PIPELINE HAZARDS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

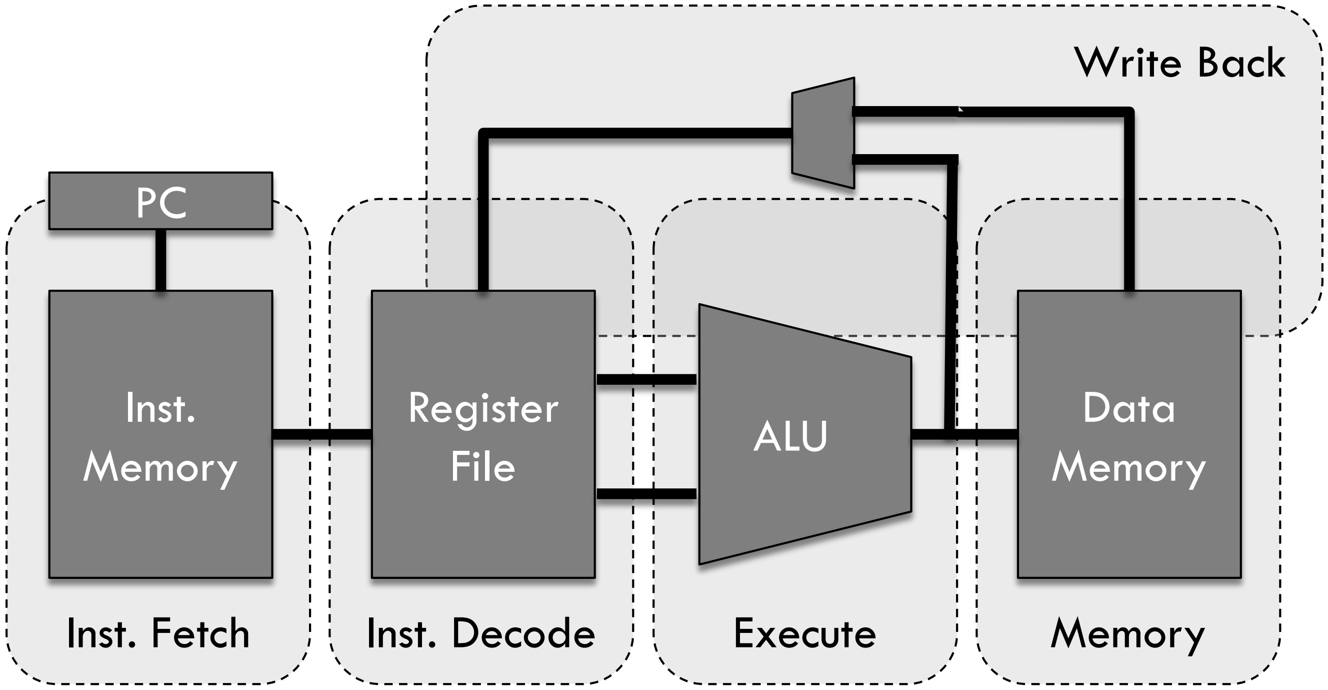
University of Utah

Overview

- This lecture
  - Pipeline Hazards
    - Structural
    - Data
    - Control

Pipelined Architecture

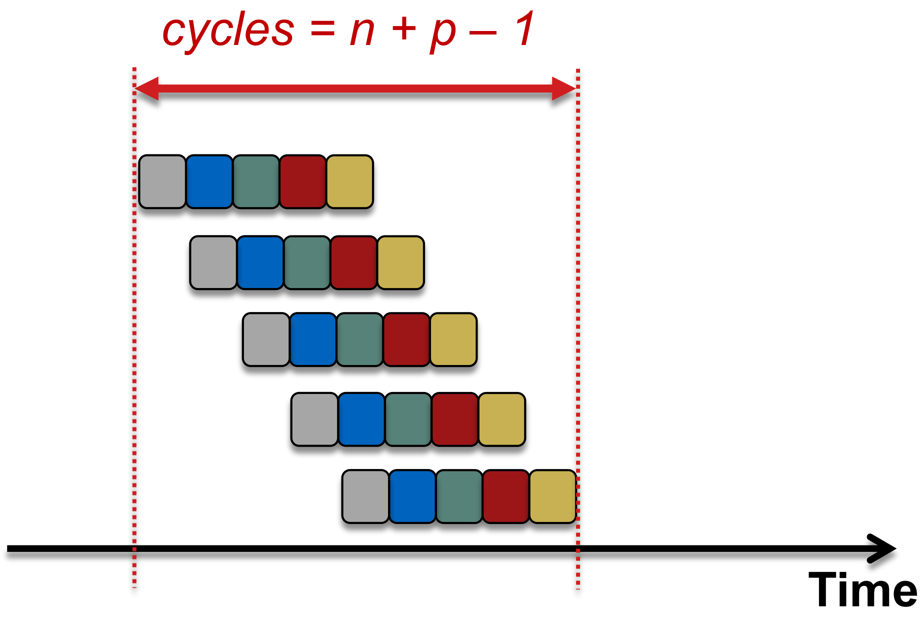
- Five stage pipeline
  - Critical path determines the cycle time



Pipelined Architecture

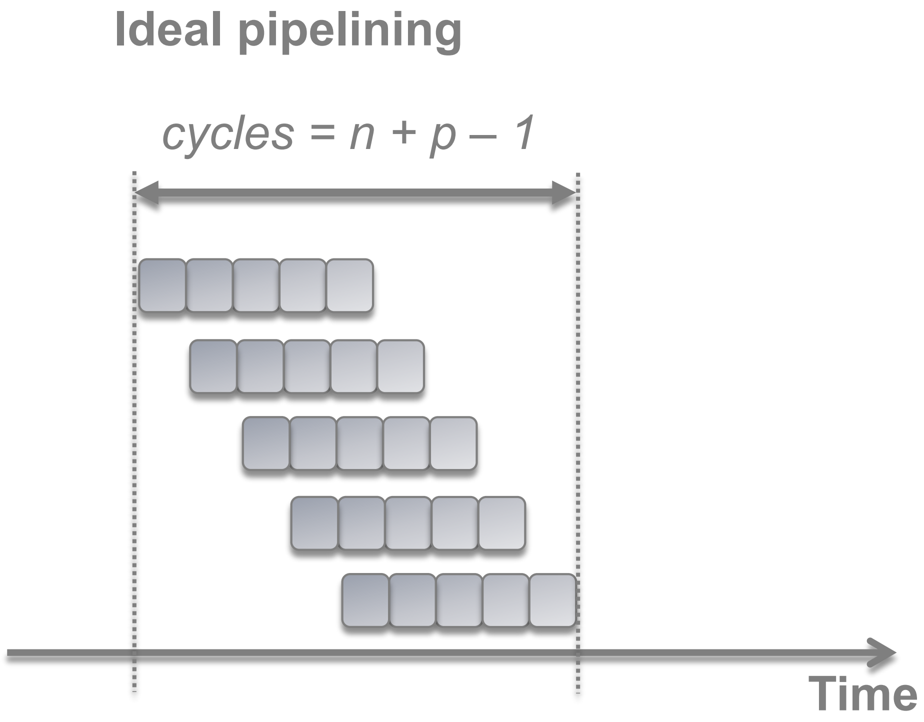
- The more overlapping instructions: the better performance.
  - n: # instructions, p: # pipeline stages, and s: # stall cycles

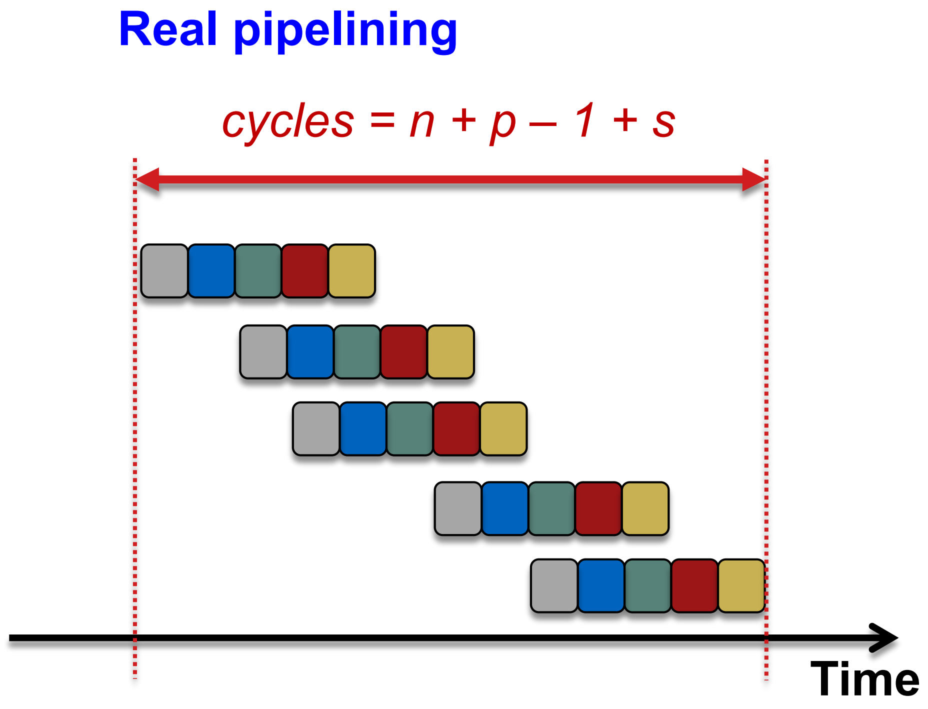
Ideal pipelining



Pipelined Architecture

- The more overlapping instructions: the better performance.
  - n: # instructions, p: # pipeline stages, and s: # stall cycles



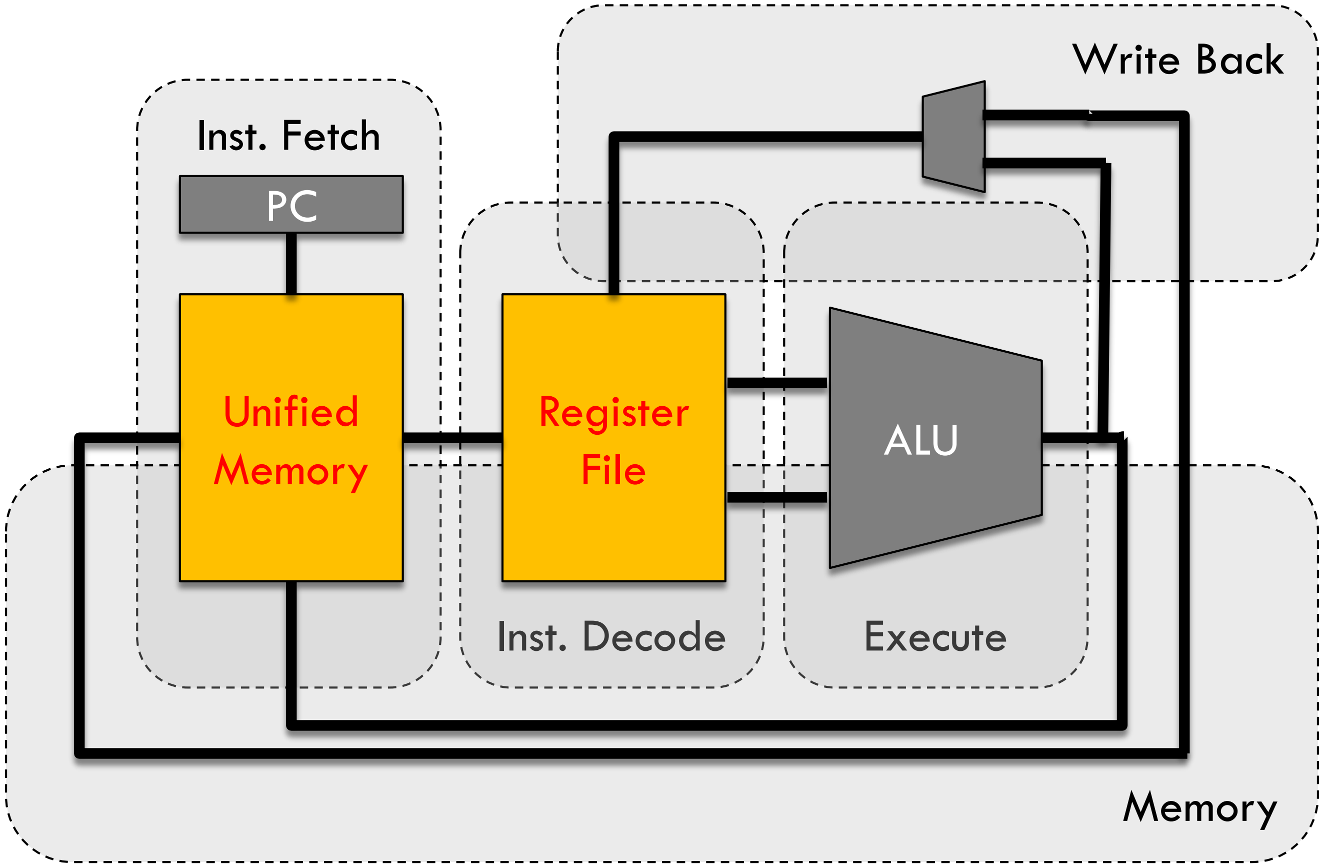


Pipeline Hazards

- Structural hazards: multiple instructions compete for the same resource
- Data hazards: a dependent instruction cannot proceed because it needs a value that hasn't been produced
- Control hazards: the next instruction cannot be fetched because the outcome of an earlier branch is unknown

Structural Hazard in the Pipeline

- Unified memory and register file.



Structural Hazards

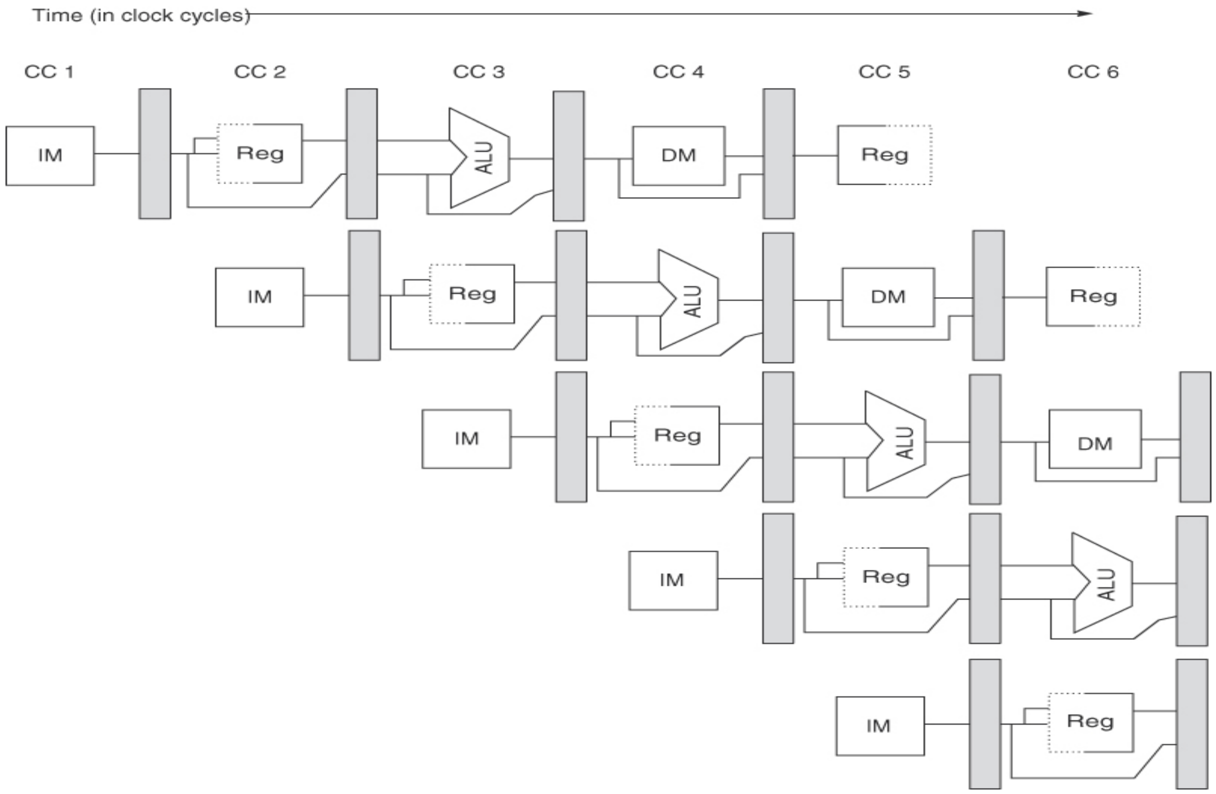
- 1. Unified memory for instruction and data

lw $1,0($2)

lw $3, 0($20)

sub $6, $4, $5

add $7, $1, $0



Structural Hazards

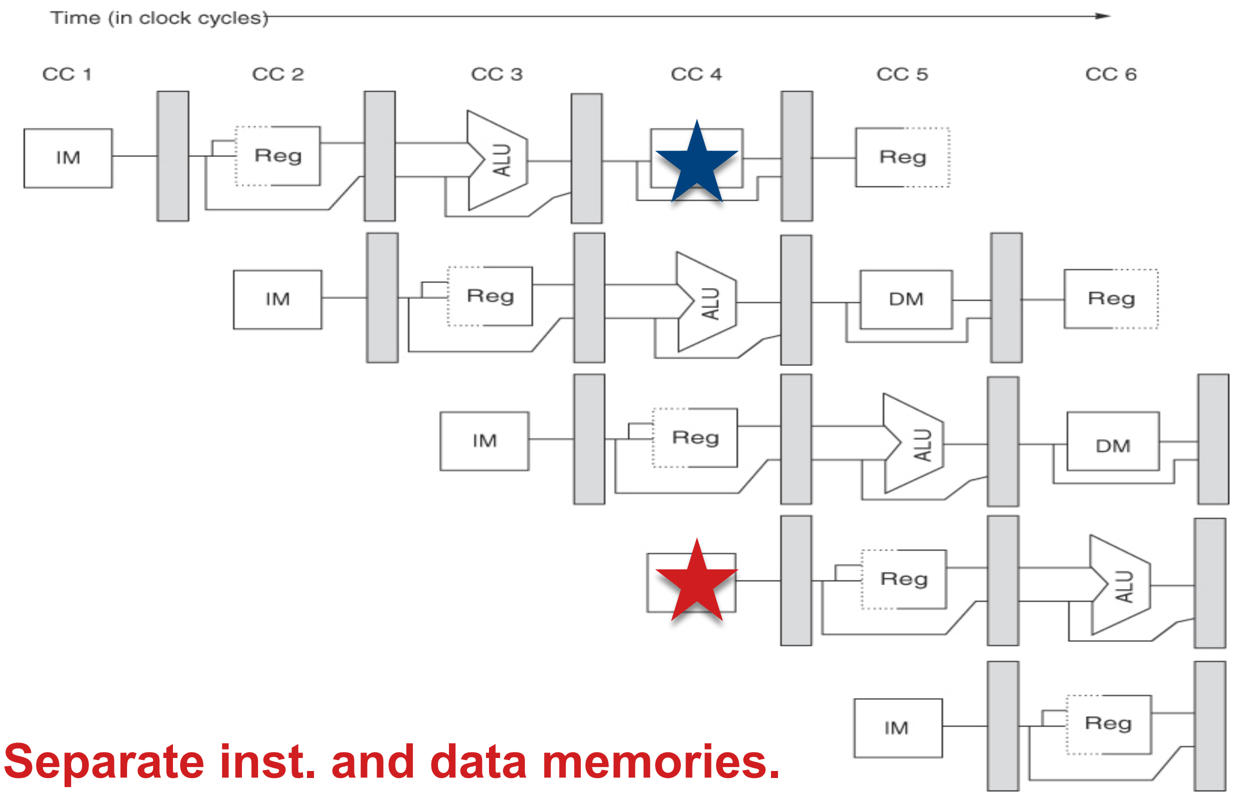
- 1. Unified memory for instruction and data

lw $1,0($2)

lw $3, 0($20)

sub $6, $4, $5

add $7, $1, $0



Separate inst. and data memories.

Structural Hazards

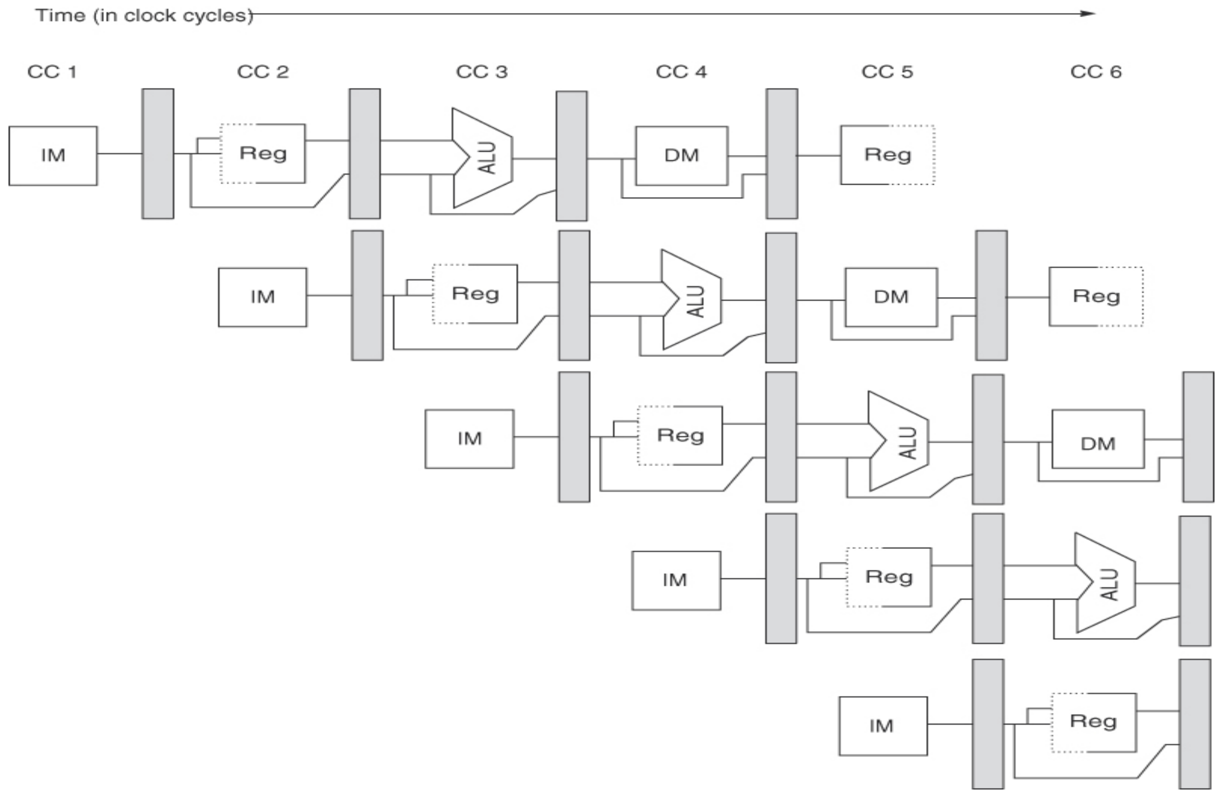
- 1. Unified memory for instruction and data
- 2. Register file with shared read/write access ports

lw $1,0($2)

lw $3, 0($20)

sub $6, $4, $5

add $7, $1, $0



Structural Hazards

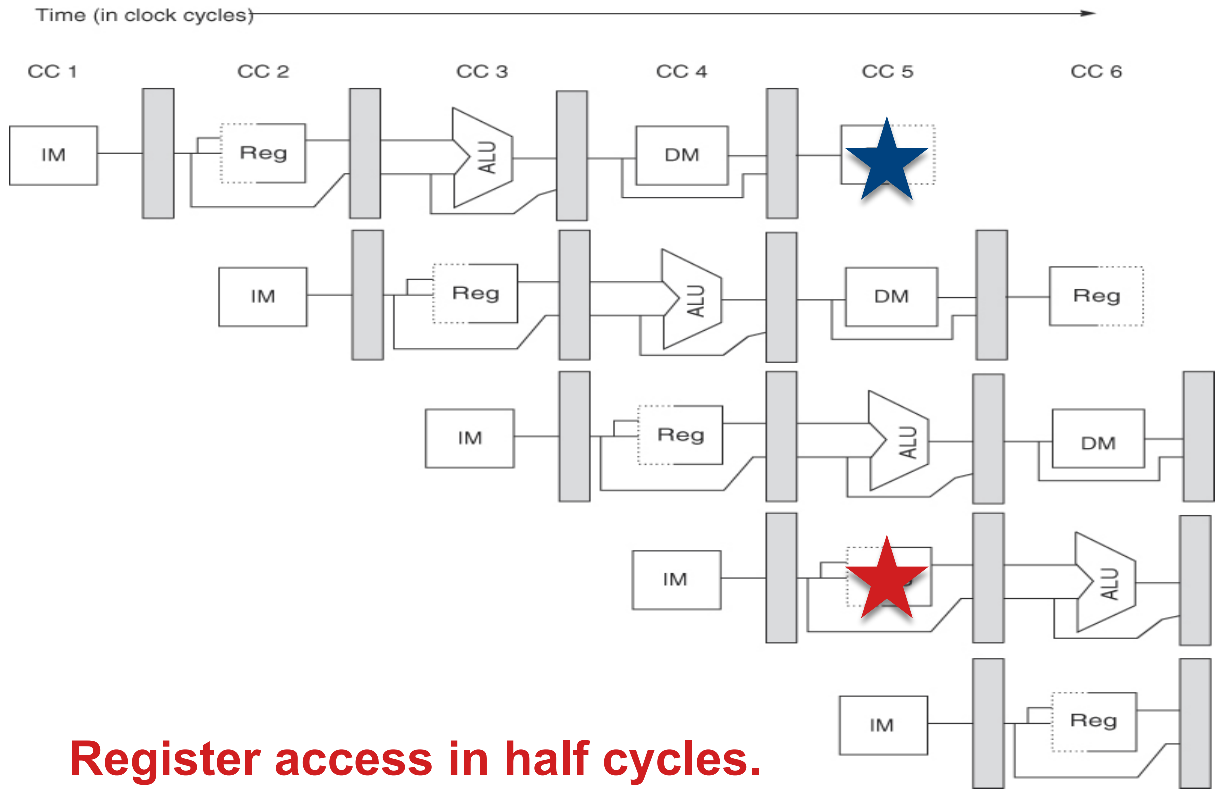
- 1. Unified memory for instruction and data
- 2. Register file with shared read/write access ports

lw $1,0($2)

lw $3, 0($20)

sub $6, $4, $5

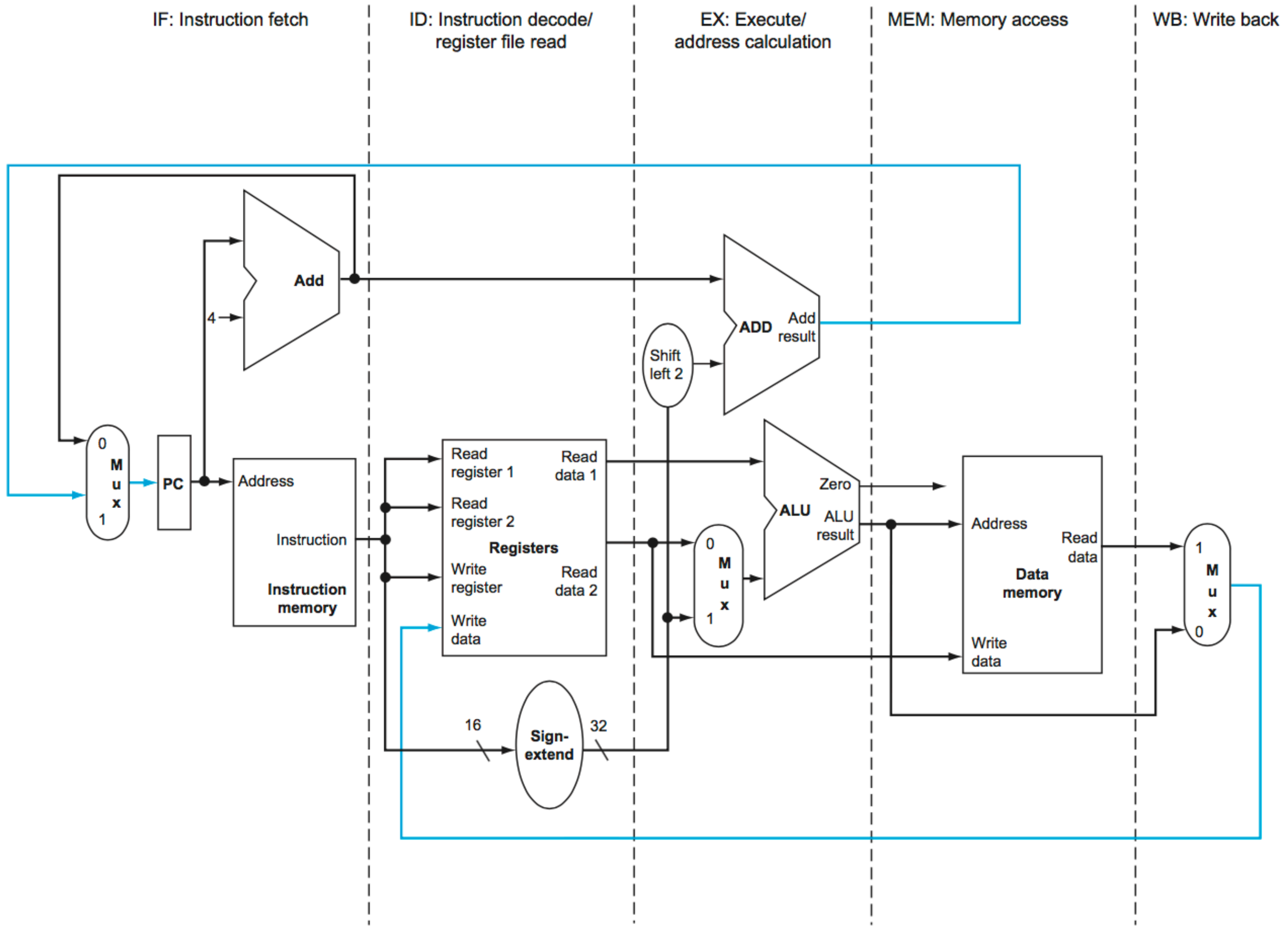
add $7, $1, $0



Register access in half cycles.

Data Hazards

- Solution: register read and write in half cycles



Data Hazards

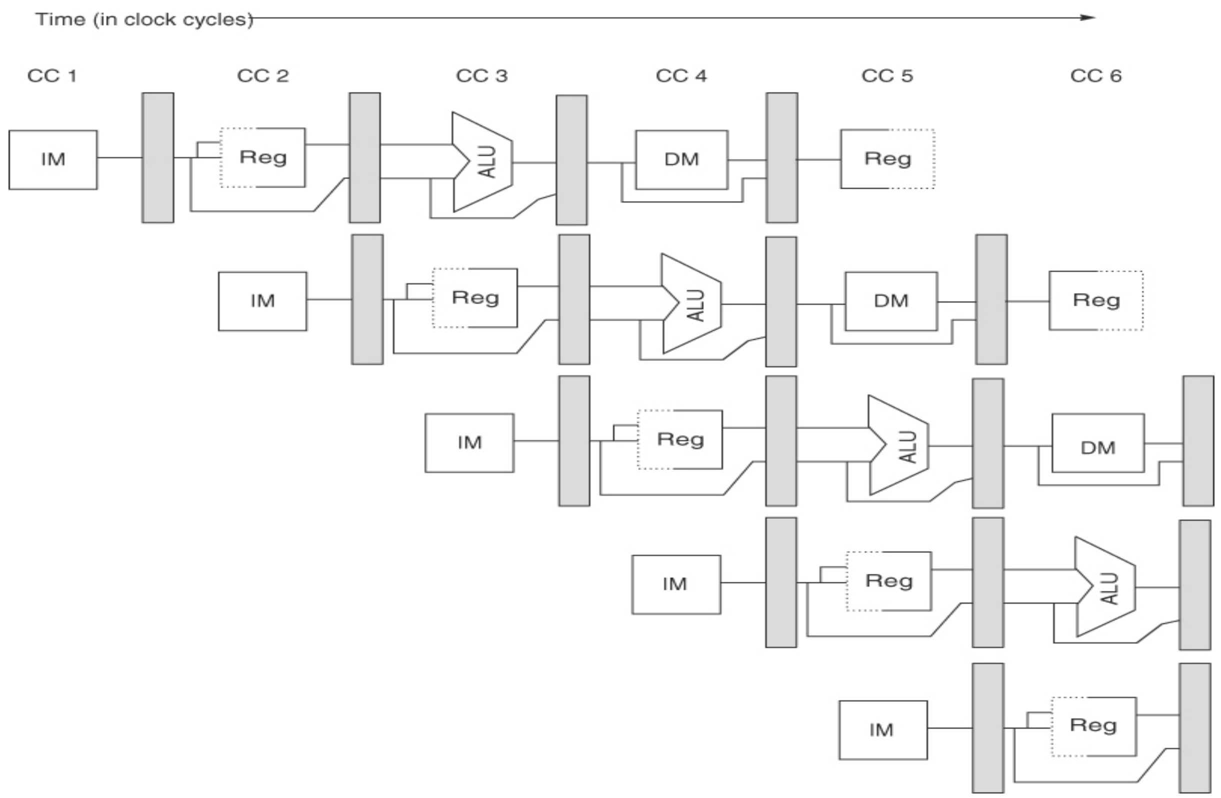
- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

Loading data from memory.

lw \$1, 0(\$2)

add \$3, \$1, \$0

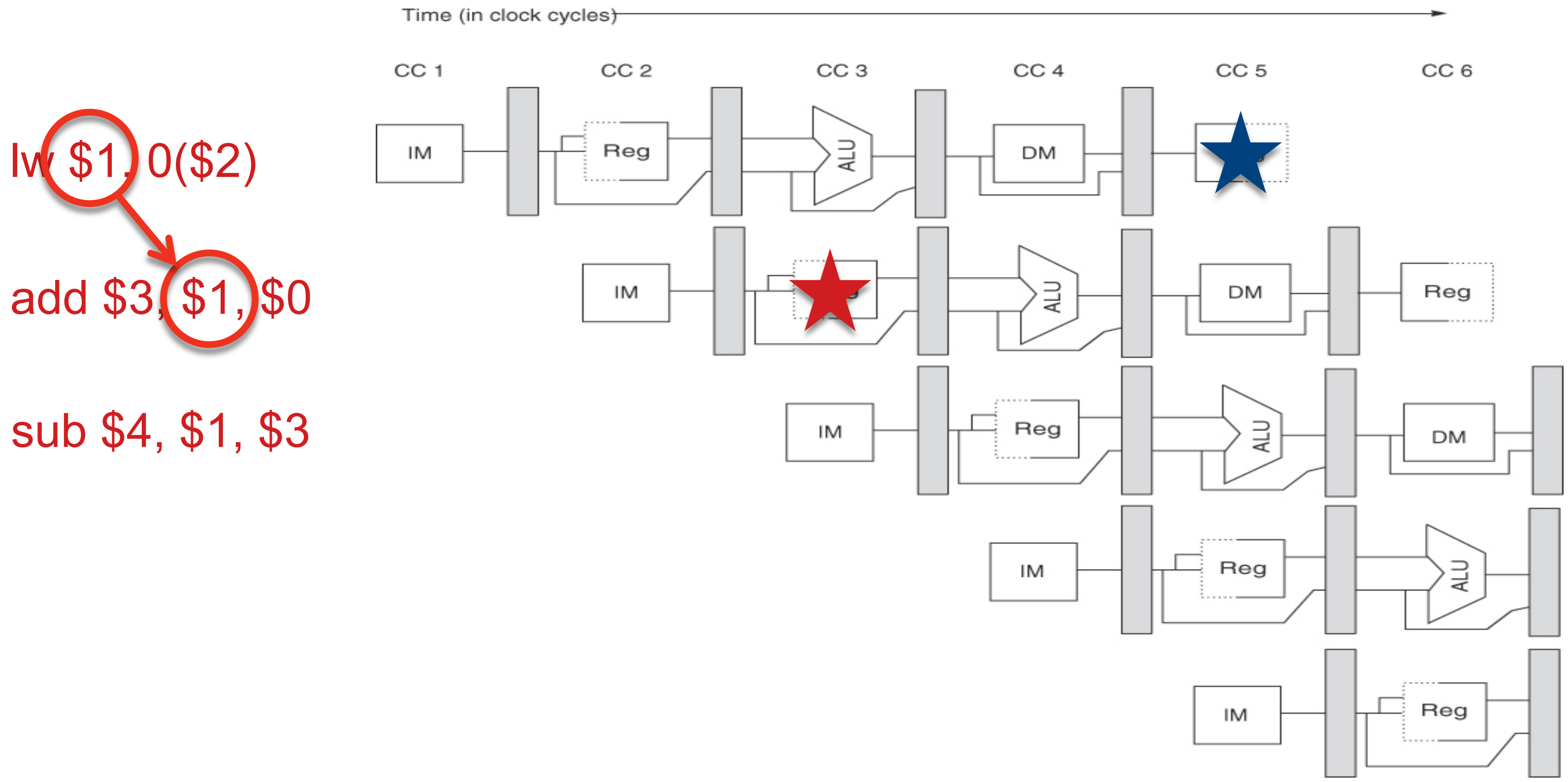
sub \$4, \$1, \$3



Data Hazards

- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

Loaded data will be available two cycles later.



Data Hazards

- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

Inserting two bubbles.

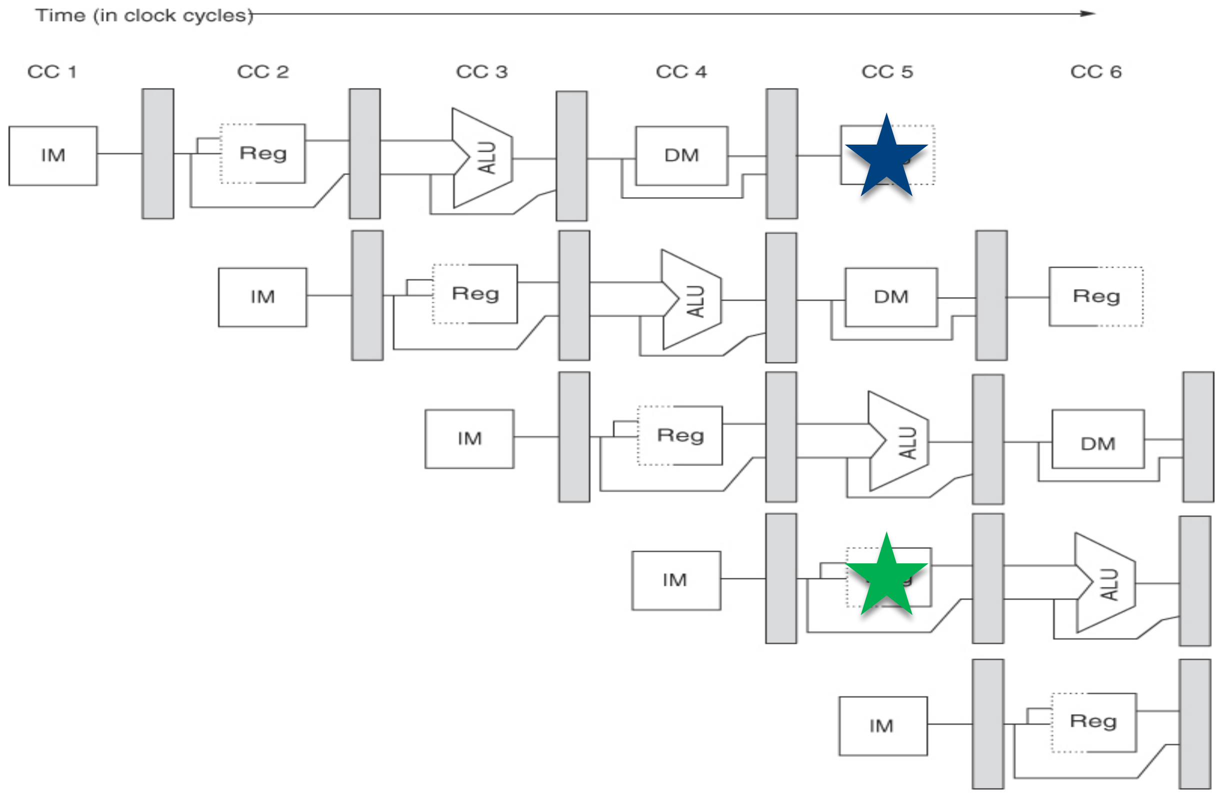
lw $1, 0($2)

Nothing

Nothing

add $3, $1, $0

sub $4, $1, $3



Data Hazards

- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

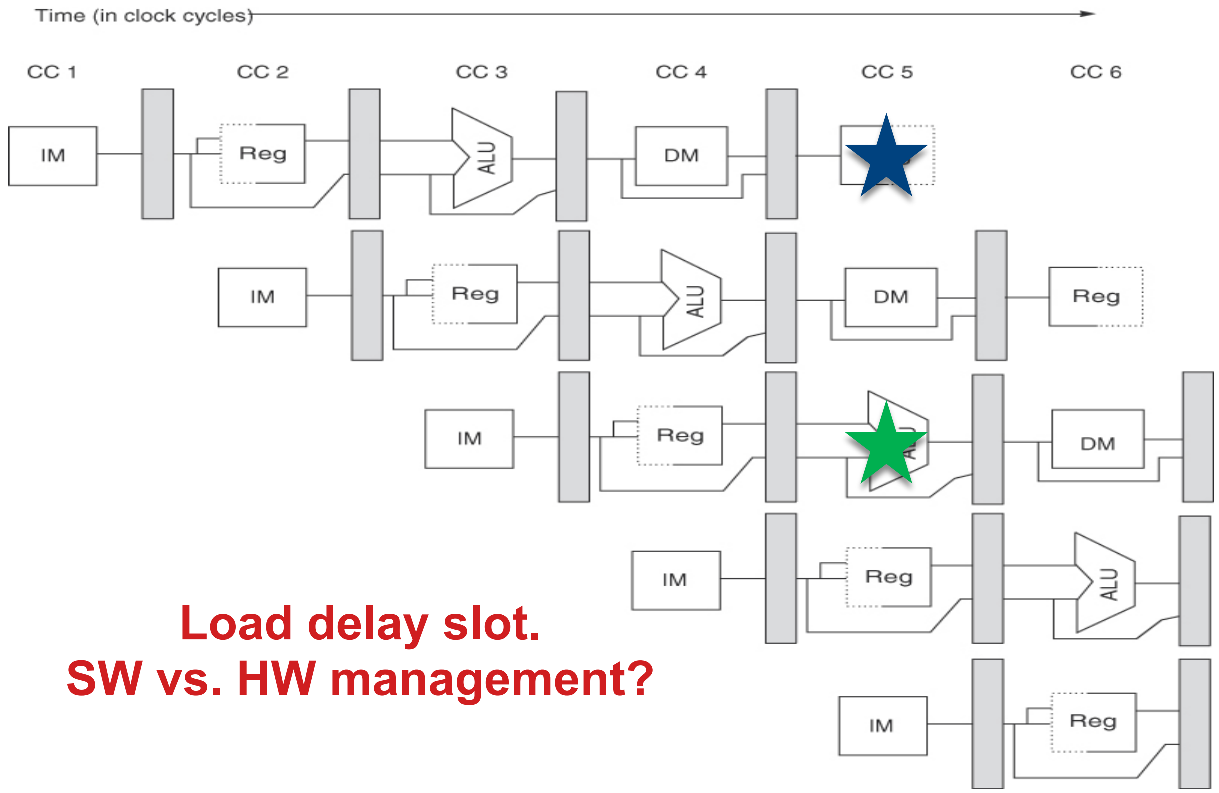
Inserting single bubble + RF bypassing.

lw $1, 0($2)

Nothing

add $3, $1, $0

sub $4, $1, $3



Load delay slot.
SW vs. HW management?

Data Hazards

- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

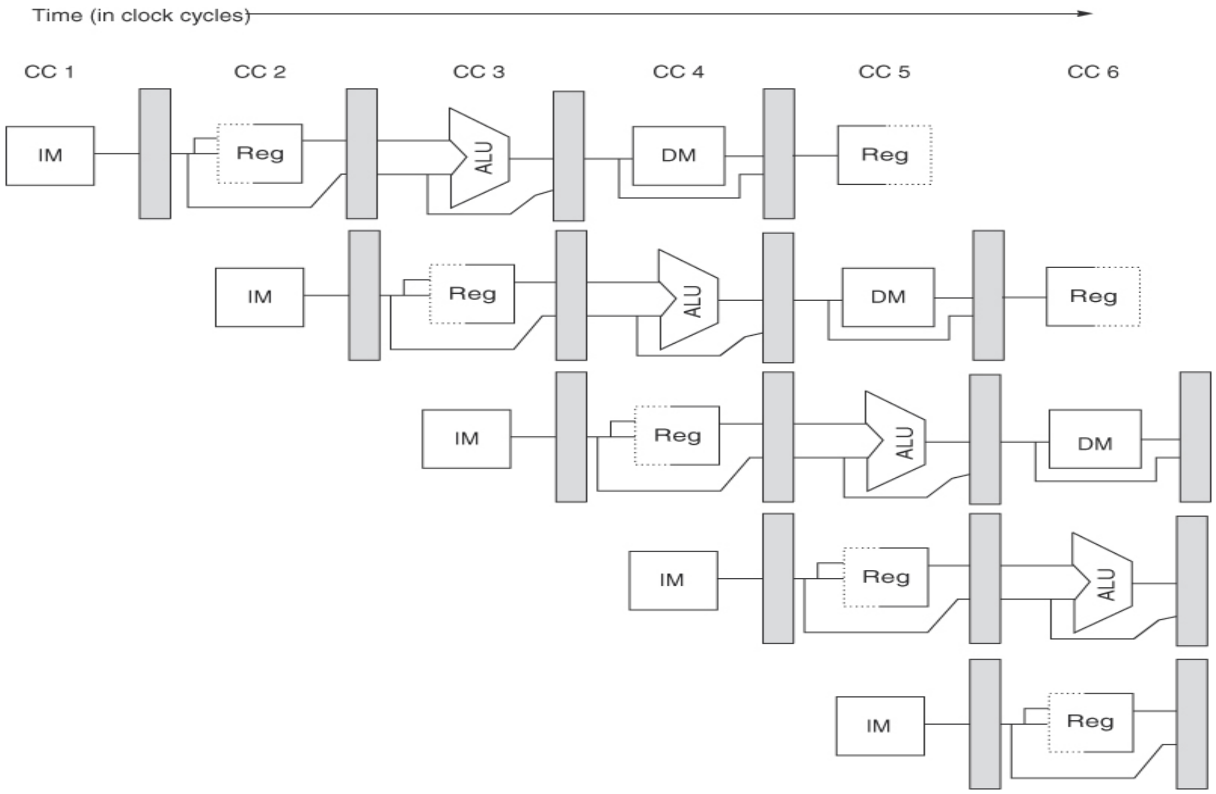
Using the result of an ALU instruction.

add $1,$2,$3

add $5,$1,$0

add $3,$1,$0

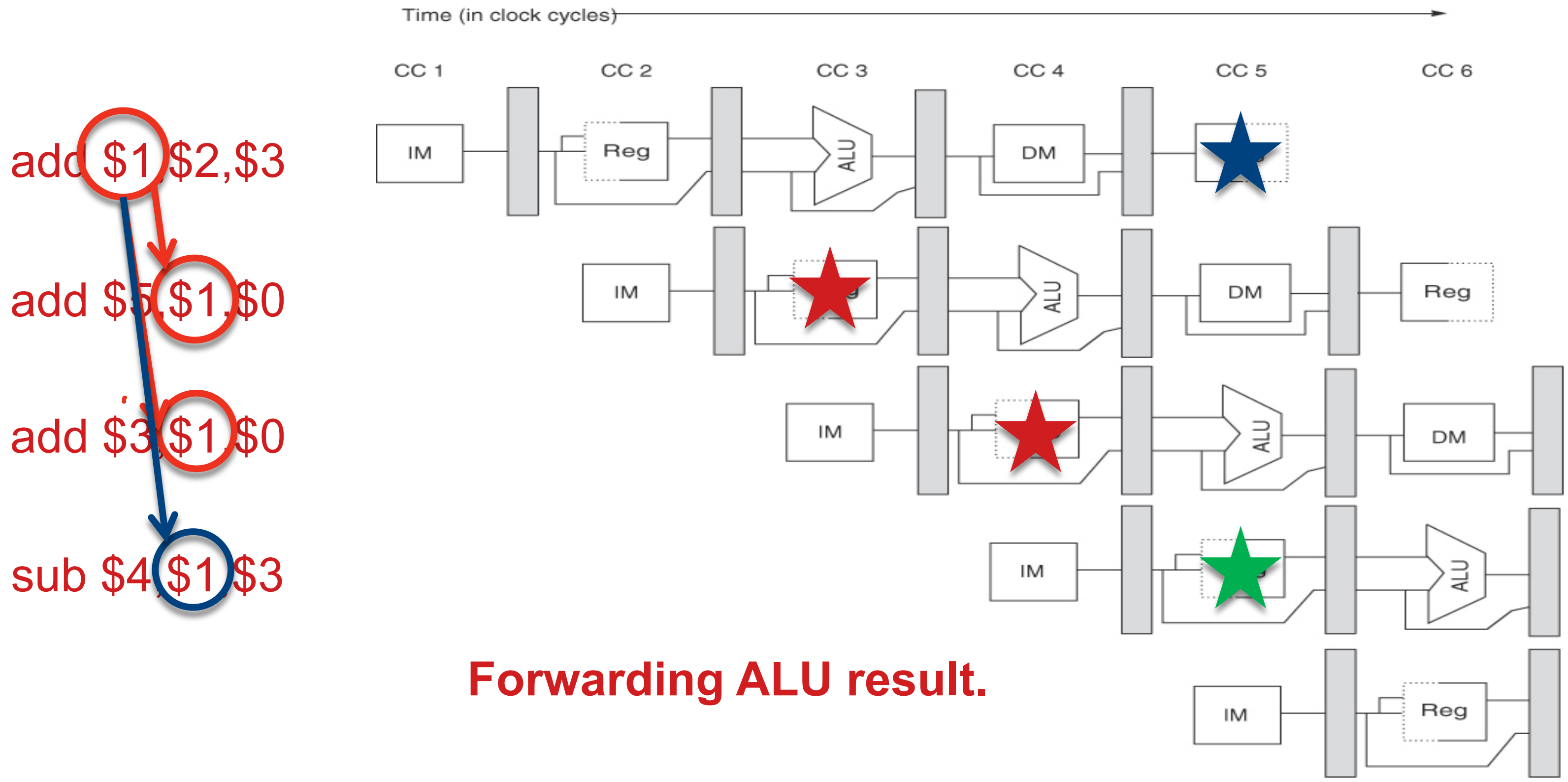
sub $4,$1,$3



Data Hazards

- True dependence: read-after-write (RAW)
  - Consumer has to wait for producer

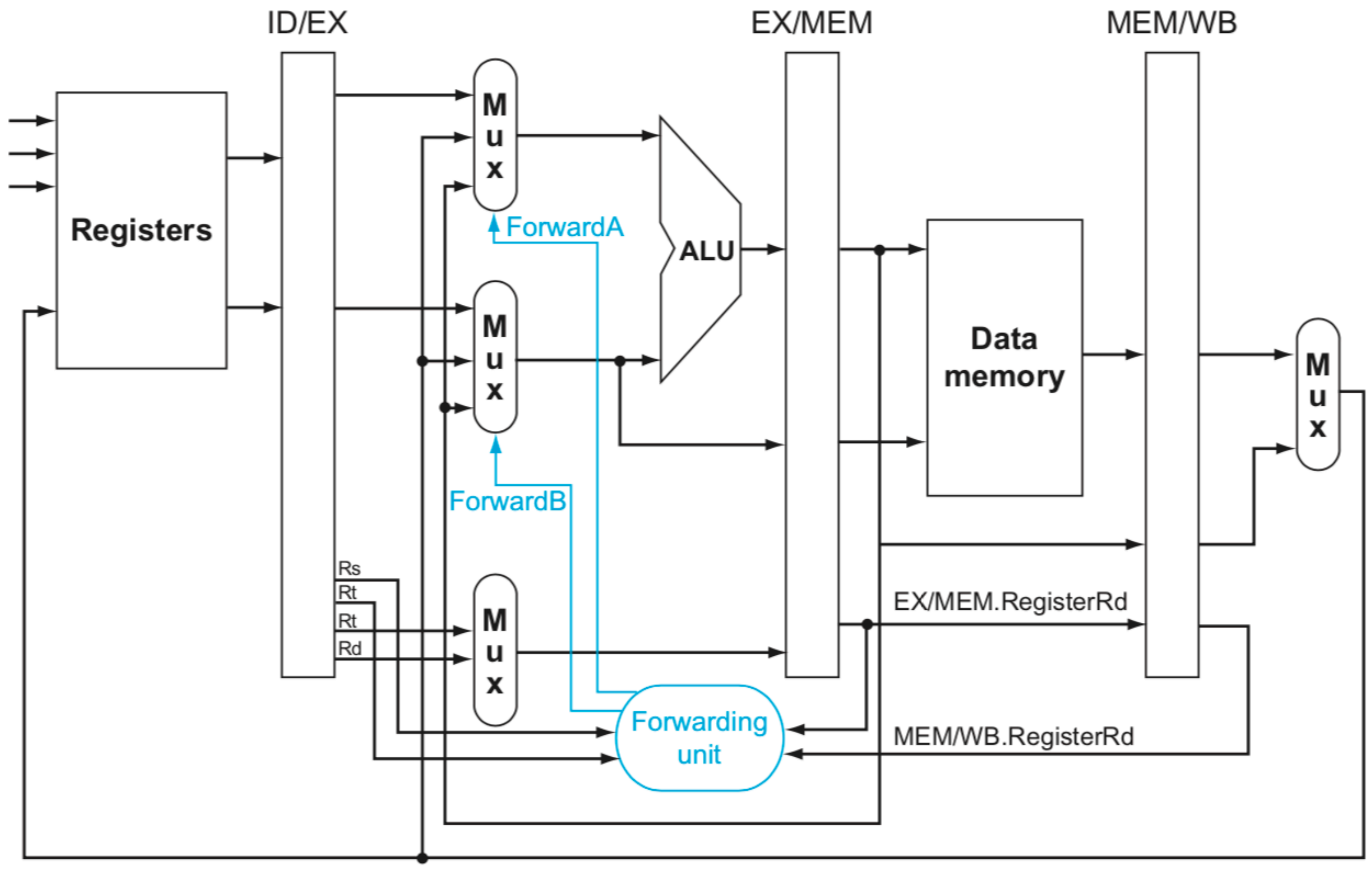
Using the result of an ALU instruction.



Forwarding ALU result.

Data Hazards

- Forwarding with additional hardware



Data Hazards

- How to detect and resolve data hazards
  - Show all of the data hazards in the code below

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

add $2, $1, $0

sub $1, $1, $2

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

Data Hazards

- How to detect and resolve data hazards
  - Show all of the data hazards in the code below

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

Data Hazards

- How to detect and resolve data hazards
  - Show all of the data hazards in the code below

WAW

```
$1← Mem[$2]

$2← $1+$0

$1← $1-$2

Mem[$3] ← $2
```

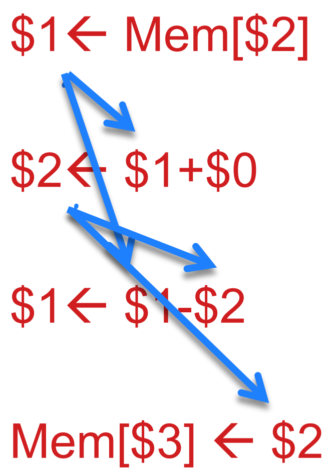
Data Hazards

- How to detect and resolve data hazards
  - Show all of the data hazards in the code below

$$\begin{tikzcd}
\$1 \leftarrow \mathrm{Mem}[\$2] \arrow[d] & \textbf{WAR} \\
\$2 \leftarrow \$1+\$0 \arrow[d] & \\
\$1 \leftarrow \$1-\$2 \arrow[loop below] & \\
\mathrm{Mem}[\$3] \leftarrow \$2 &
\end{tikzcd}$$

Data Hazards

- How to detect and resolve data hazards
  - Show all of the data hazards in the code below



$1← Mem[$2]

$2← $1+$0

$1← $1-$2

Mem[$3] ← $2

RAW