CSSE232
Computer Architecture I

Data Hazards

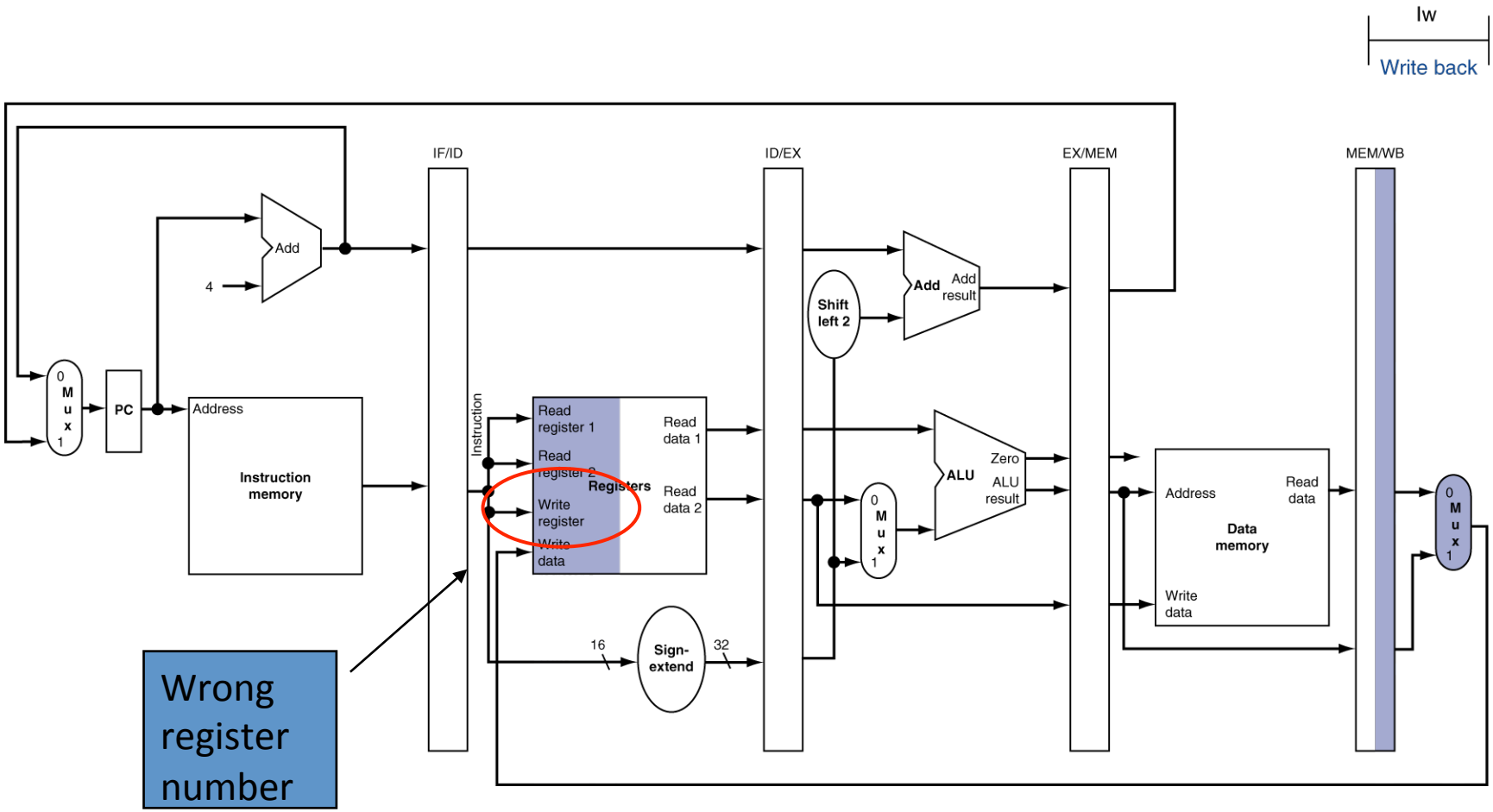
Outline

- Hazards
  - Data
  - Control
  - Structural
- Hazard detection
- Hazard prevention

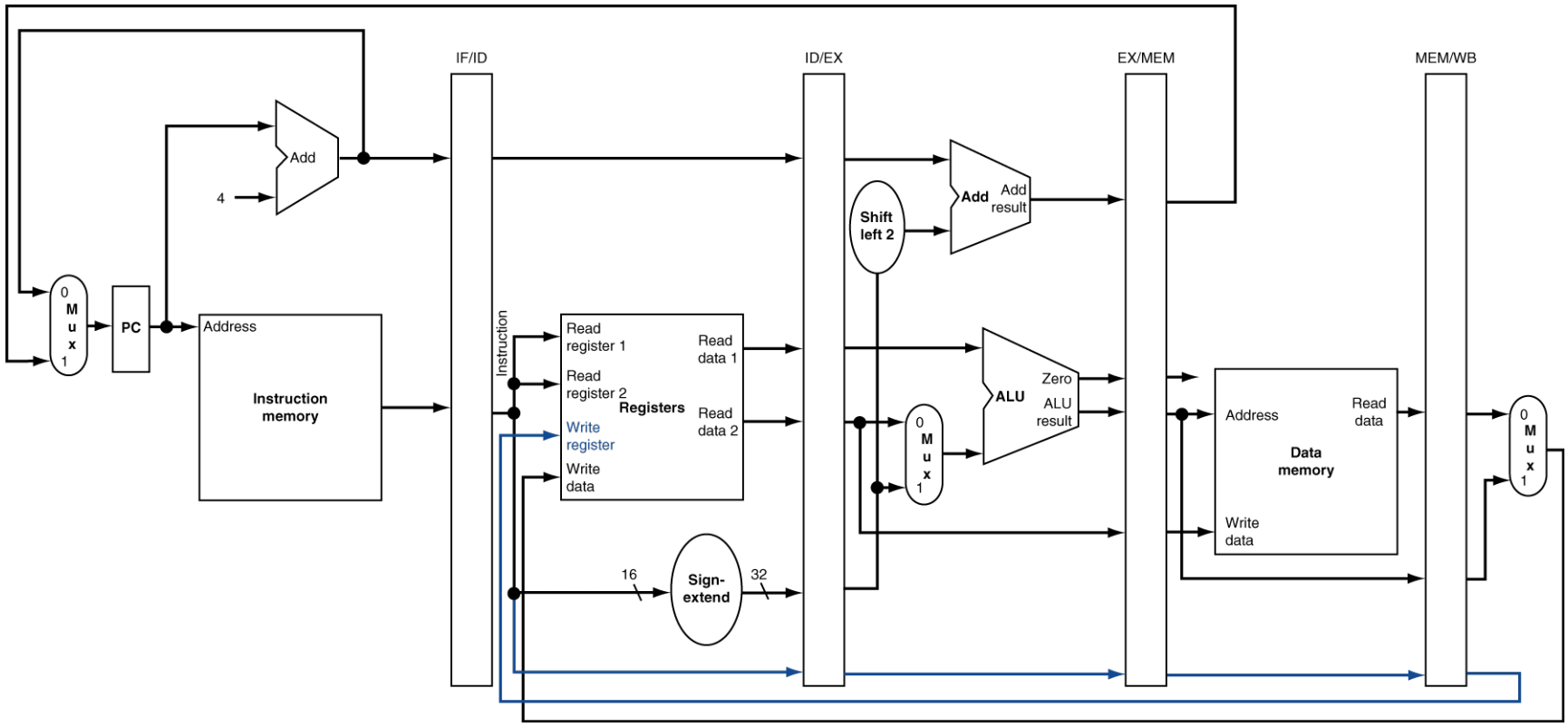
Pipelining

- From last time…

WB for Load

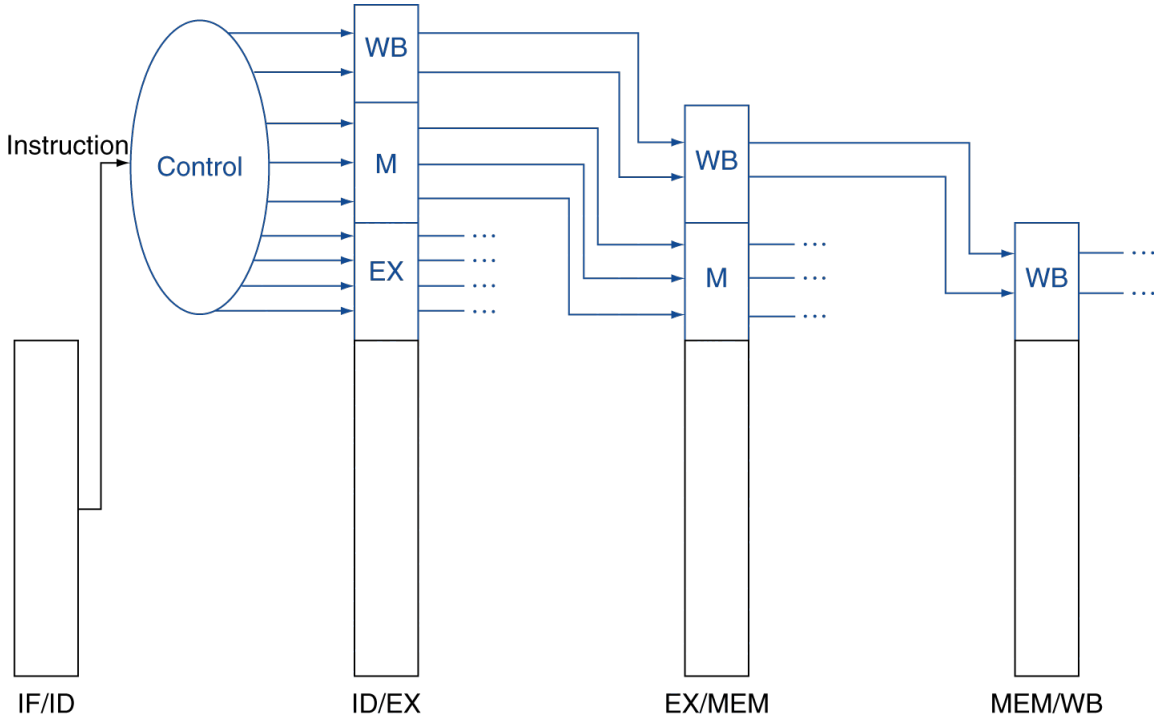

lw
Write back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
16
Sign-extend
32
Wrong register number

Corrected Datapath for Load



Pipelined Control

- Control signals derived from instruction
  - As in single-cycle implementation

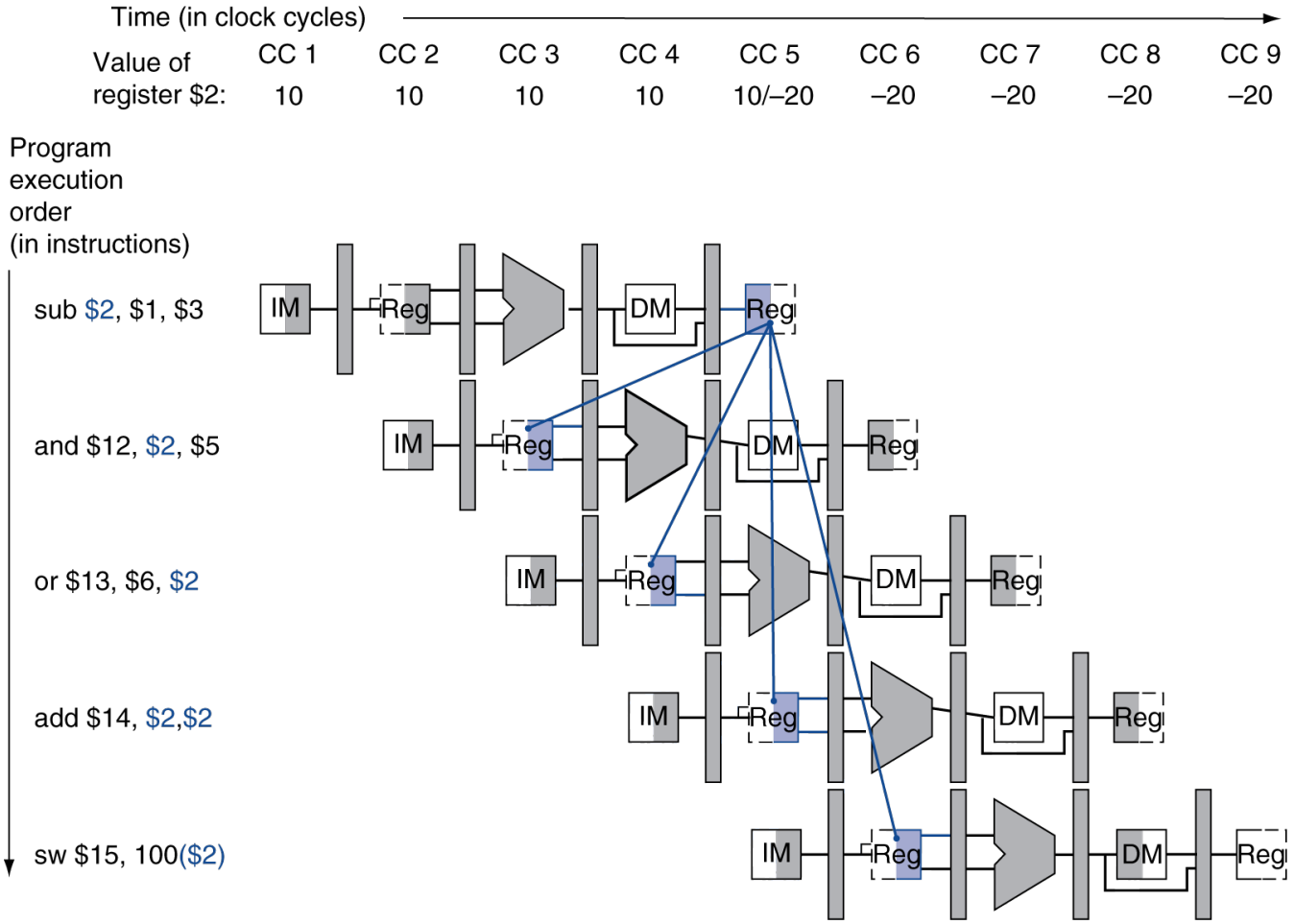


Data Hazards in ALU Instructions

- Consider this sequence:

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

Dependencies



Data Hazards in ALU Instructions

- Potential dependencies:

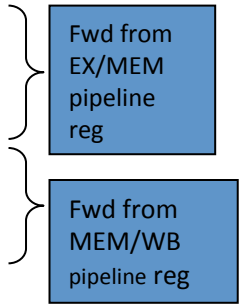
```
sub $2, $1,$3
and $12,$2,$5
or  $13,$6,$2
add $14,$2,$2
sw  $15,100($2)
```

Resolve dependancies with Forwarding

- Dependancy
  - Part of the pipeline needs data
  - Not available in standard pipeline path
  - May be available somewhere in datapath
- Forwarding
  - New path from one pipeline stage to another
  - Send data when dependency detected

Detecting the Need to Forward

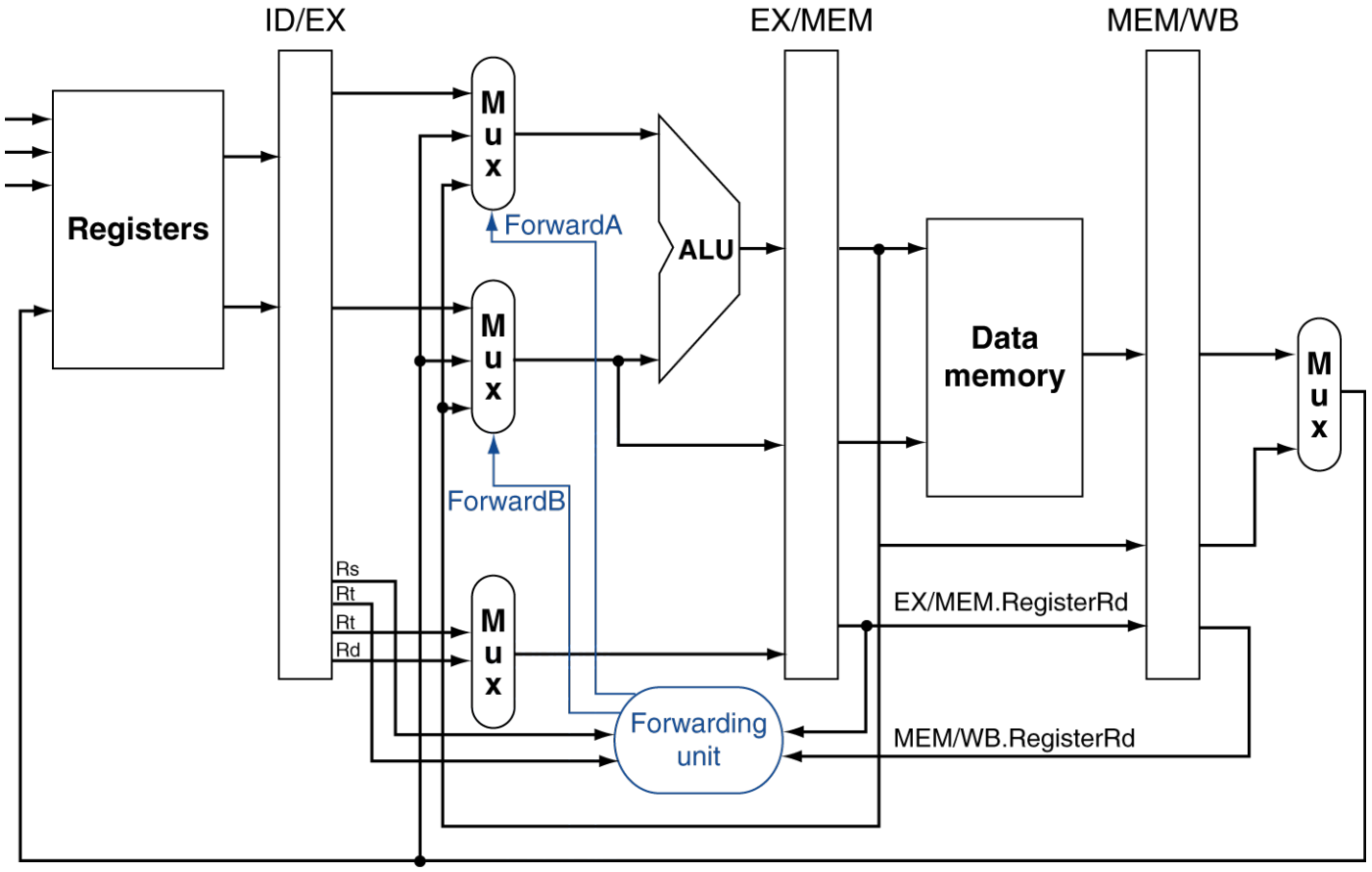
- Pass register numbers along pipeline
  - e.g., ID/EX.RegisterRs = register number for Rs stored in ID/EX pipeline register
- ALU operand register numbers in EX stage are given by
  - ID/EX.RegisterRs, ID/EX.RegisterRt
- Data hazards when
  1a. EX/MEM.RegisterRd = ID/EX.RegisterRs
  1b. EX/MEM.RegisterRd = ID/EX.RegisterRt
  2a. MEM/WB.RegisterRd = ID/EX.RegisterRs
  2b. MEM/WB.RegisterRd = ID/EX.RegisterRt



Detecting the Need to Forward

- Only forward **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 ≠ 0, MEM/WB.RegisterRd ≠ 0

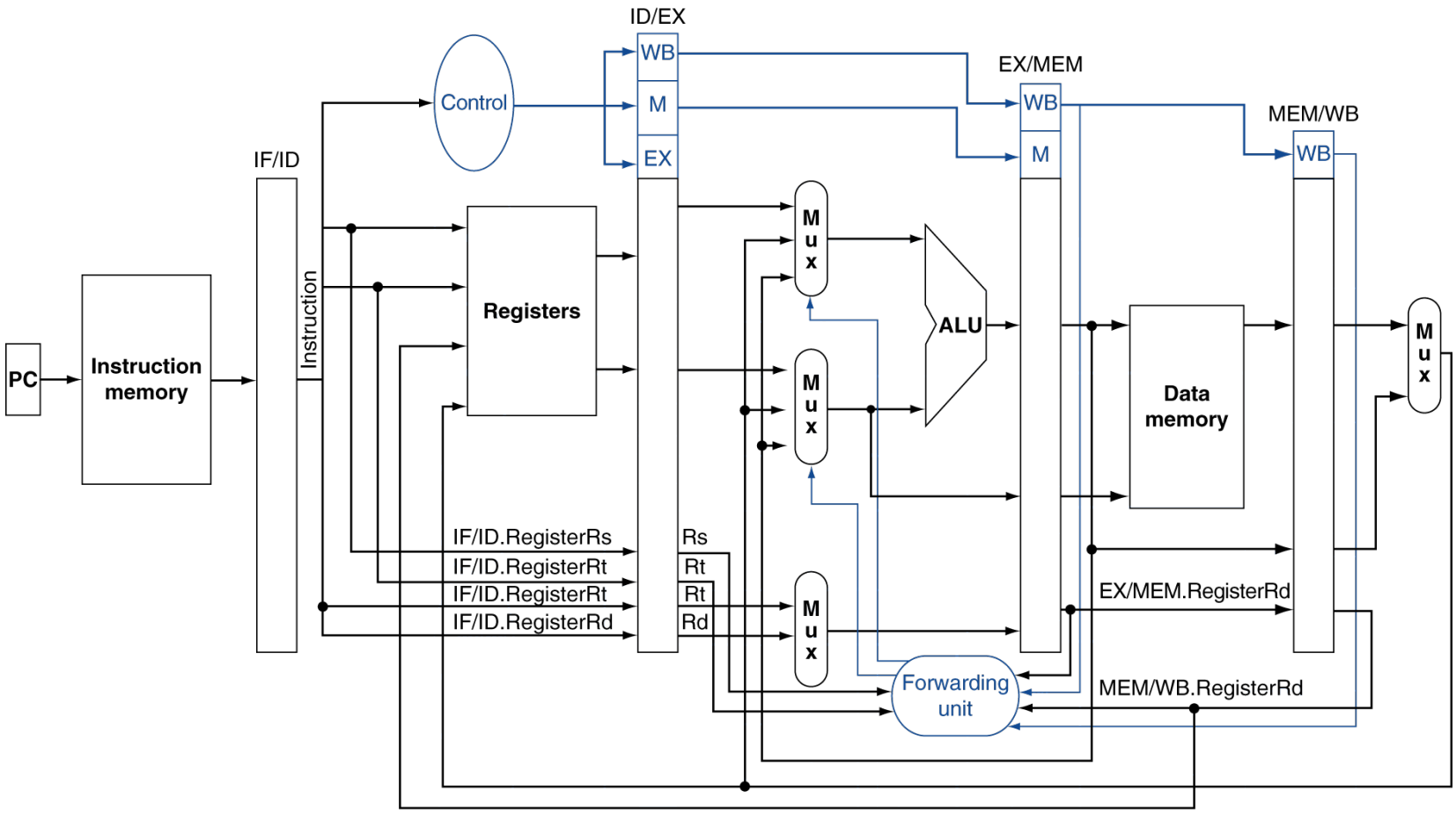
Forwarding Paths



Forwarding Conditions

- EX hazard
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Datapath with Forwarding

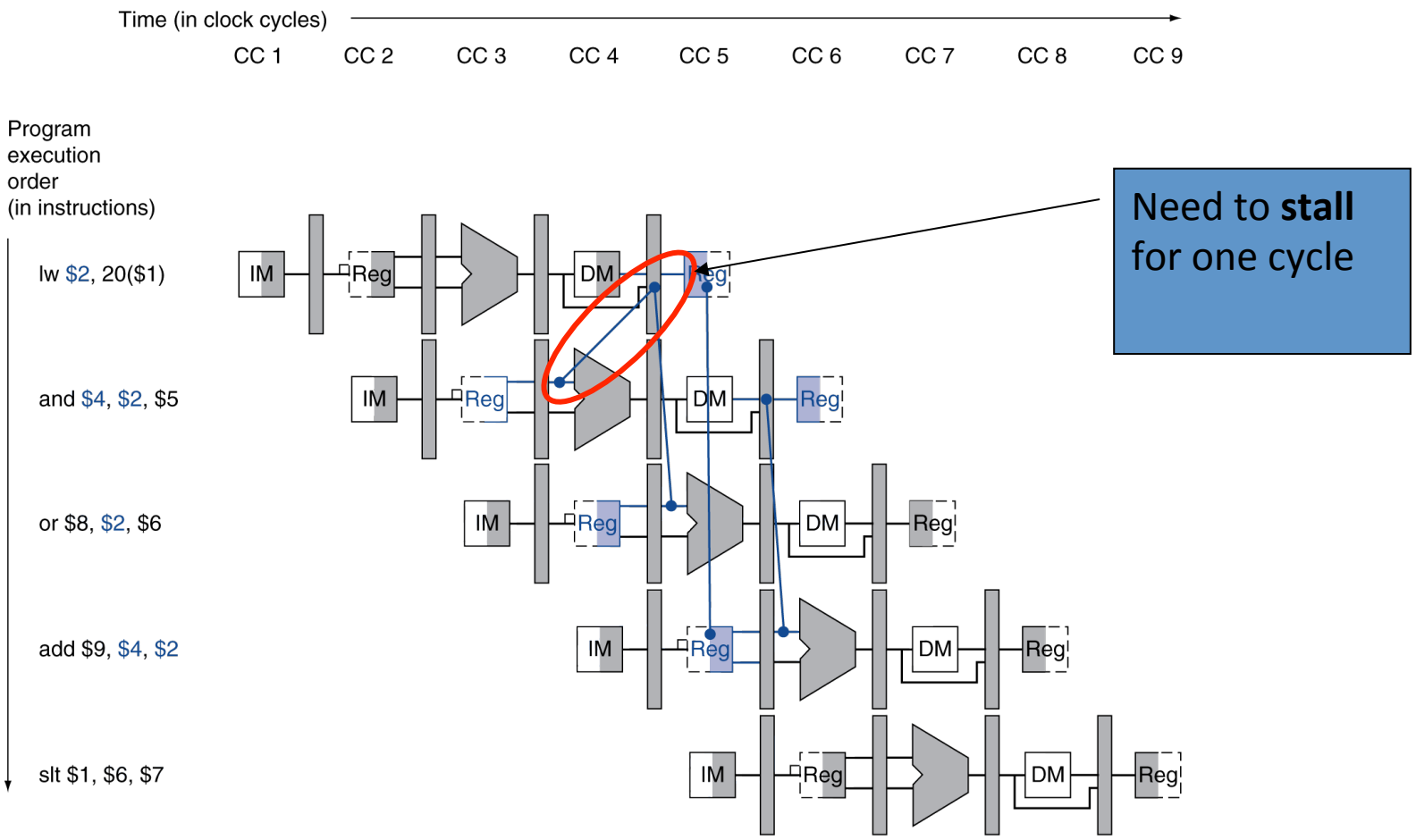


More Hazards

- Consider this sequence:

```
lw  $2, 20($1)
and $4, $2, $5
or  $8, $2, $6
add $9, $4, $2
slt $1, $6, $7
```

Load-Use Data Hazard



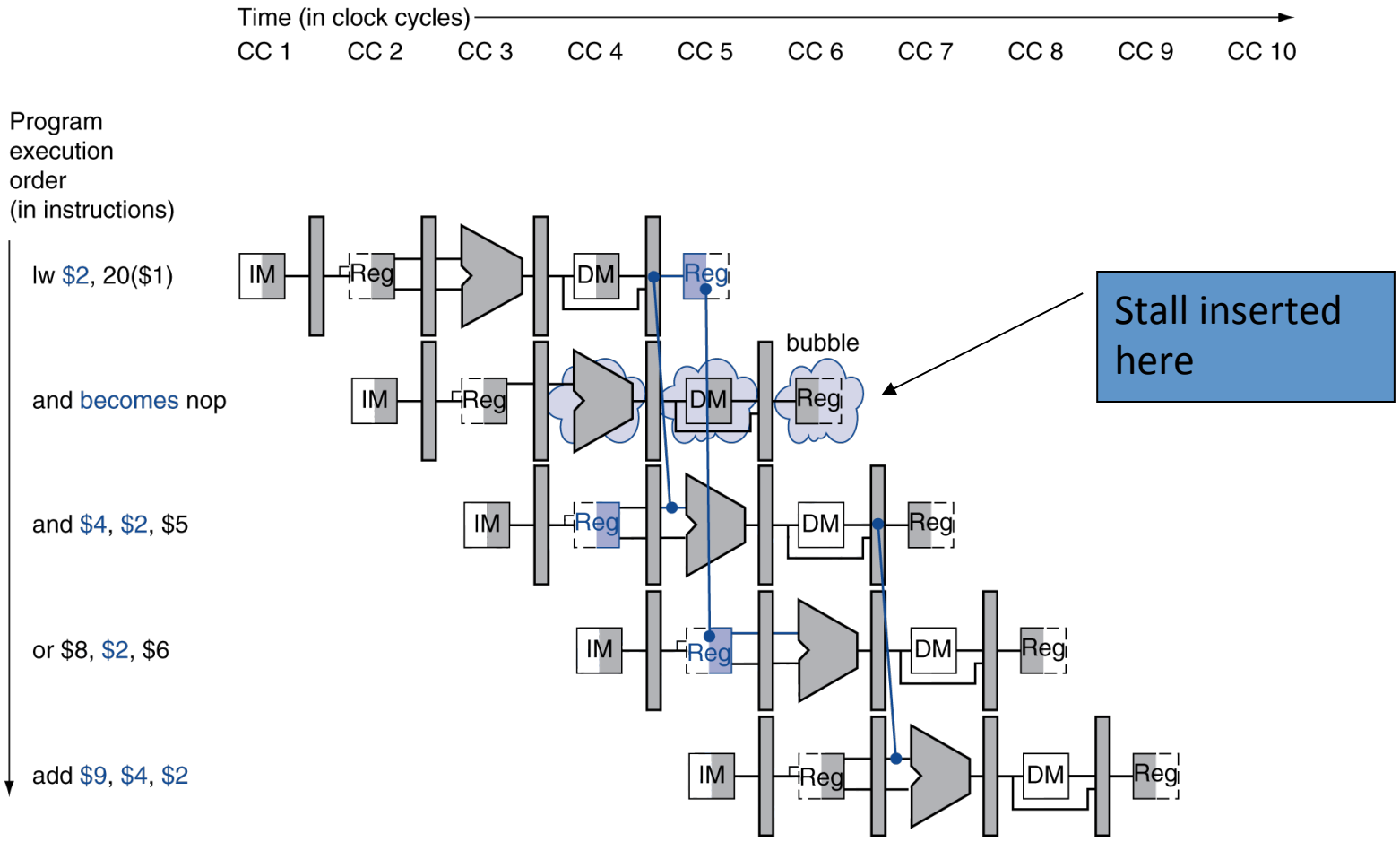
Load-Use Hazard Detection

- Check when instruction is decoded in ID stage
- ALU operand register numbers in ID stage are given by
  - IF/ID.RegisterRs, IF/ID.RegisterRt
- Load-use hazard when
  - ID/EX.MemRead and
    ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
    (ID/EX.RegisterRt = IF/ID.RegisterRt))
- If detected, stall and insert bubble

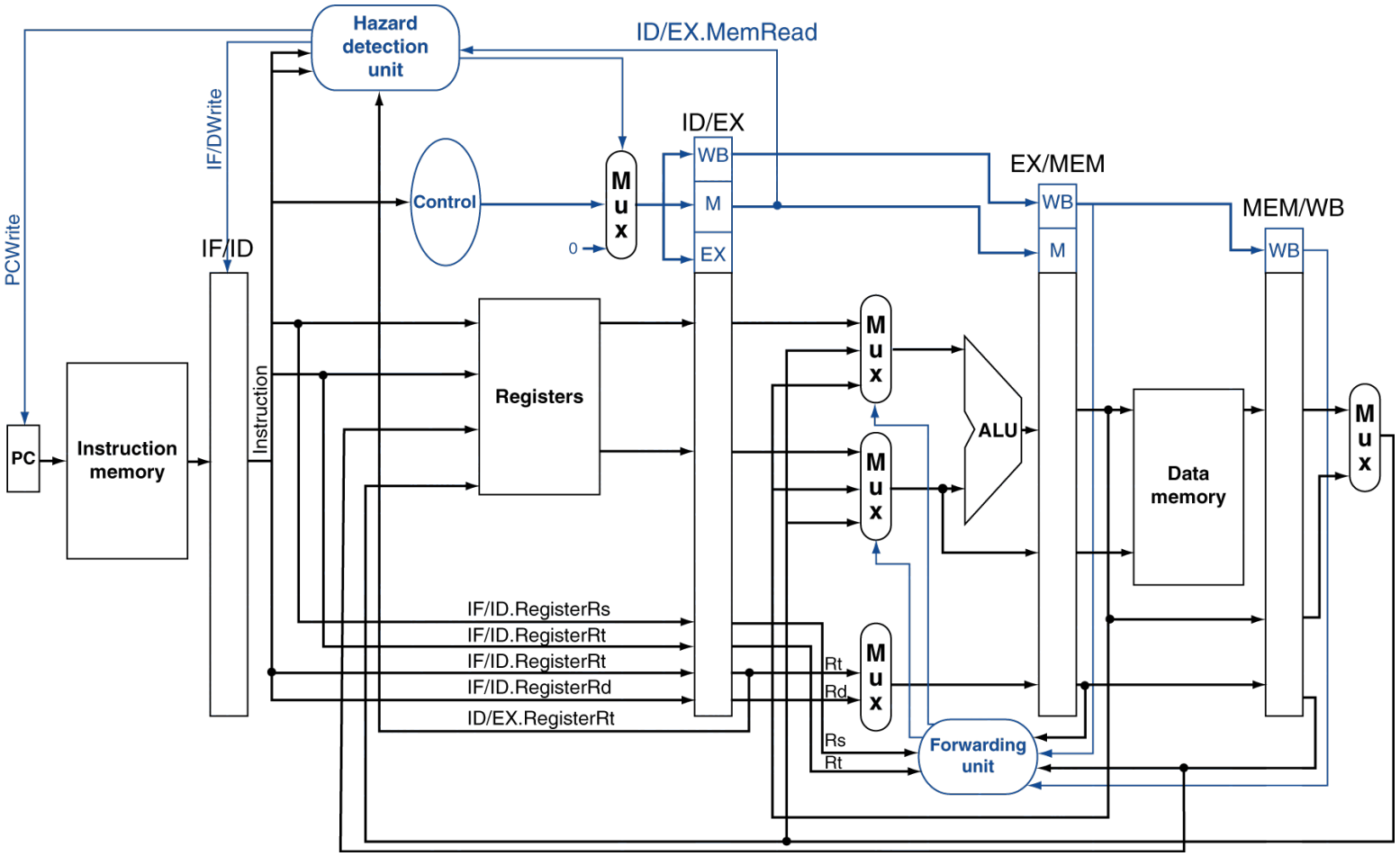
How to Stall the Pipeline

- 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 is decoded again
  - Following instruction is fetched again
  - **1-cycle stall allows MEM to read data for lw**
    - Can subsequently forward to EX stage

Stall/Bubble in the Pipeline



Datapath with Hazard Detection



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

Review and Questions

- Hazards
  - Data
  - Control
  - Structural
- Hazard detection
- Hazard prevention