

# CSCI 210: Computer Architecture

## Lecture 31: Data Hazards

Stephen Checkoway

Oberlin College

May 9, 2022

Slides from Cynthia Taylor

# Announcements

- Problem Set 10 due Friday
- Lab 8 due Sunday, May 15
- Office Hours tomorrow 13:30–14:30

# Data Hazards in ALU Instructions

- Consider this sequence:

```
sub $2, $1, $3
and $12, $2, $5
or  $13, $6, $2
add $14, $2, $2
sw  $15, 100($2)
```

- We can resolve hazards with forwarding
  - How do we detect when to forward?

# Forwarding

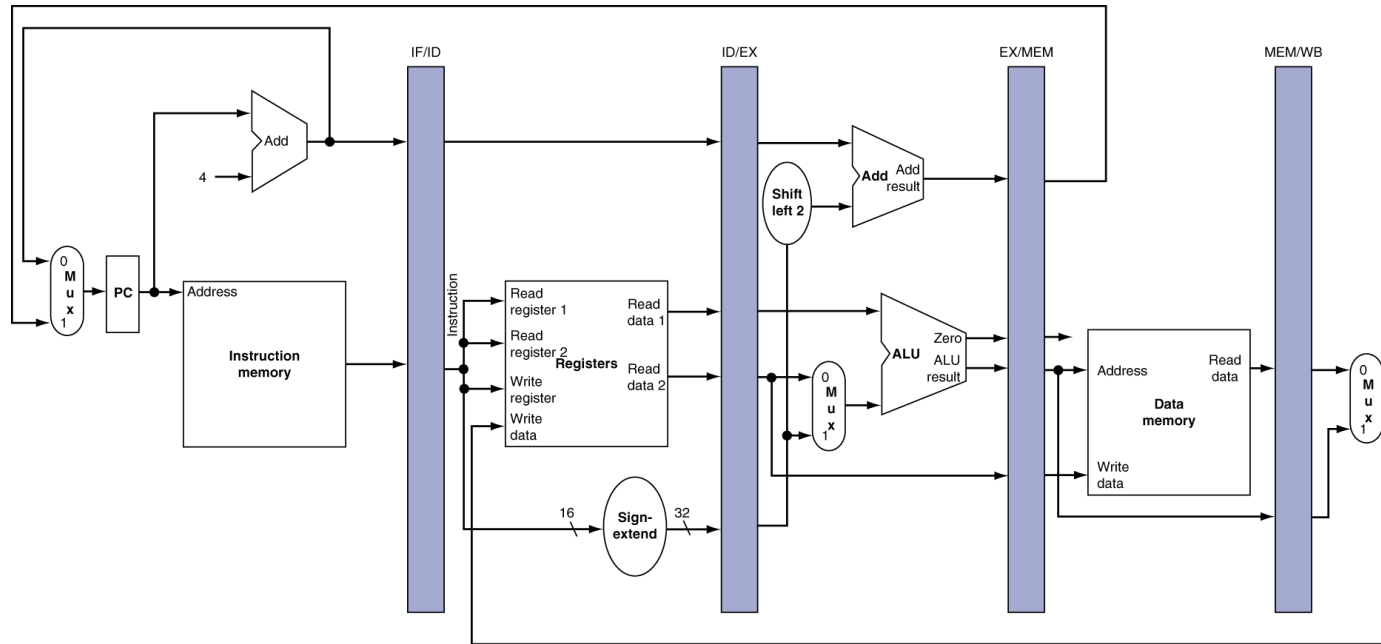
## Datapath

- Connect the outputs of EX and MEM stages to both ALU inputs controlled by muxes

## Control path

- Pass rs, rt, and rd register numbers through the pipeline registers
- Add a forwarding unit to control the muxes
  - Depends on RegWrite and rs/rt/rd from various stages

# Detecting the Need to Forward



- Data hazards when
  - 1a.  $\text{EX/MEM.RegisterRd} = \text{ID/EX.RegisterRs}$
  - 1b.  $\text{EX/MEM.RegisterRd} = \text{ID/EX.RegisterRt}$
  - 2a.  $\text{MEM/WB.RegisterRd} = \text{ID/EX.RegisterRs}$
  - 2b.  $\text{MEM/WB.RegisterRd} = \text{ID/EX.RegisterRt}$

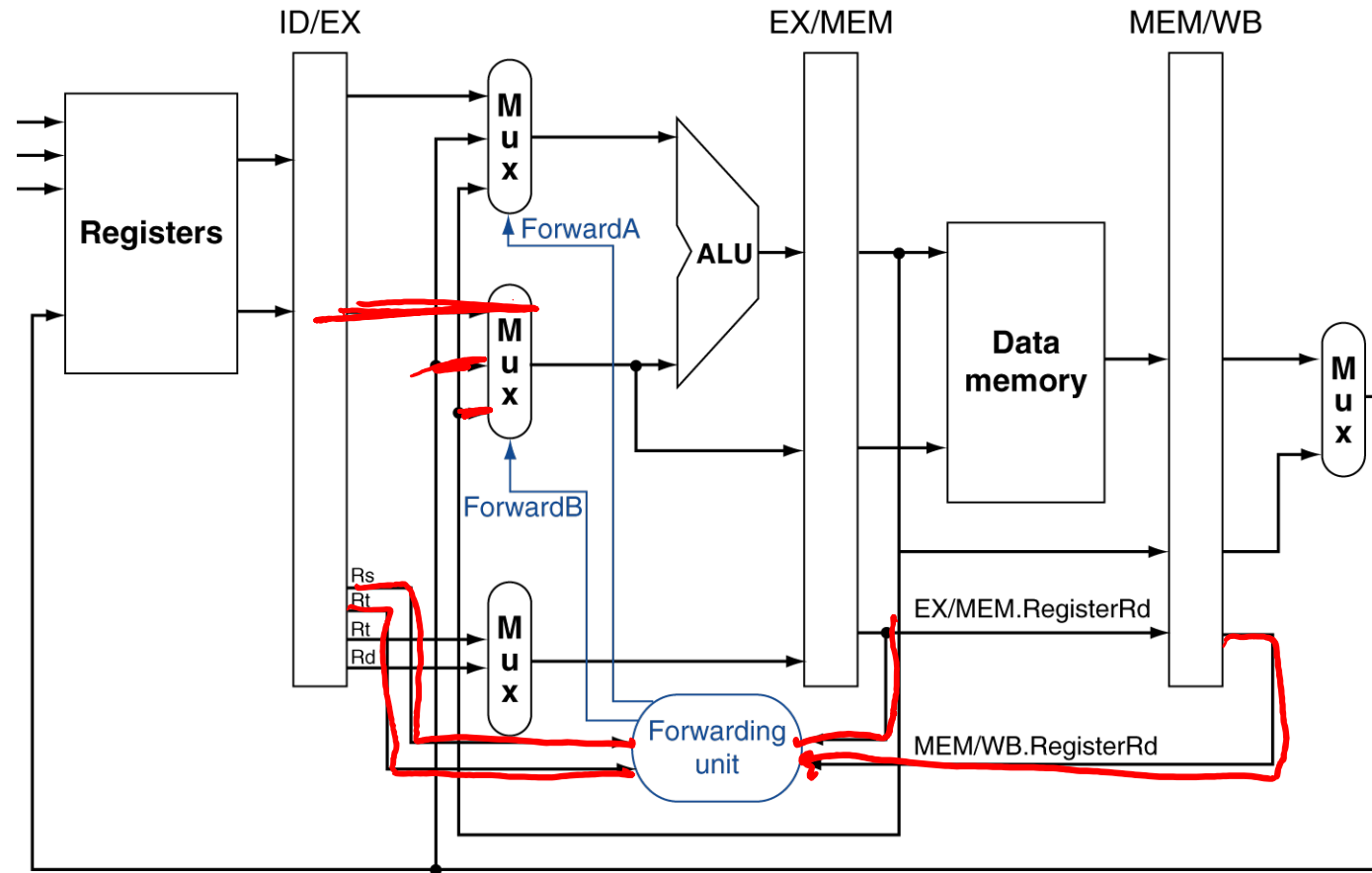
Fwd from  
EX/MEM  
pipeline reg

Fwd from  
MEM/WB  
pipeline reg

# Detecting the Need to Forward

- But only if forwarding instruction will write to a register!
  - EX/MEM.RegWrite, MEM/WB.RegWrite
- And only if Rd for that instruction is not \$zero
  - EX/MEM.RegisterRd  $\neq$  0,  
MEM/WB.RegisterRd  $\neq$  0

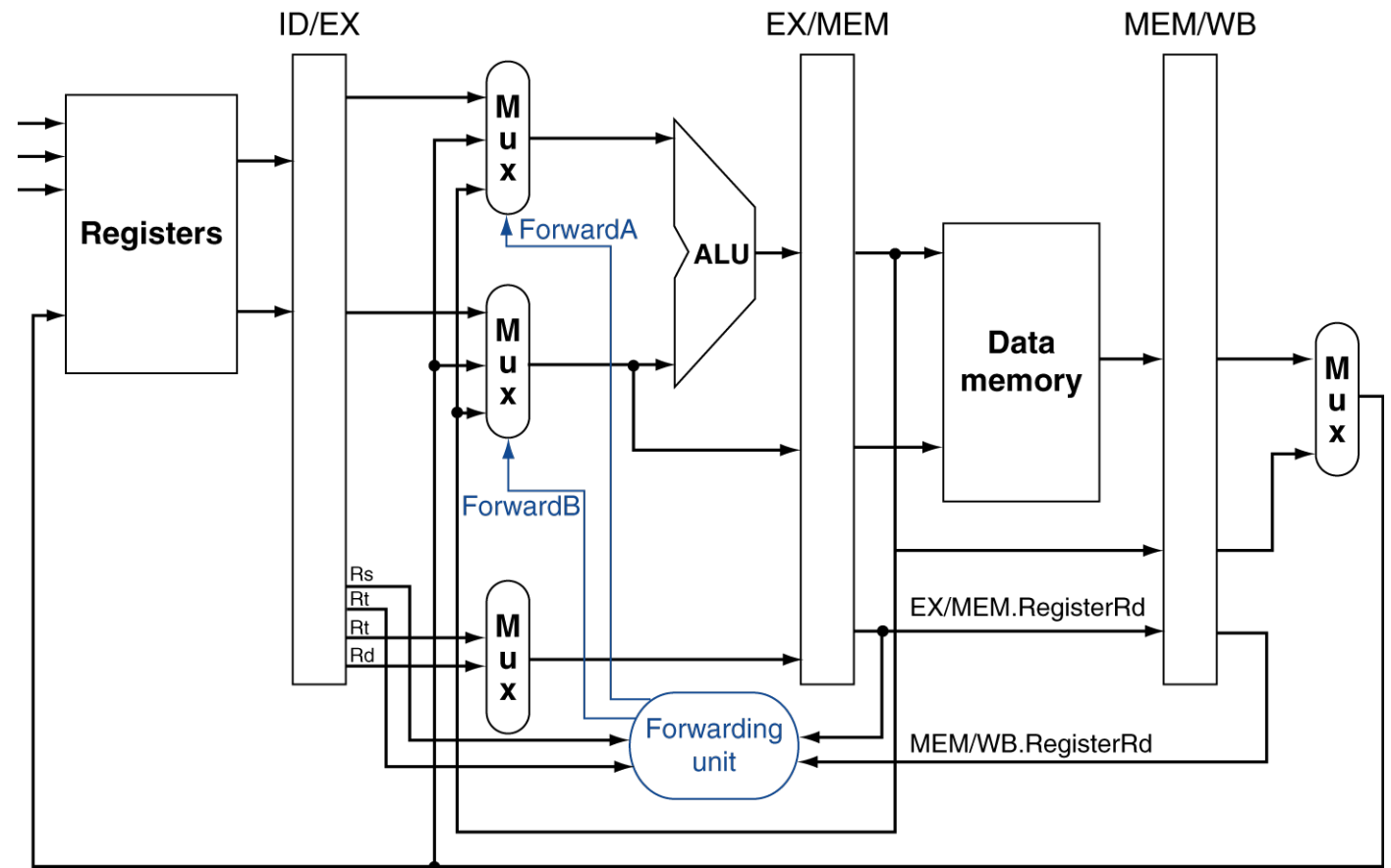
# Forwarding Paths



b. With forwarding

If  $\text{EX/MEM.RegisterRd} = \text{MEM/WB.RegisterRd} = \text{rs}$  (i.e., both pipeline registers contain a value that will be written to the same register that's about to be used for the ALU), which value should be used by the ALU?

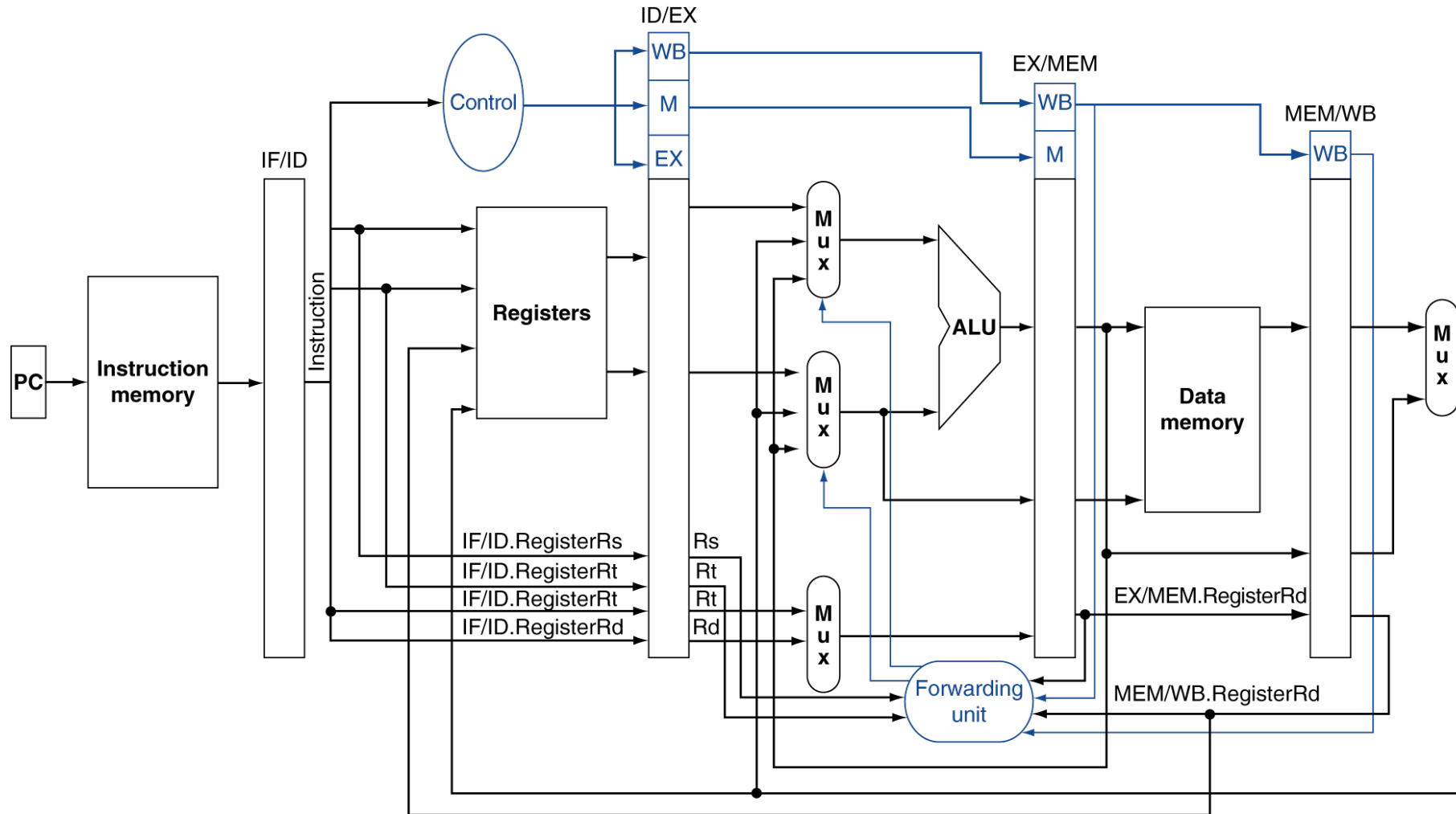
- A. The one in EX/MEM
- B. The one in MEM/WB
- C. Either works since both write to rs
- D. The rs value from the register file



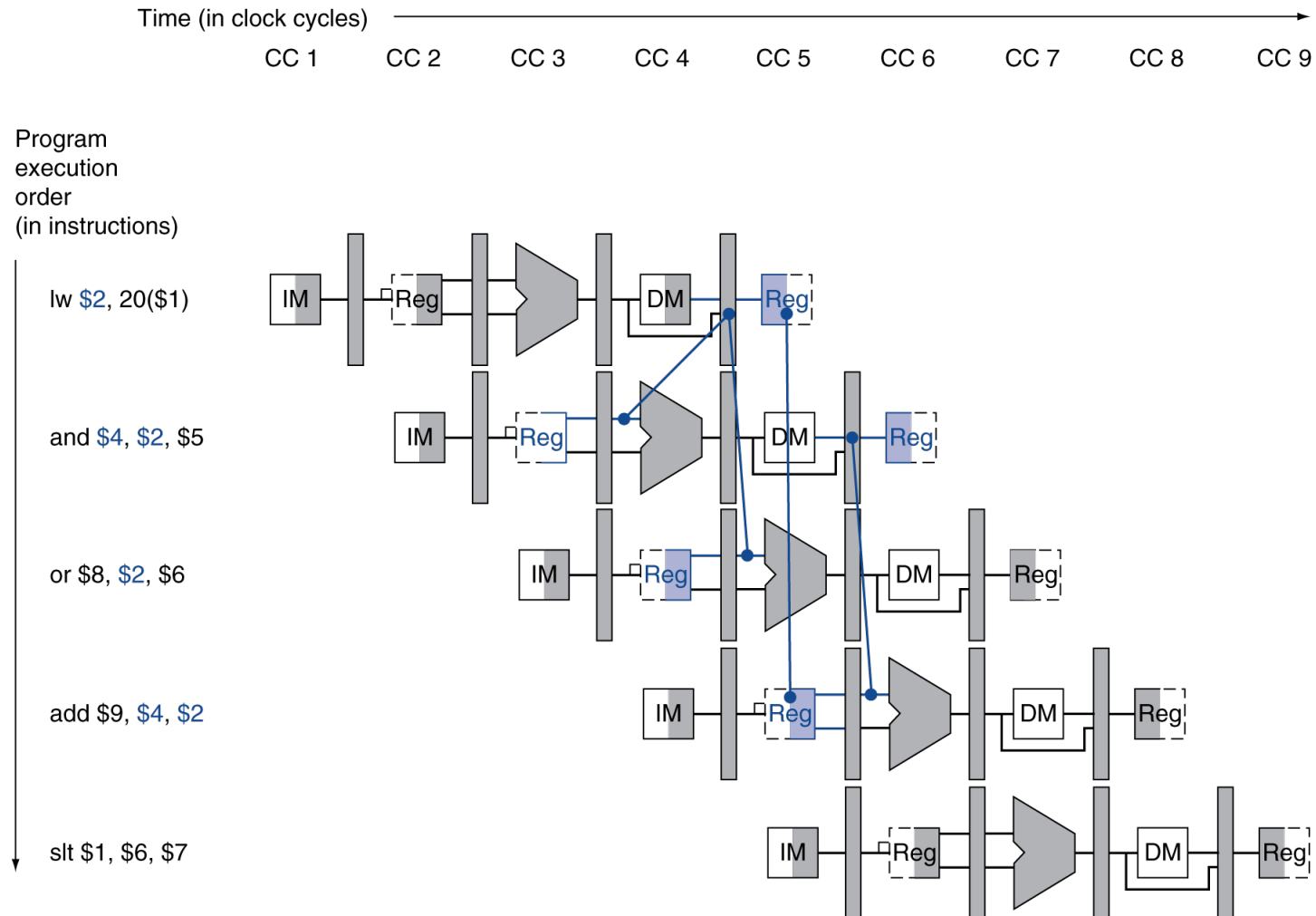
b. With forwarding



# Datapath with Forwarding



# Load-Use Data Hazard



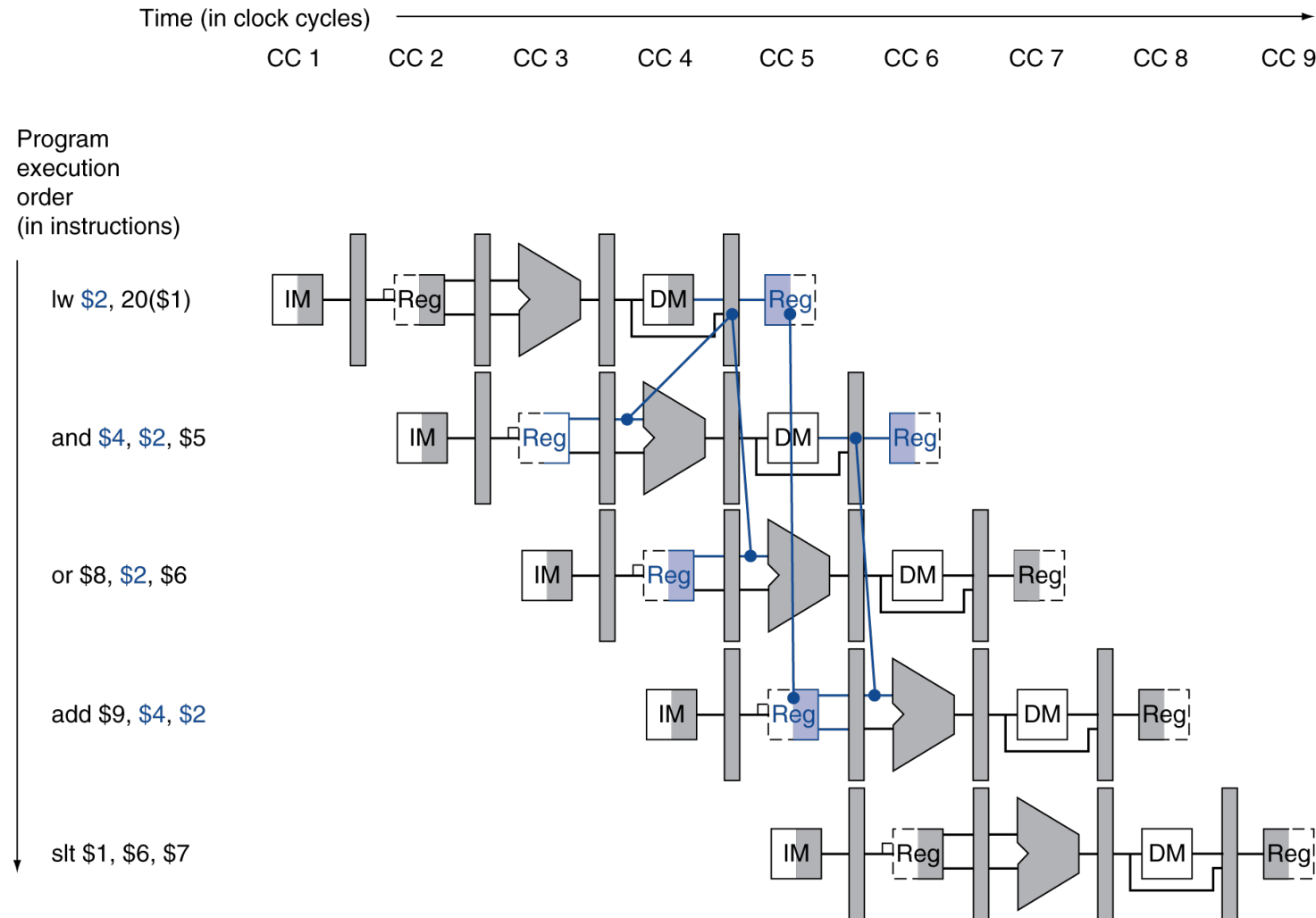
# We can BEST solve these data hazards

A. By stalling.

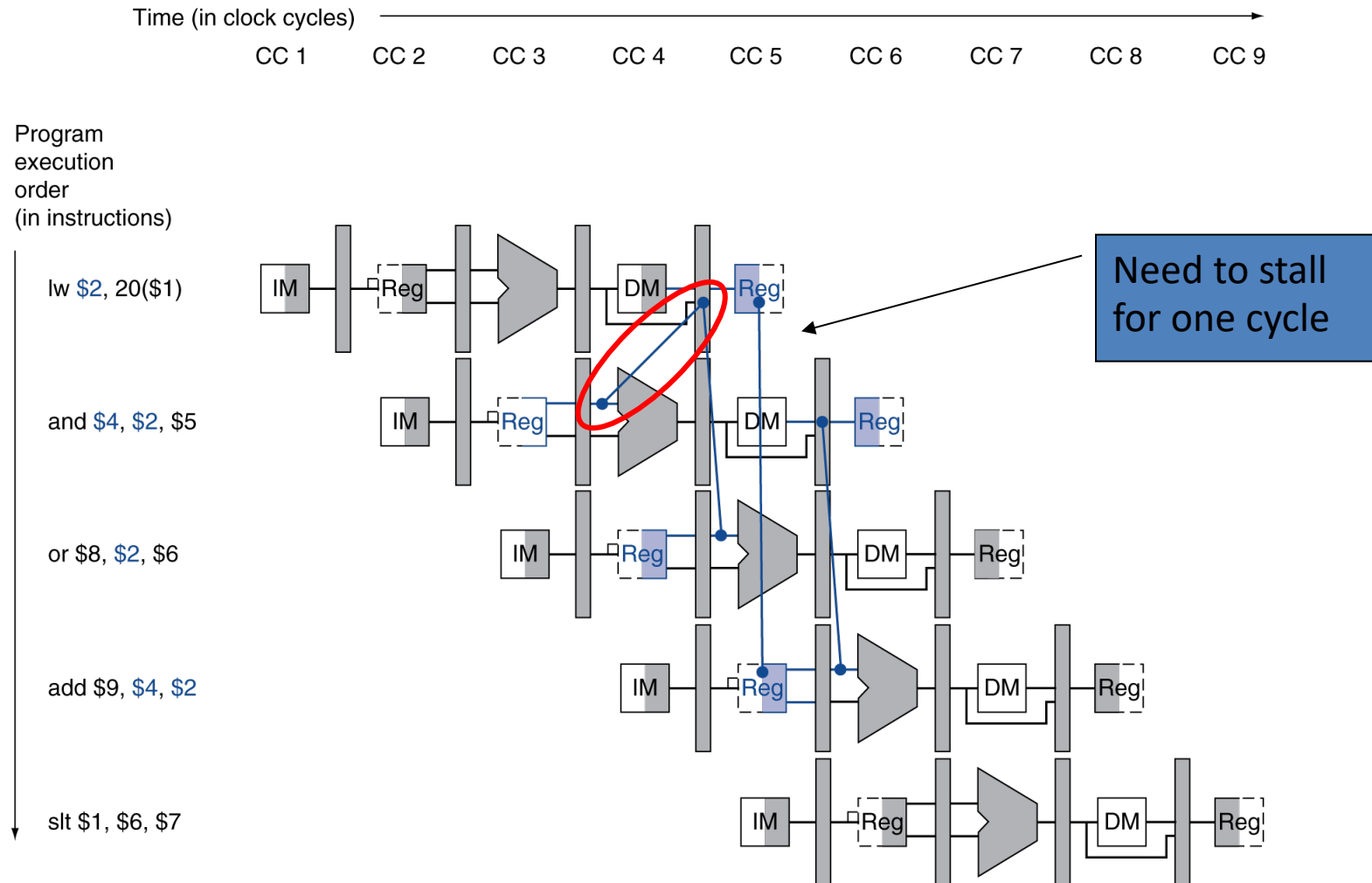
B. By forwarding.

C. By combining forwards and stalls.

D. By doing something else.



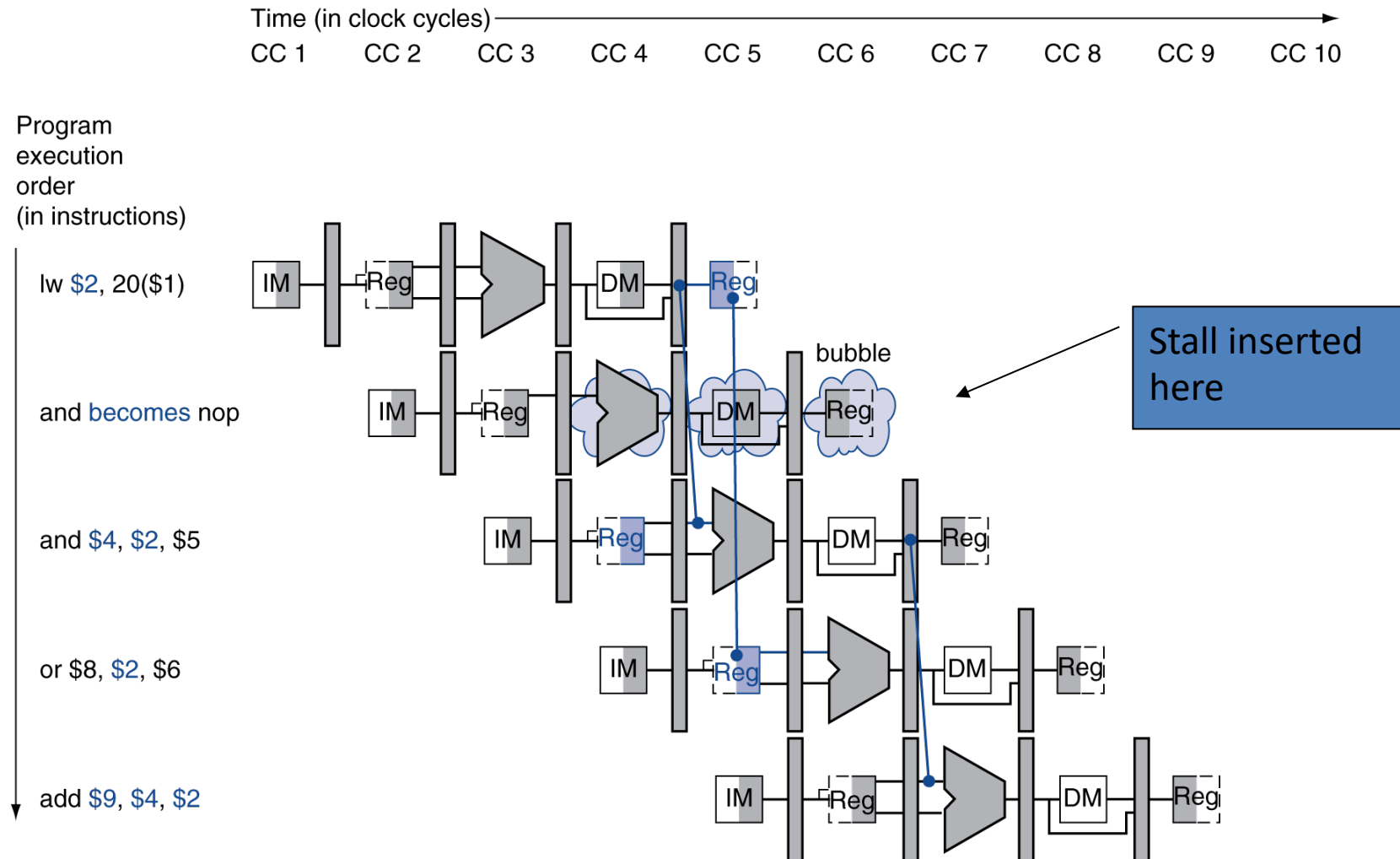
# Load-Use Data Hazard



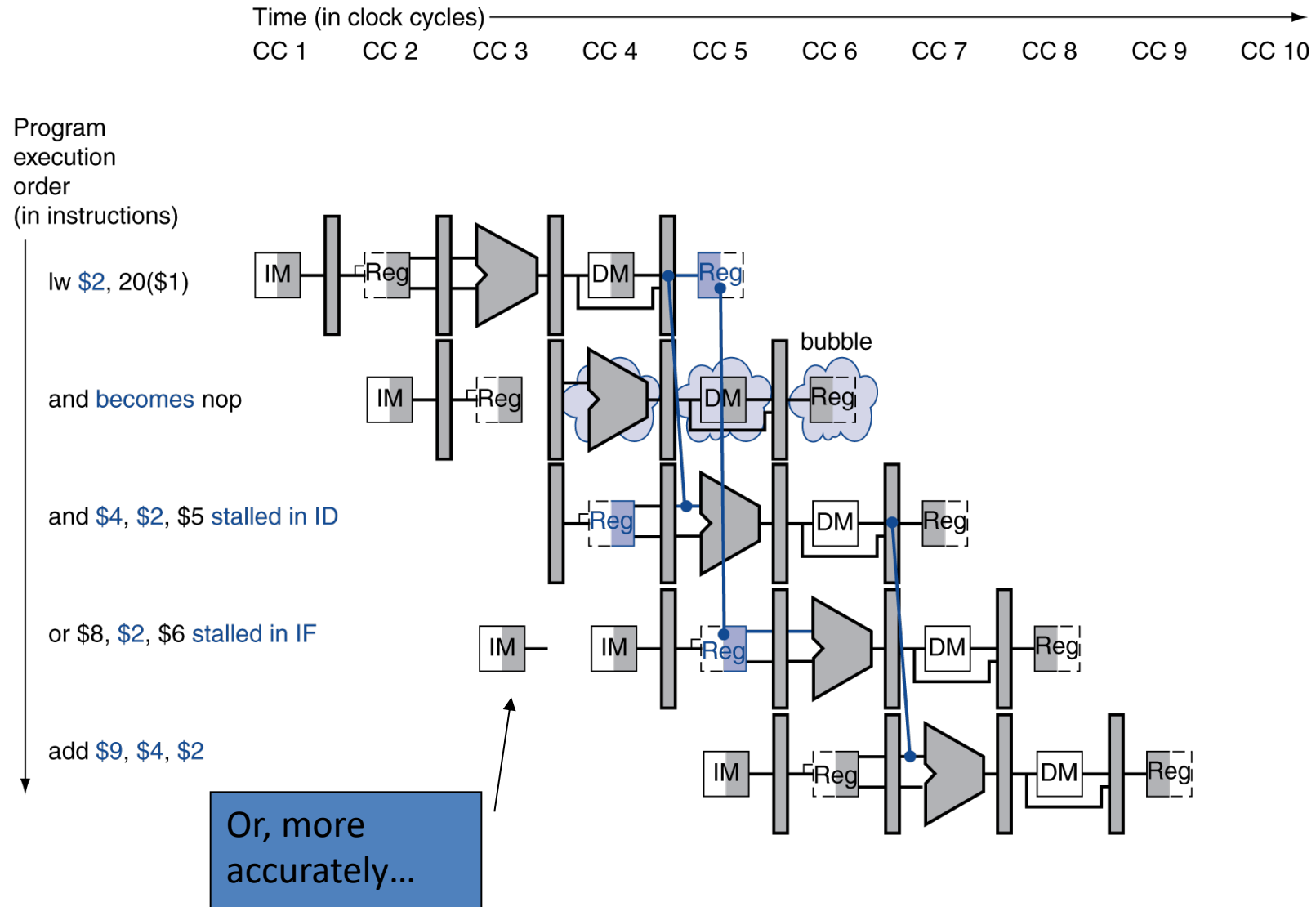
# How to Stall the Pipeline

- Detect hazard in ID stage using Hazard detection unit
  - Check if instruction in EX stage is load with destination rs or rt
- Force control values in ID/EX register to 0
  - EX, MEM and WB do nop (no-operation)
- Prevent update of PC and IF/ID register
  - Instruction with dependency is decoded again
  - Following instruction is fetched again
  - 1-cycle stall allows MEM to read data for  $1w$ 
    - Can subsequently forward to EX stage

# Stall/Bubble in the Pipeline

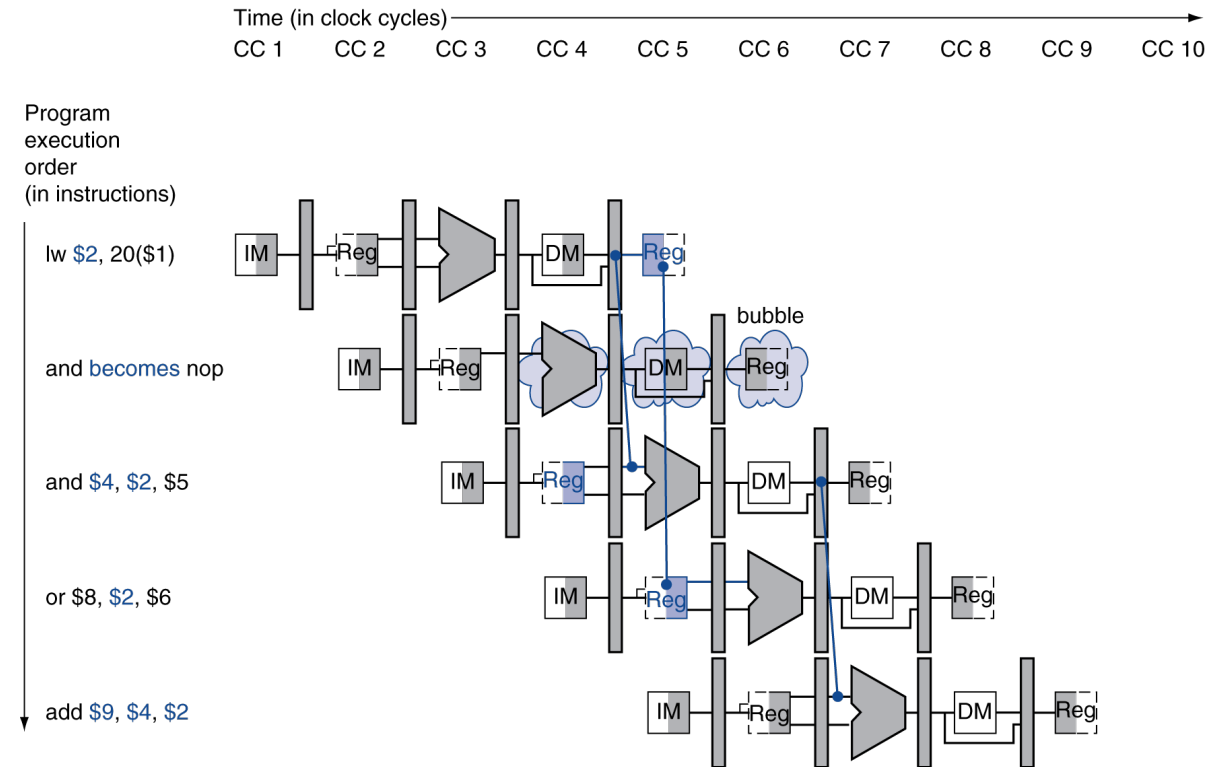


# Stall/Bubble in the Pipeline



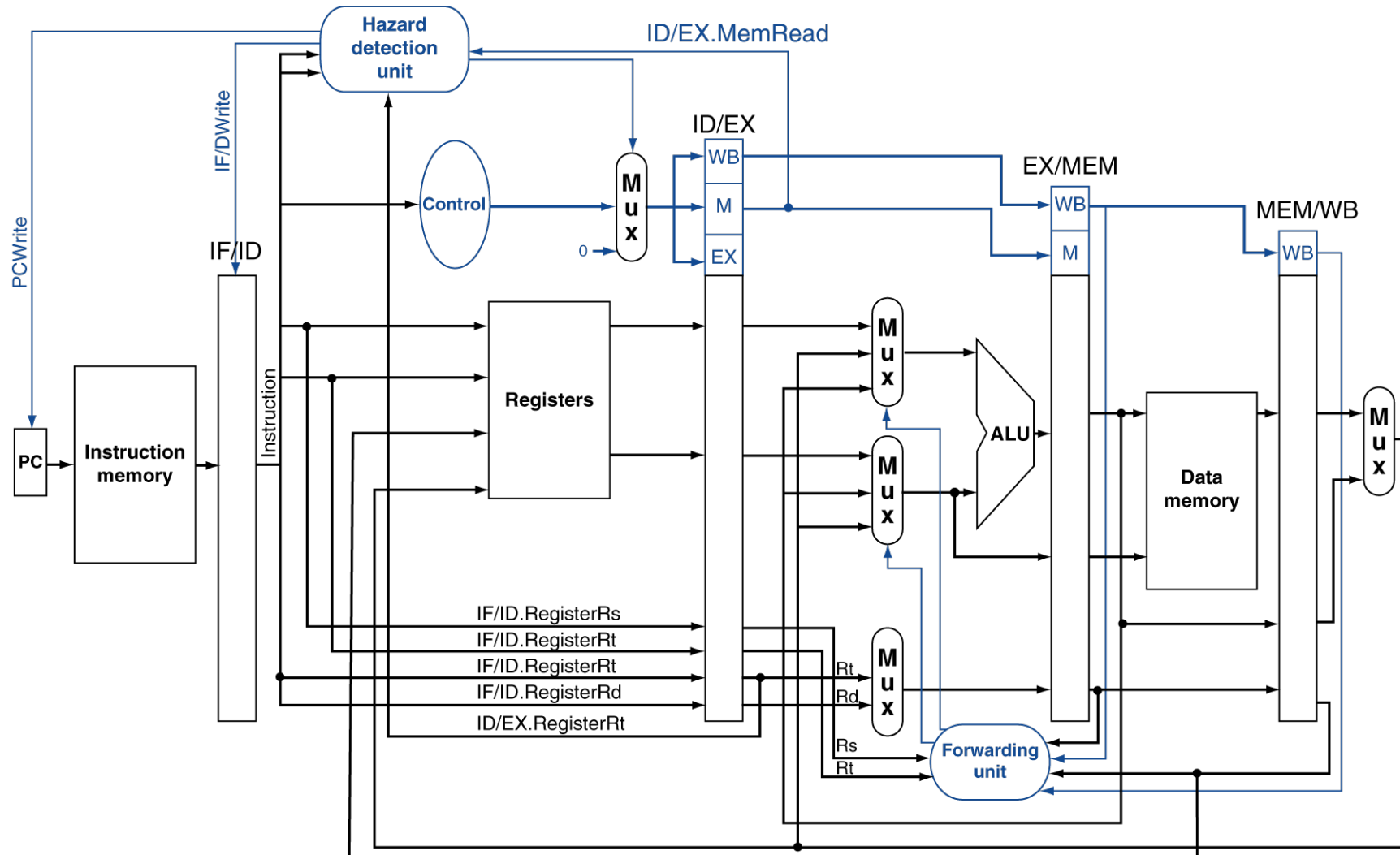
# After we add the stall

- A. Everything works with our existing forwarding
- B. We need to forward between the register files to solve the 2<sup>nd</sup> hazard
- C. We need to do something else





# Datapath with Hazard Detection



Consider the code

```
addi    $s0, $s0, 4  
lw      $t0, 0($s0)  
sub     $t1, $t2, $t2  
add     $t0, $t0, $t1
```

Does this code require a forward, a stall, both, or neither?

- A. Forward
- B. Stall
- C. Both
- D. Neither

# Stalls and Performance

- Stalls reduce performance
  - But are required to get correct results
- Compiler can arrange code to avoid hazards and stalls
  - Requires knowledge of the pipeline structure
  - Different microarchitectures (variants of the same underlying architecture) can have vastly different pipelines
  - Compilers have to pick one to target

# Reading

- Next lecture: Control Hazards
  - Section 5.9
- Problem Set 10 due Friday
- Lab 8 due Monday