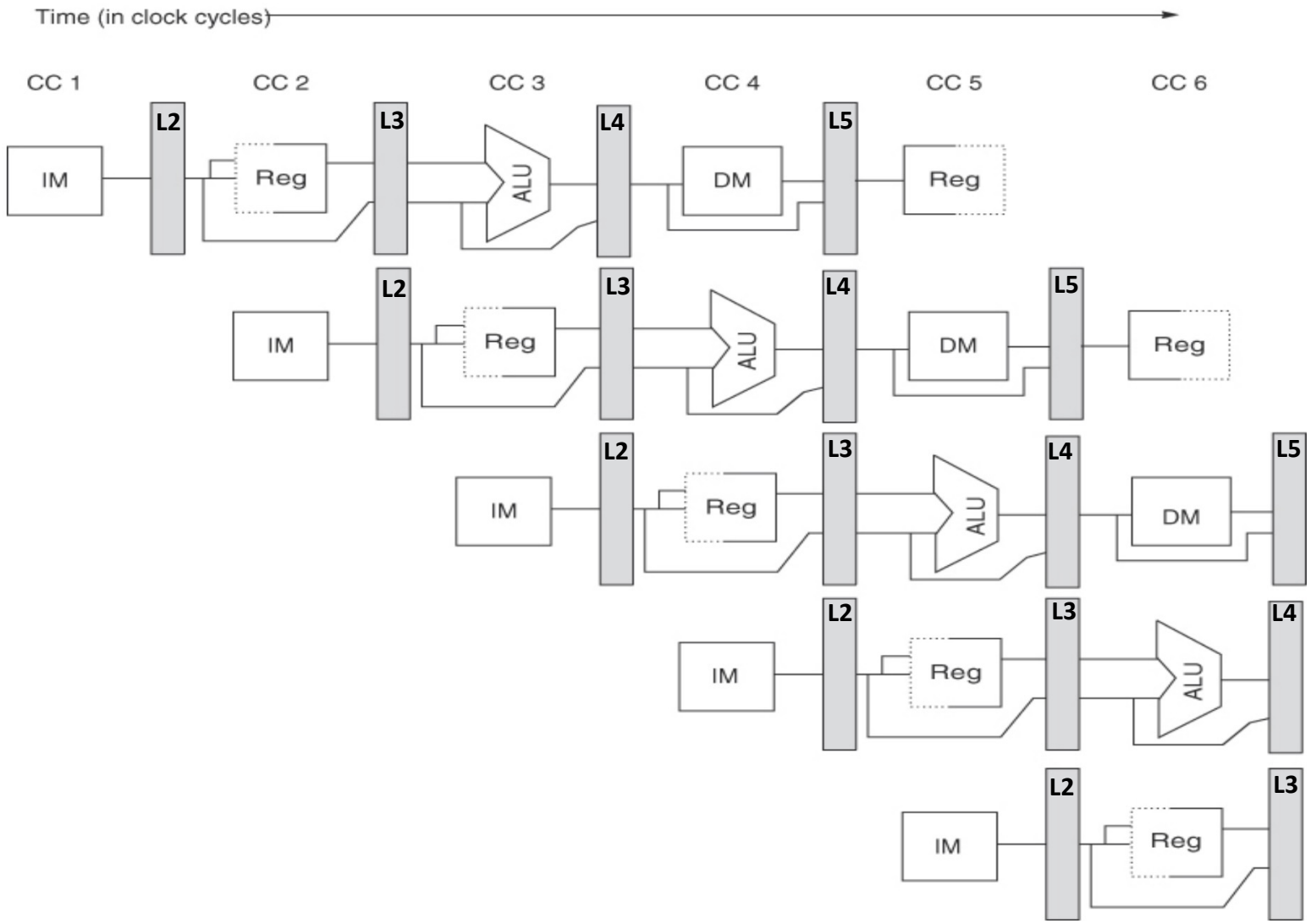


Handling Data Hazards: Stalling

I1: add $r3, $r1, $r2

I2: add $r5, $r3, $r4

I3: sub $r7, $r8, $r9



Handling Data Hazards: Stalling

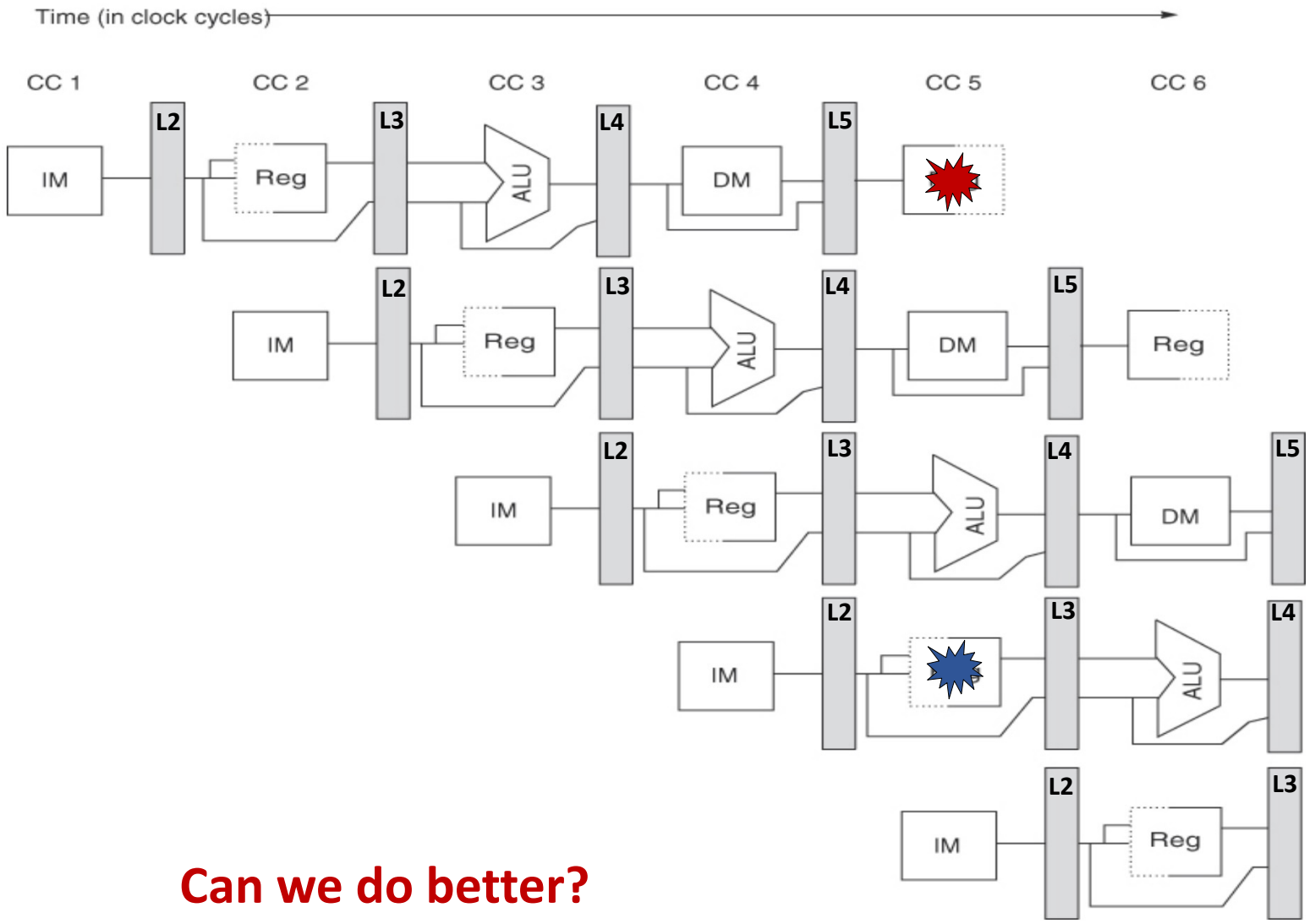
add $r3, $r1, $r2

nop

nop

add $r5, $r3, $r4

sub $r7, $r8, $r9

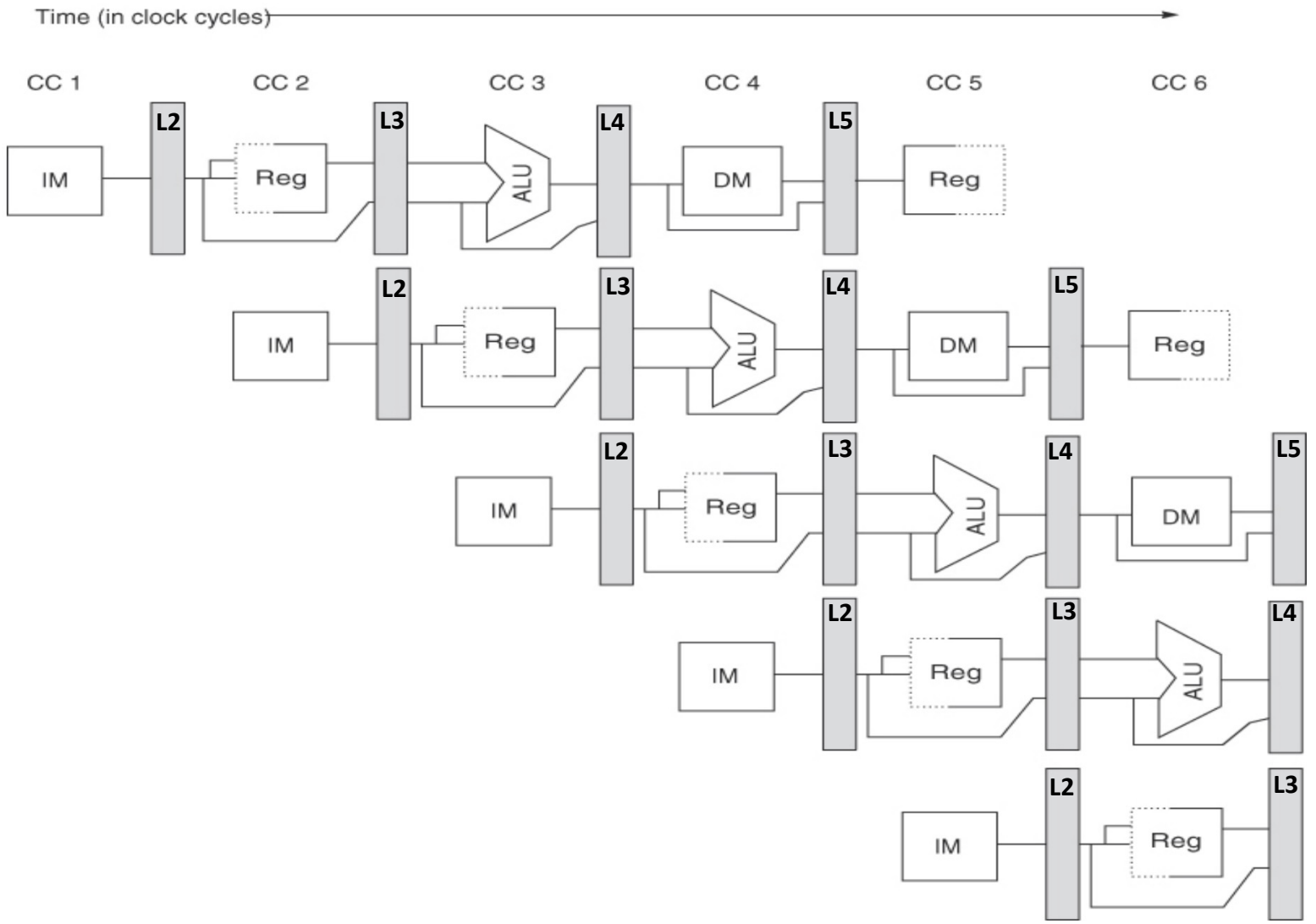


Can we do better?

Handling Data Hazards: Forwarding/Bypassing

I1: add $r3, $r1, $r2

I2: add $r5, $r3, $r4



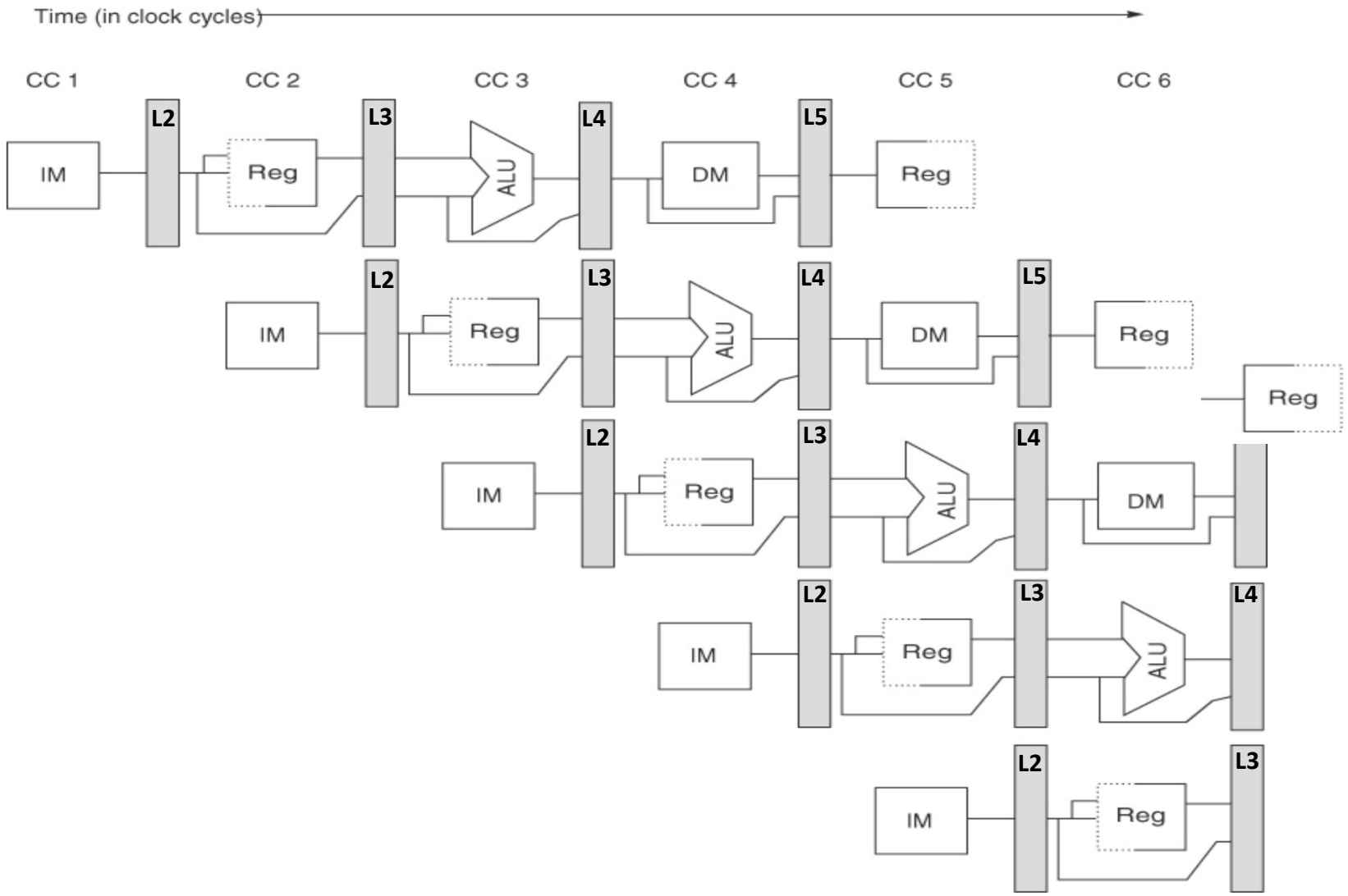
Handling Data Hazards: Forwarding/Bypassing

I1: add $r3, $r1, $r2

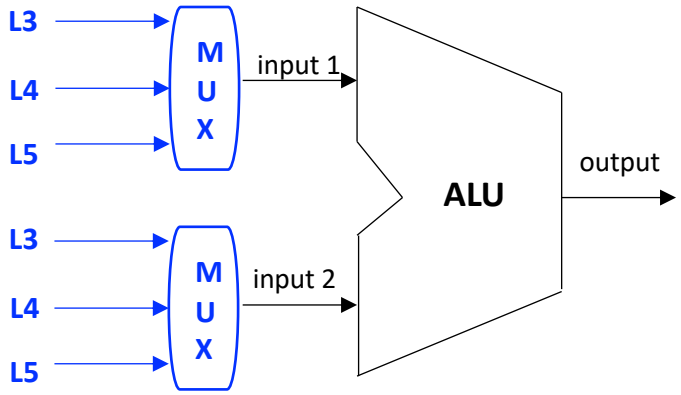
I2: add $r5, $r3, $r4

I3: sub $r7, $r3, $r9

I4: add $r0, $r3, $r8



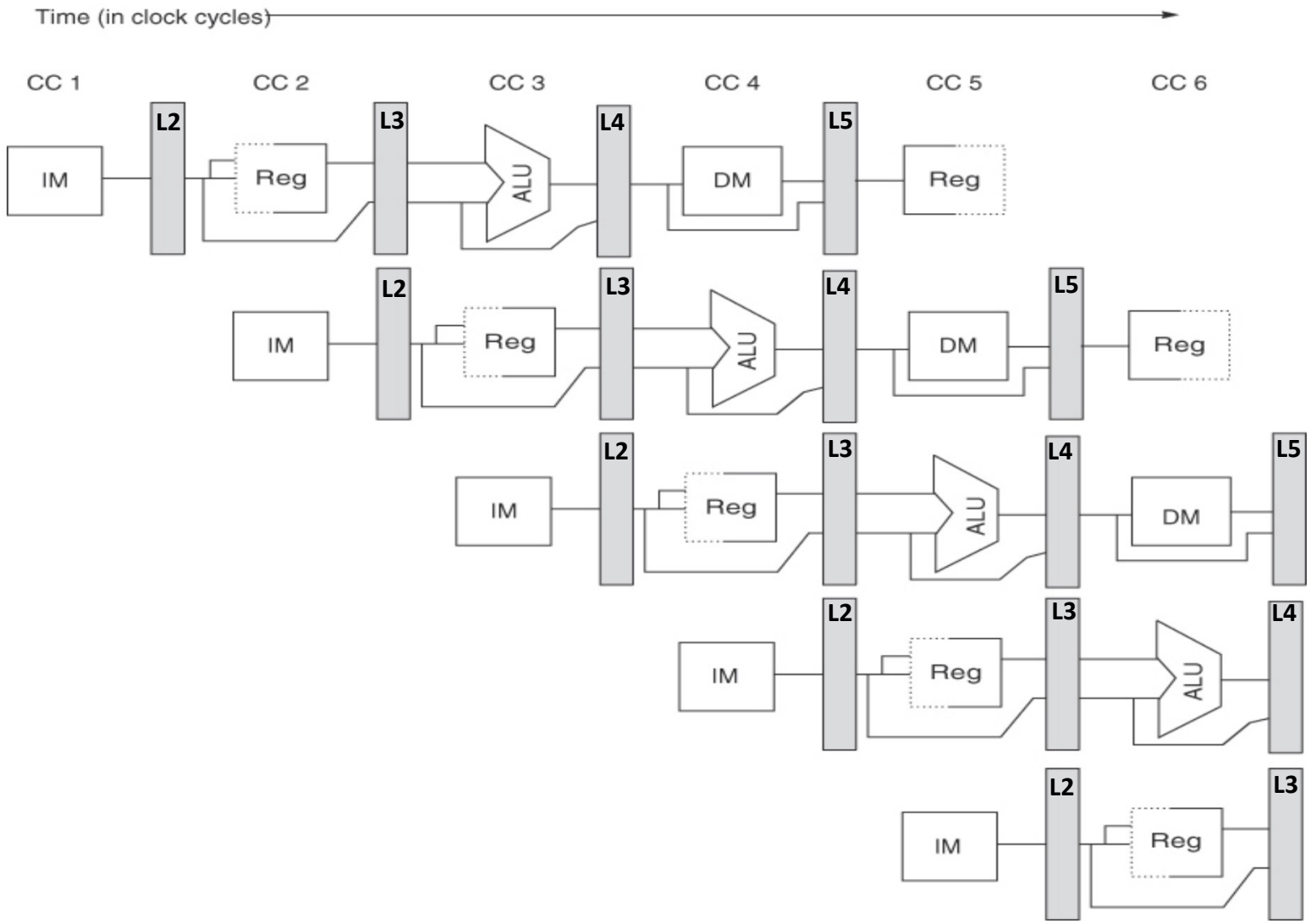
ALU With Forwarding



Forwarding/Bypassing With Loads

I1: lw $r1, 8($r2)

I2: lw $r4, 8($r1)

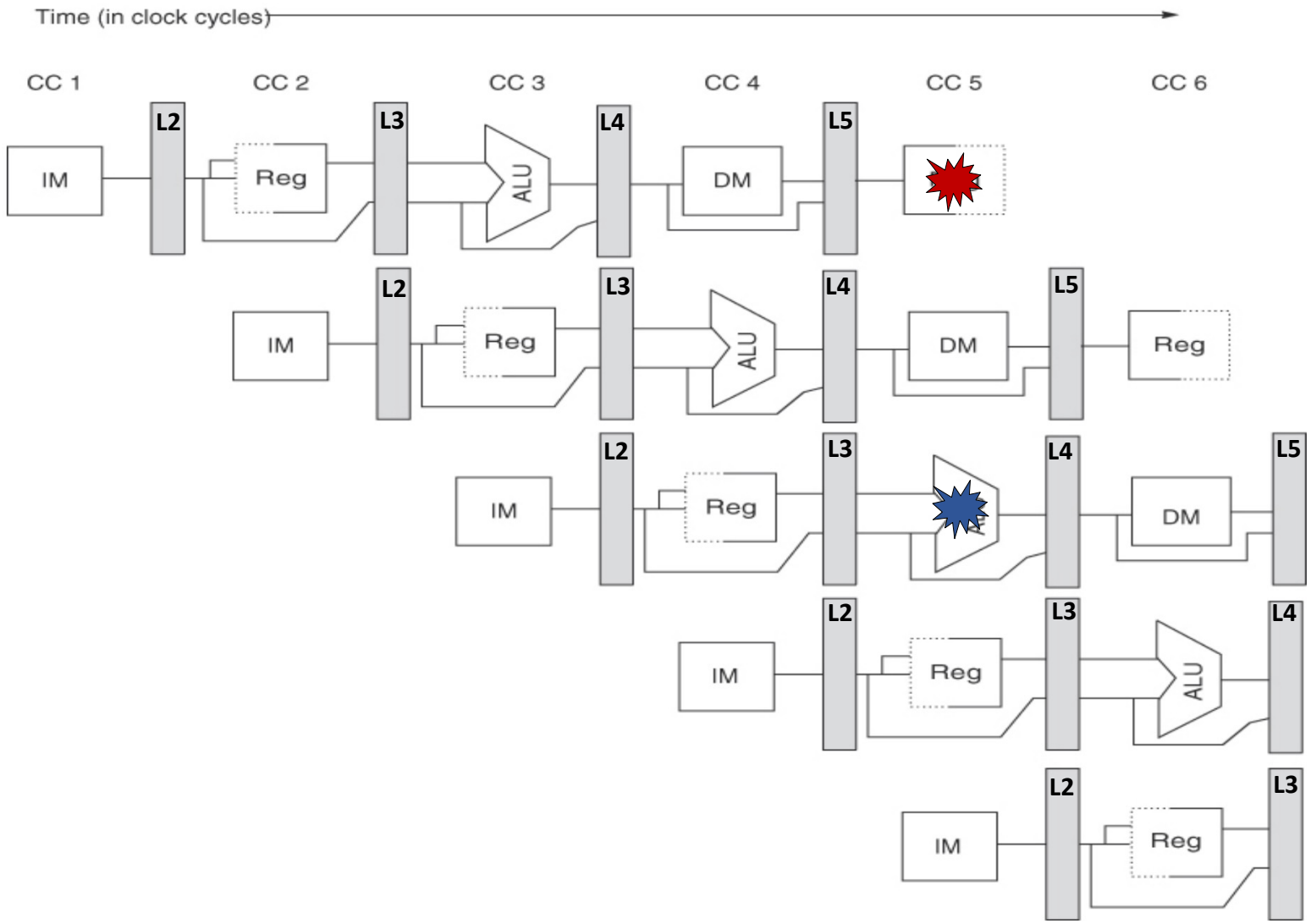


Forwarding/Bypassing With Loads

I1: lw $r1, 8($r2)

nop

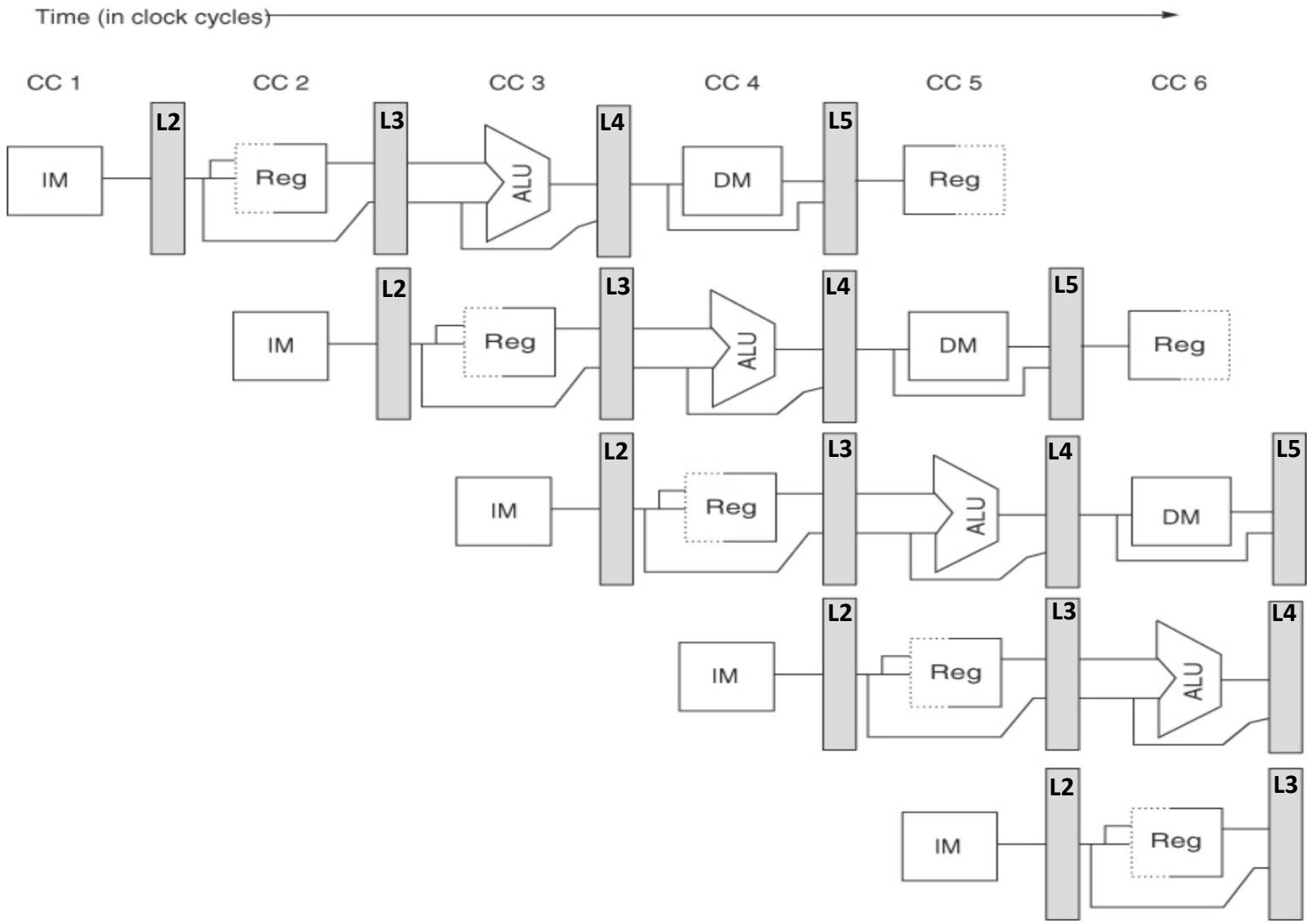
I2: lw $r4, 8($r1)



Forwarding/Bypassing With Loads

I1: lw $r1, 8($r2)

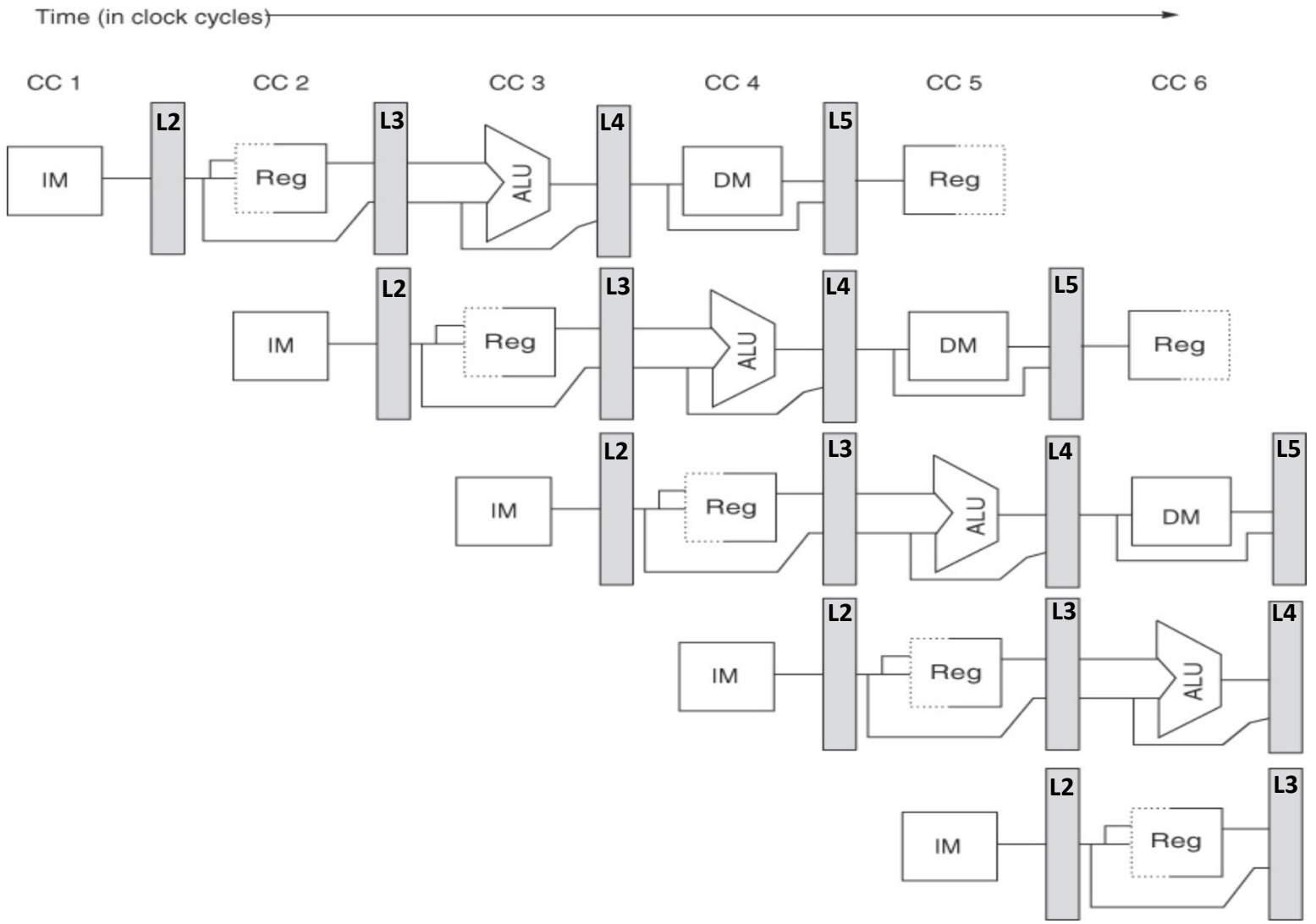
I2: lw $r1, 8($r3)



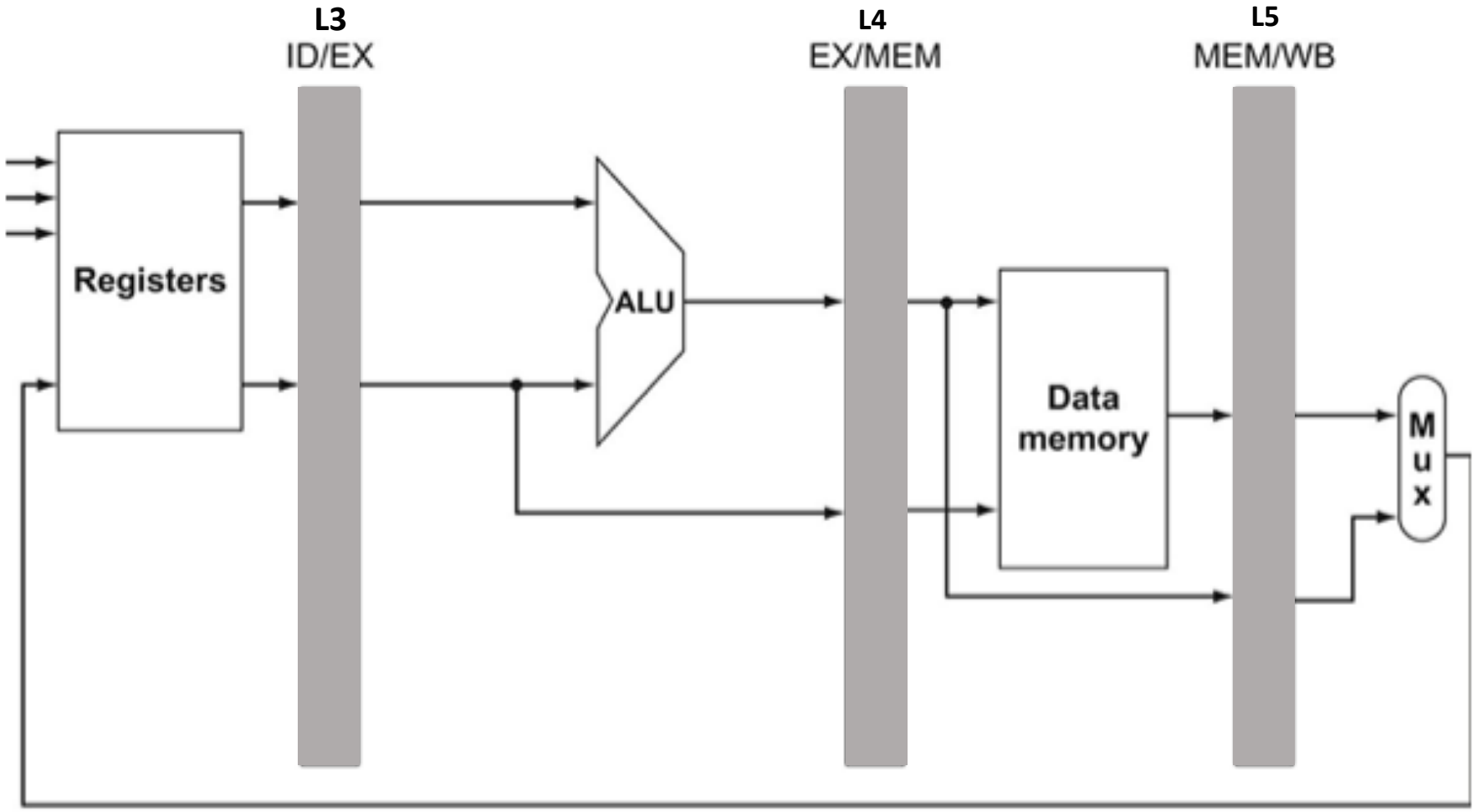
Forwarding/Bypassing With Loads

I1: lw $r3, 8($r2)

I2: lw $r1, 8($r3)

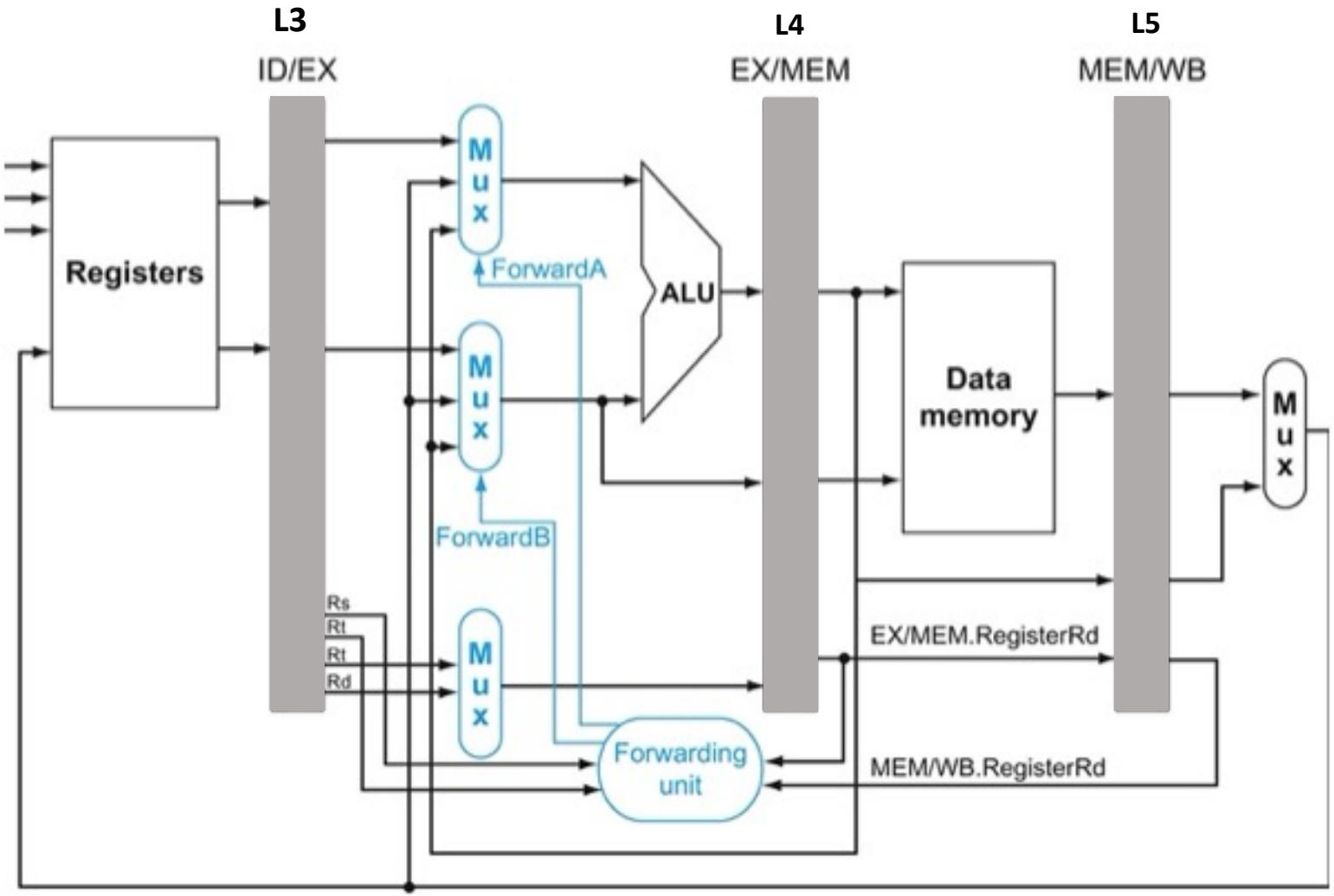


Forwarding Unit



Without Bypassing

Forwarding Unit



With Bypassing

Exercise: Identify all Data Hazards and Resolve Them

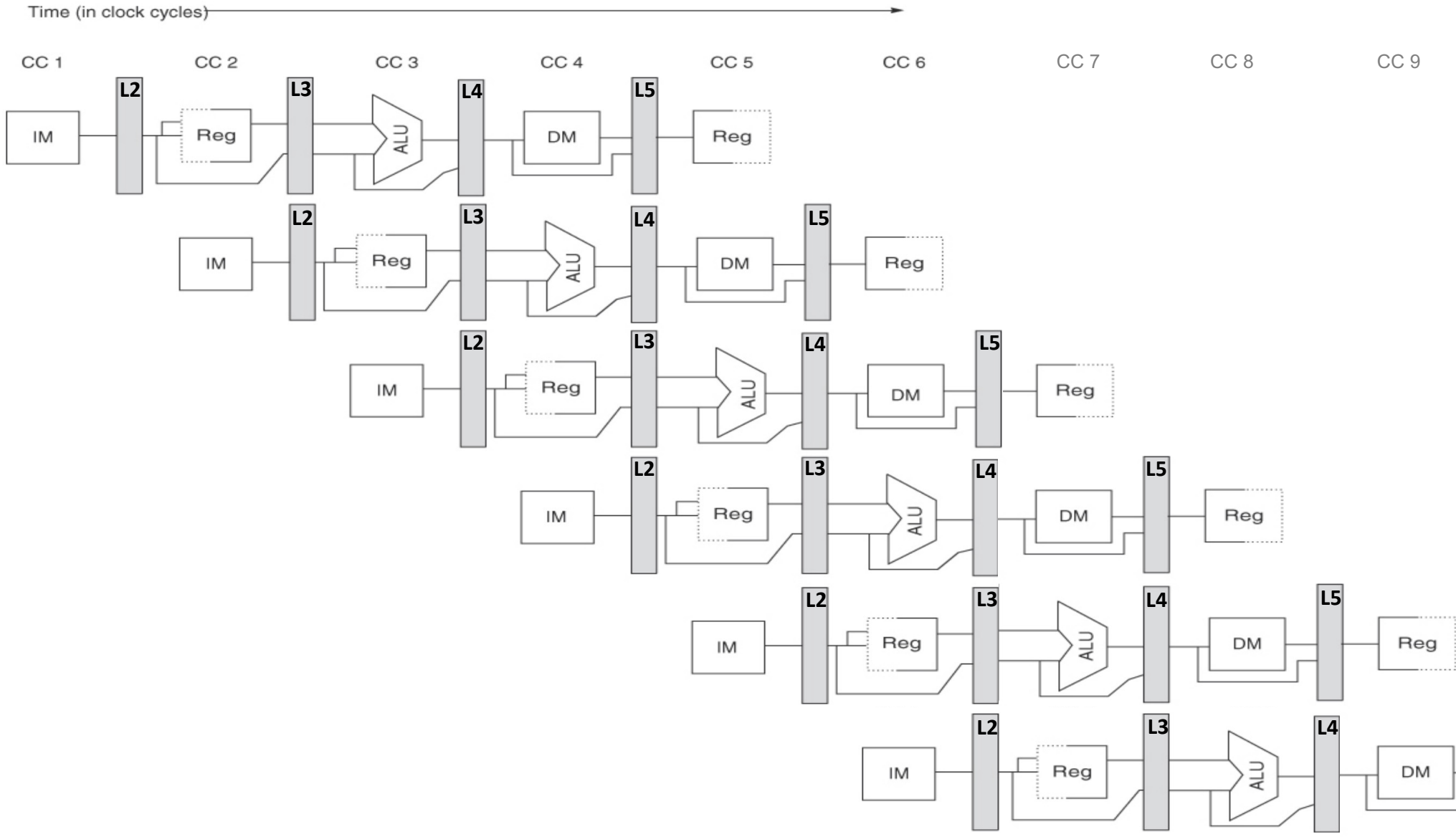
I1: add $r3, $r1, $r2

I2: and $r5, $r3, $r4

I3: add $r2, $r6, $r3

I4: lw $r6, 8($r3)

I5: sw $r6, 4($r2)



Detecting Data Hazards

Data Dependence Types

```
lw $1, 0($2)

add $2, $1, $0

sub $1, $1, $2

sw $2, 0($3)
```

Detecting Data Hazards

Data Dependence Types

lw $1, 0($2)	$1← Mem[$2]
add $2, $1, $0	$2← $1+$0
sub $1, $1, $2	$1← $1-$2
sw $2, 0($3)	Mem[$3] ← $2

Detecting Data Hazards

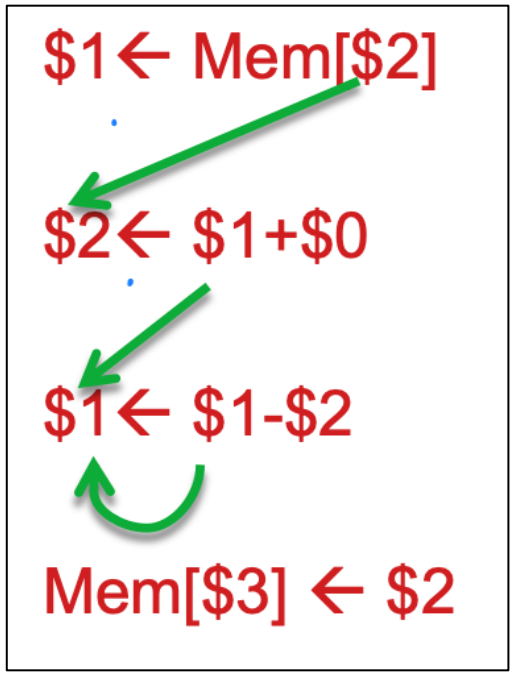
Data Dependence Types

Read-After-Write (RAW)

Detecting Data Hazards

Data Dependence Types

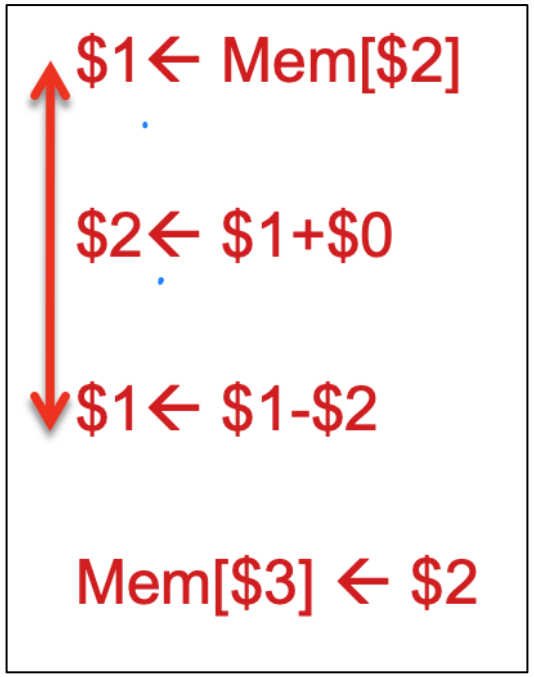
***W**rite-**A**fter-**R**ead* (WAR)



Detecting Data Hazards

Data Dependence Types

Write-After-Write (WAW)



Exercise: Identify Data Hazards

```
lw $7, 0($1)
sub $1, $4, $6
add $5, $5, $1
sw $1, 0($7)
add $4, $1, $2
lw $5, 0($1)
```

Exercise: Identify Data Hazards

```
lw $7, 0($1)
sub $1, $4, $6
add $5, $5, $1
sw $1, 0($7)
add $4, $1, $2
lw $5, 0($1)
```

$7 ← Mem[$1]

$1 = $4 – $6

$5 = $5 + $1

Mem[$7] ← $1

$4 = $1 + $2

$5 ← Mem[$1]

Exercise: Identify Data Hazards

10 HAZARDS

RAW

$7 ← Mem[$1]
$1 = $4 – $6
$5 = $5 + $1
Mem[$7] ← $1
$4 = $1 + $2
$5 ← Mem[$1]

5 Hazards

WAR

$7 ← Mem[$1]
$1 = $4 – $6
$5 = $5 + $1
Mem[$7] ← $1
$4 = $1 + $2
$5 ← Mem[$1]

4 Hazards

WAW

$7 ← Mem[$1]
$1 = $4 – $6
$5 = $5 + $1
Mem[$7] ← $1
$4 = $1 + $2
$5 ← Mem[$1]

1 Hazard

Thank you!!!

 **maheshl@cs.utah.edu**