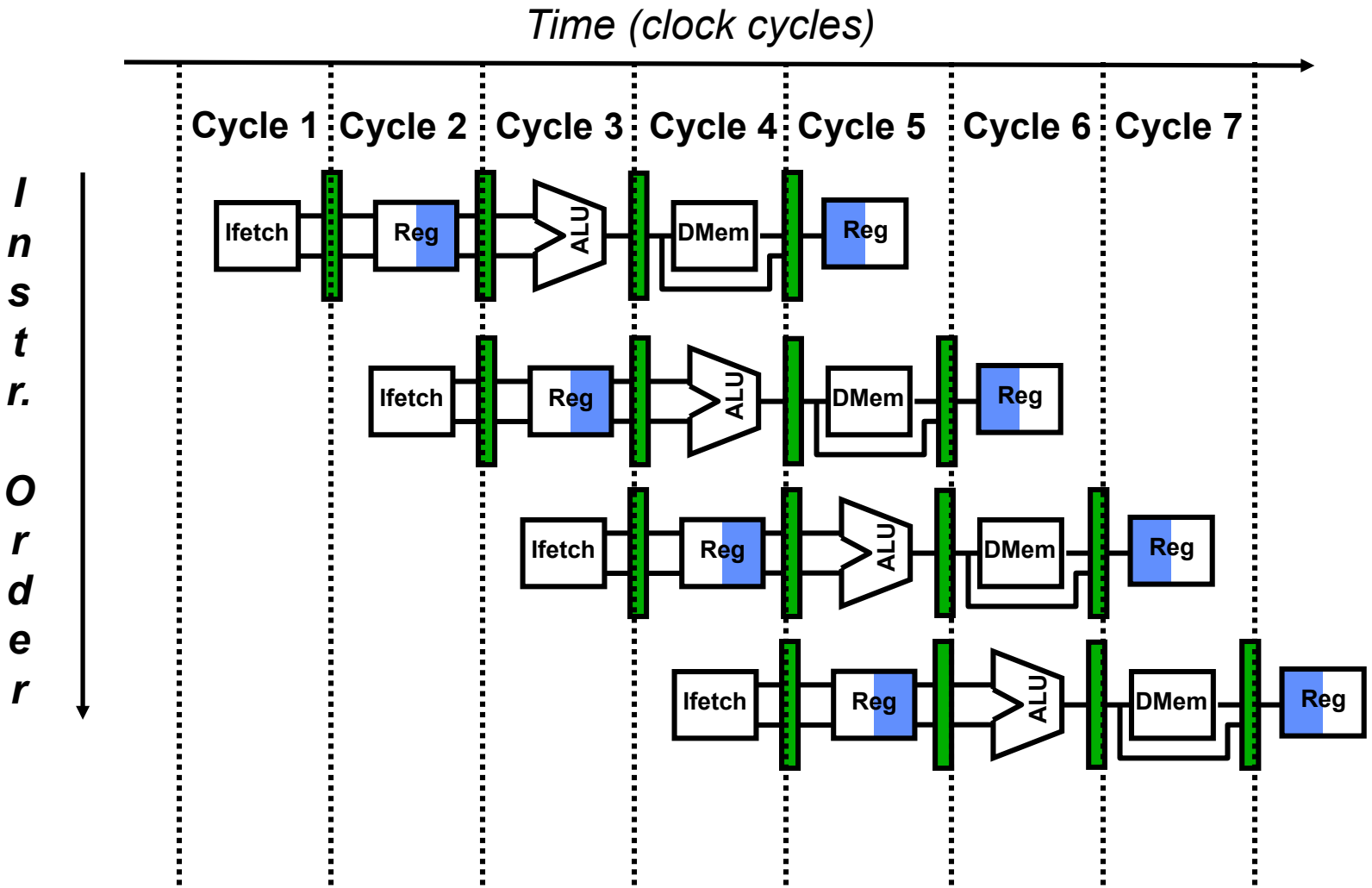
CPT_S 260 Intro to Computer Architecture
Lecture 34

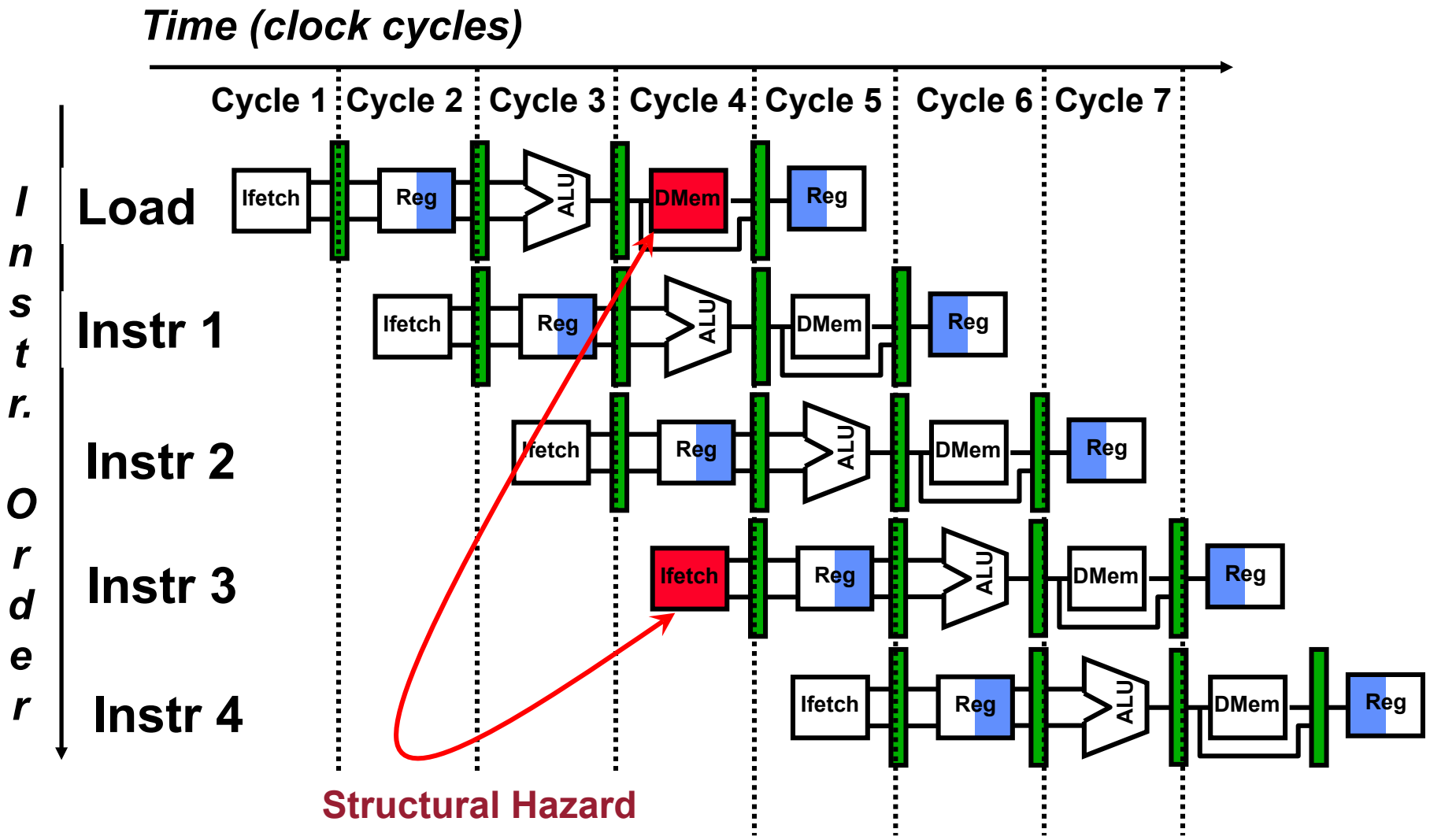
Pipeline Hazards
April 8, 2022

Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

Recap: Visualizing Pipelining

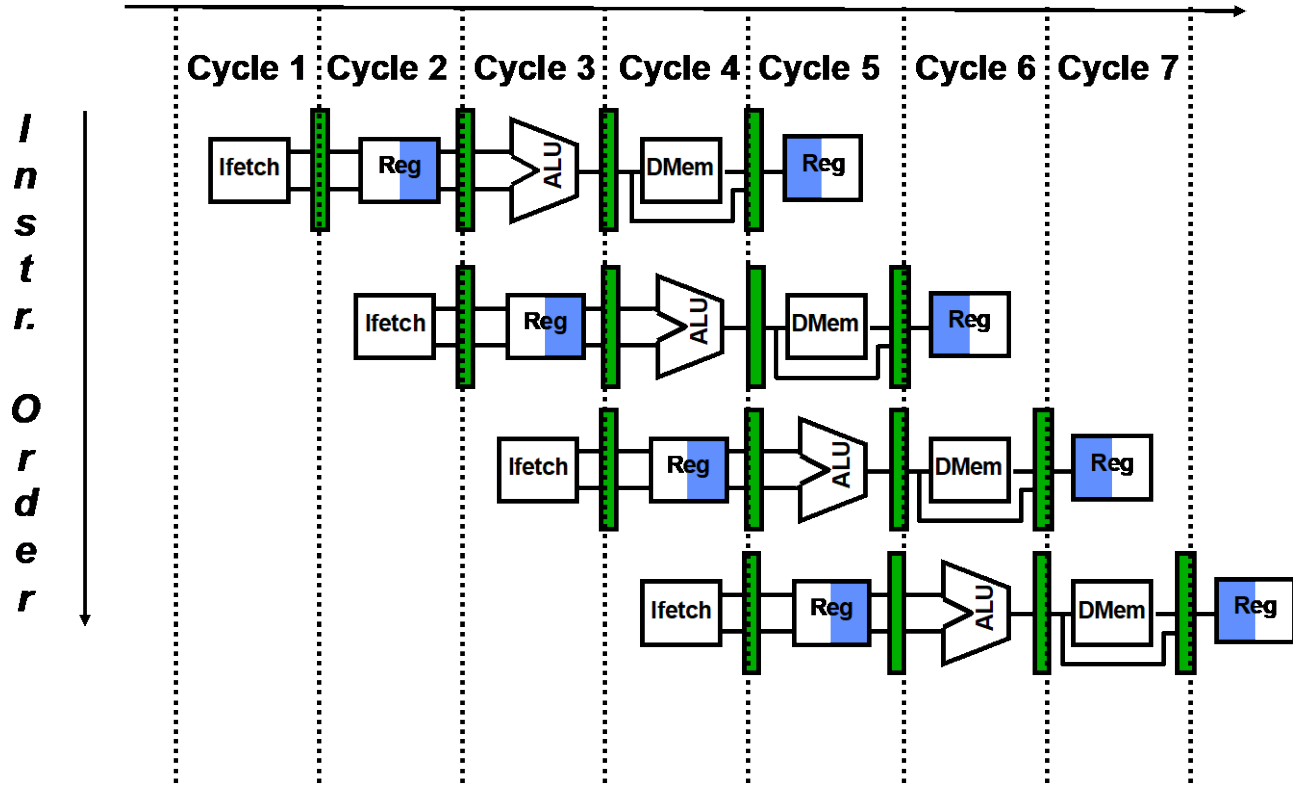


Recap: One Memory Port/Structural Hazards



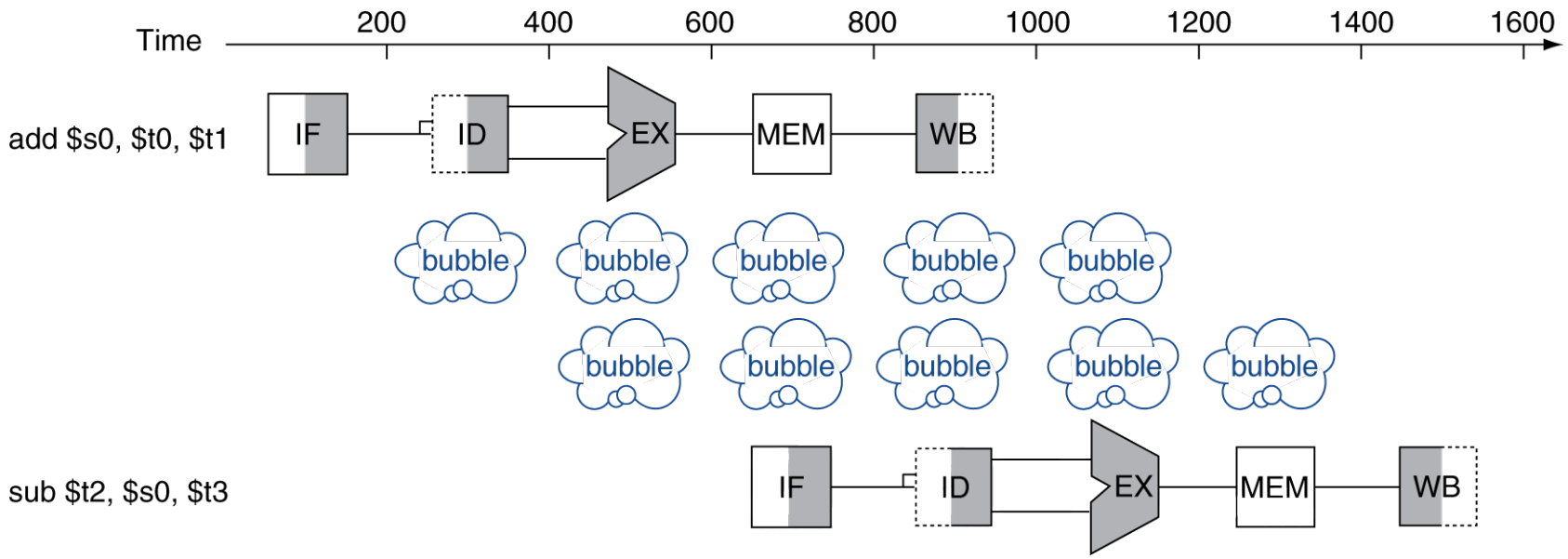
Structural Hazard

- **Register file is used in both ID and Write Back Stages**
- **This leads to a structural hazard**
- **Solution**
  - Write in first half of the clock cycle
  - Read in the second of the clock cycle

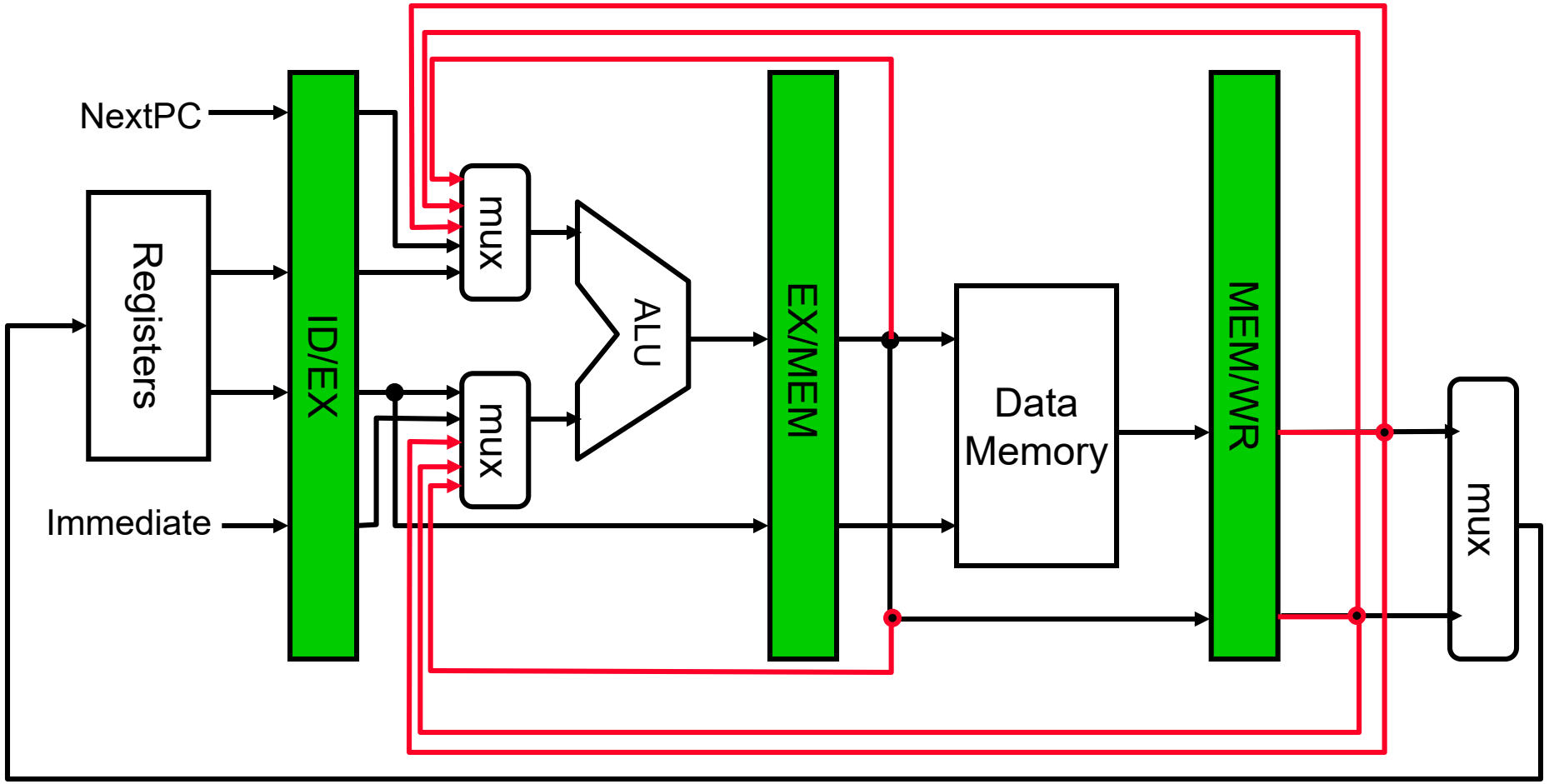


Recap: Data Hazards

- **An instruction depends on completion of data access by a previous instruction**
  - `add $s0, $t0, $t1`
    `sub $t2, $s0, $t3`

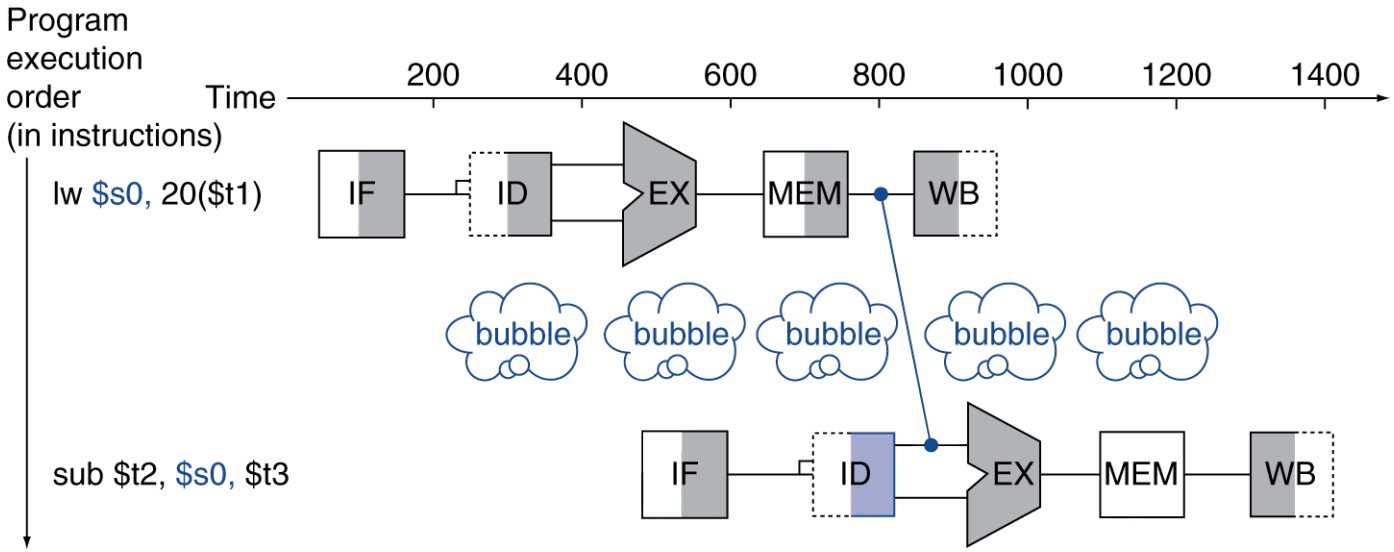


HW Change for Forwarding



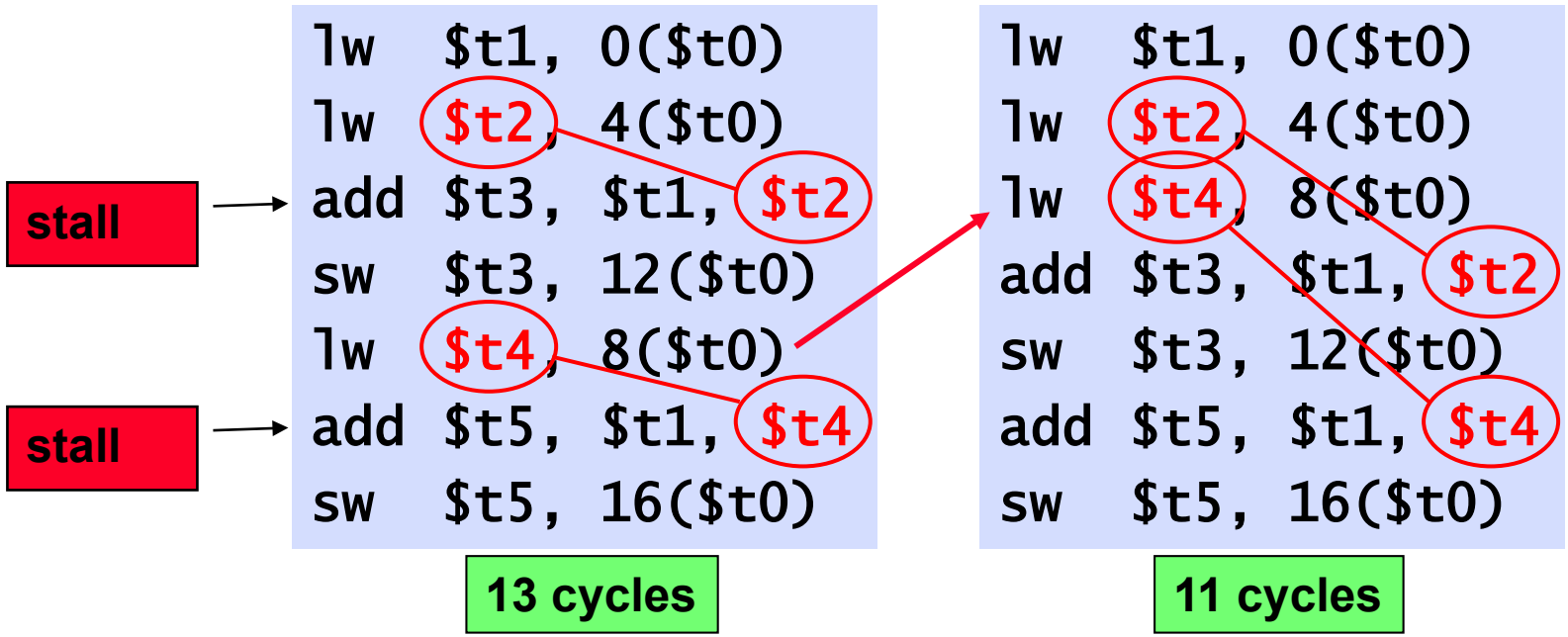
Recap: Load-Use Data Hazard

- **Can't always avoid stalls by forwarding**
  - If value not computed when needed
  - Can't forward backward in time!



Code Scheduling to Avoid Stalls

- Reorder code to avoid use of load result in the next instruction
- C code for `A = B + E; C = B + F;`

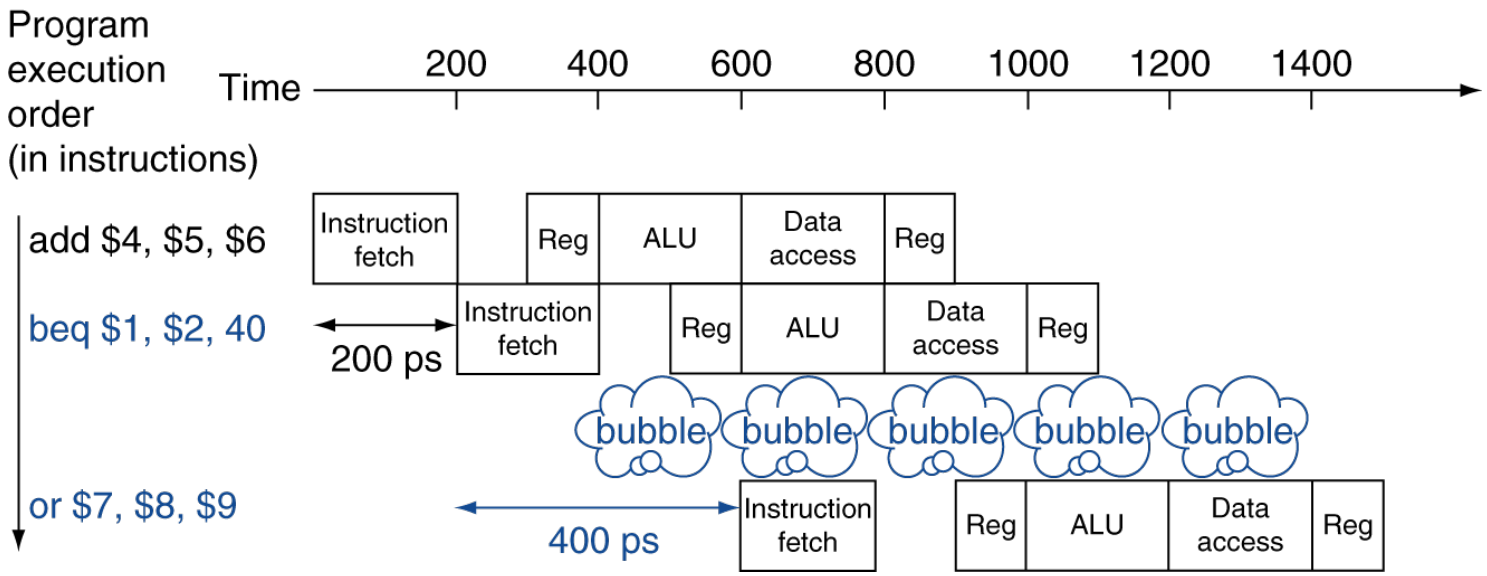


Control Hazards

- **Branch determines flow of control**
  - Fetching next instruction depends on branch outcome
  - Pipeline can't always fetch correct instruction
    - » Still working on ID stage of branch
- **In MIPS pipeline**
  - Need to compare registers and compute target early in the pipeline
  - Add hardware to do it in ID stage

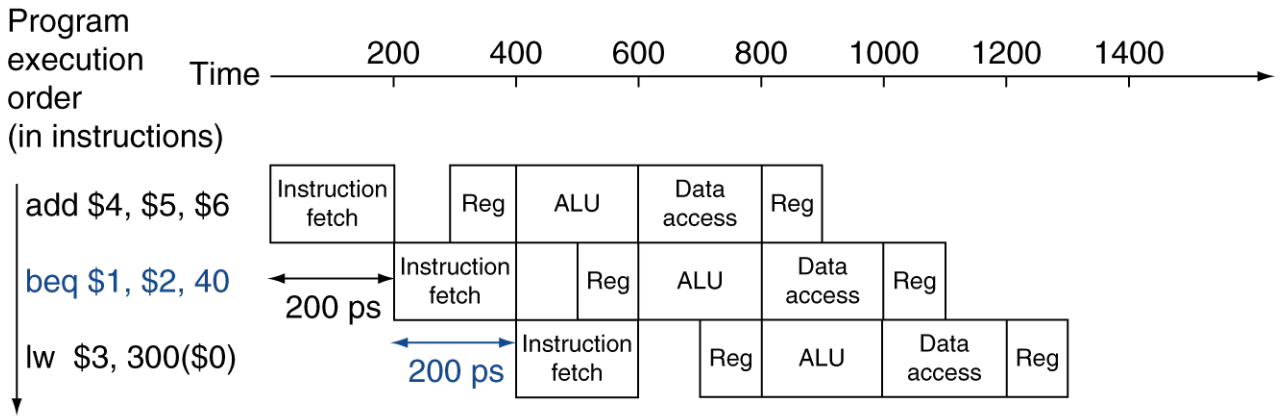
Stall on Branch

- **Wait until branch outcome determined before fetching next instruction**

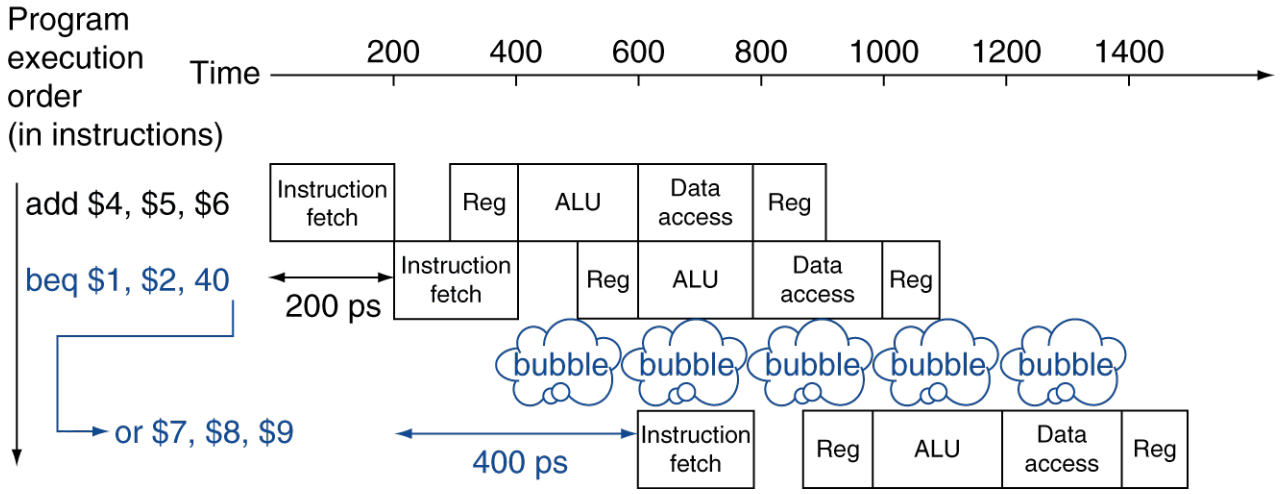


MIPS with Predict Not Taken

Prediction correct



Prediction incorrect



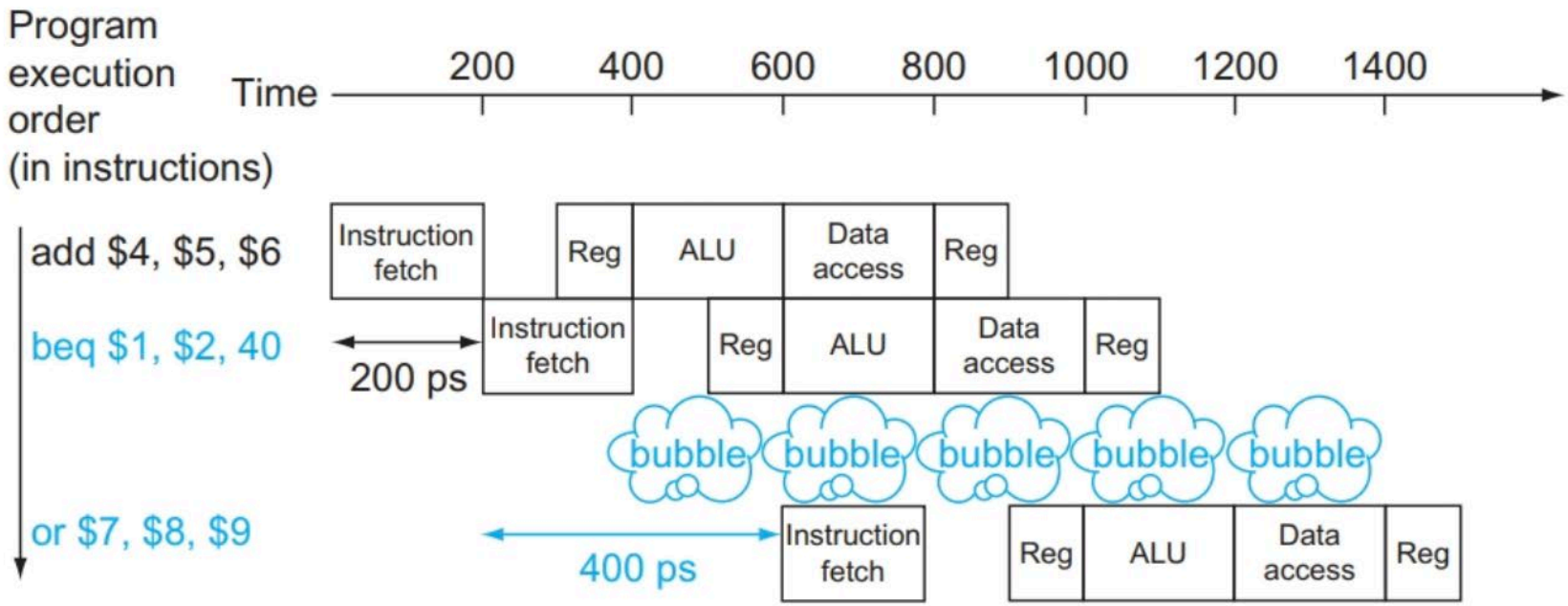
More-Realistic Branch Prediction

- **Static branch prediction**
  - Based on typical branch behavior
  - Example: loop and if-statement branches
    - » Predict backward branches taken
    - » Predict forward branches not taken
- **Dynamic branch prediction**
  - Hardware measures actual branch behavior
    - » e.g., record recent history of each branch
  - Assume future behavior will continue the trend
    - » When wrong, stall while re-fetching, and update history

Branch Prediction

- **Longer pipelines can't readily determine branch outcome early**
  - Stall penalty becomes unacceptable
- **Predict outcome of branch**
  - Only stall if prediction is wrong
- **In MIPS pipeline**
  - Can predict branches not taken
  - Fetch instruction after branch, *delayed branch*

Branch Visualization



Check Point Problems

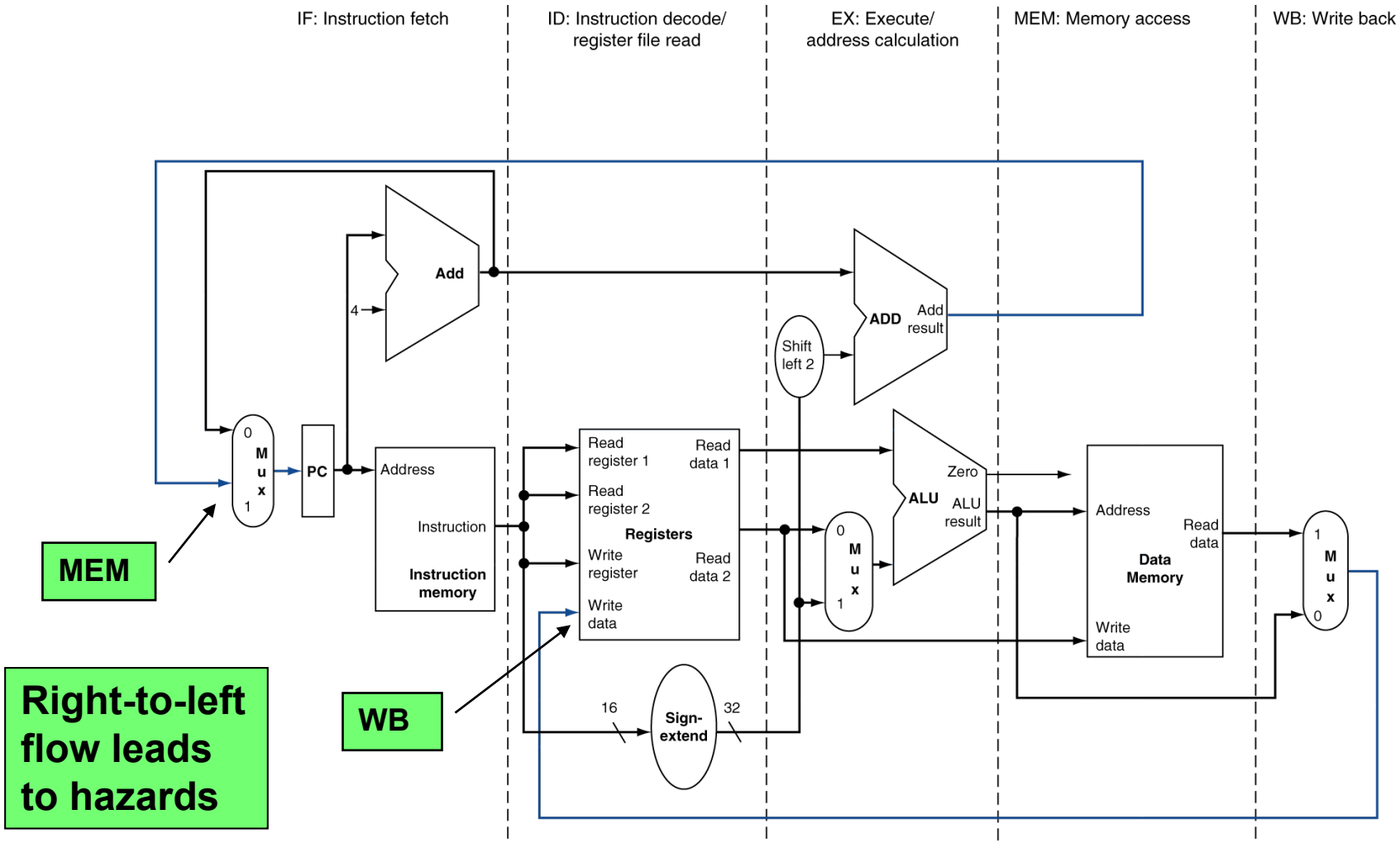
For each code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding.

Sequence 1	Sequence 2	Sequence 3

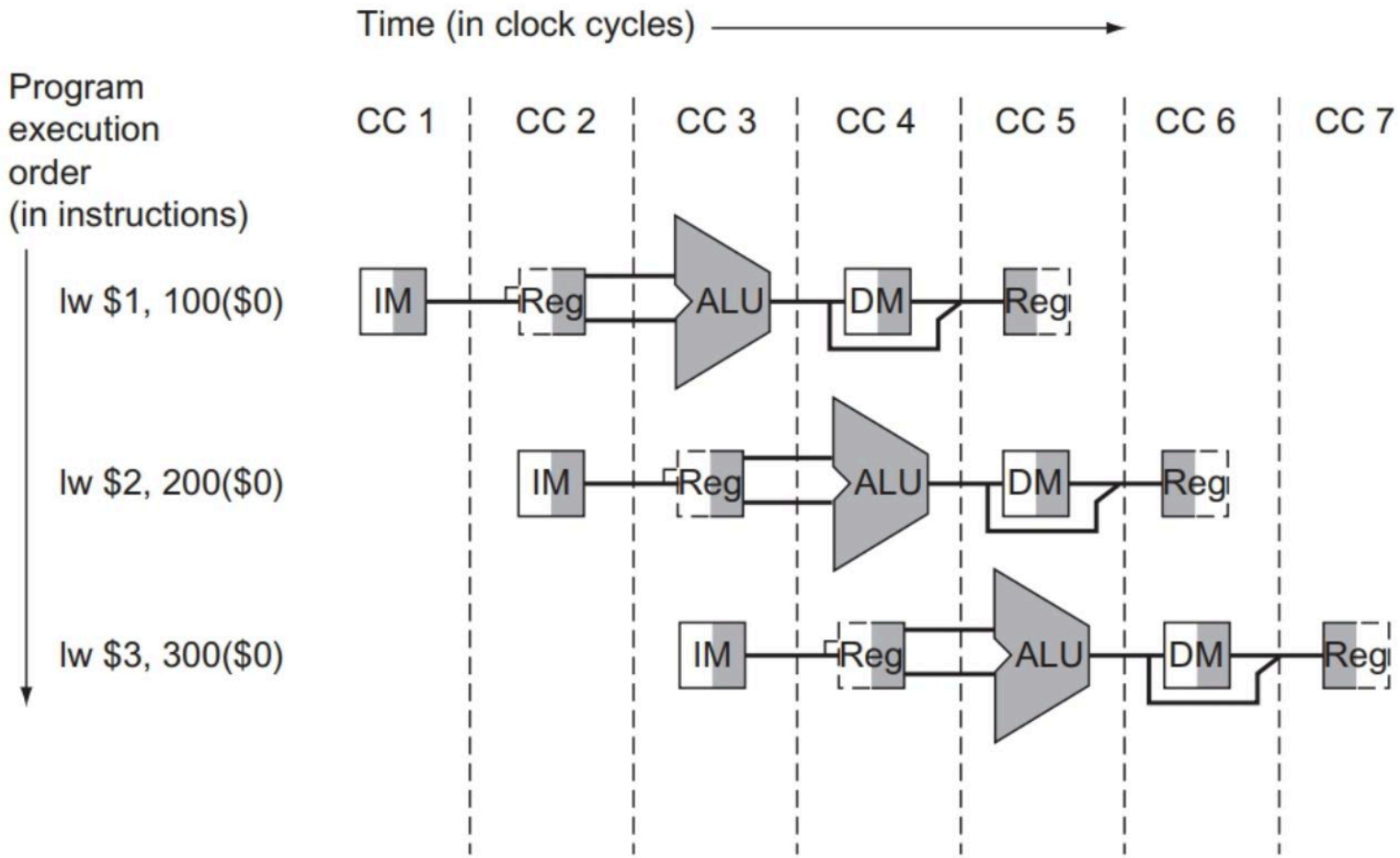
`lw $t0,0($t0)`
`add $t1,$t0,$t0`
	`add $t1,$t0,$t0`
	`addi $t2,$t0,#5`
	`addi $t4,$t1,#5`
		`addi $t1,$t0,#1`
		`addi $t2,$t0,#2`
		`addi $t3,$t0,#2`
		`addi $t3,$t0,#4`
		`addi $t5,$t0,#5`

Pipeline Datapath

MIPS Pipelined Datapath

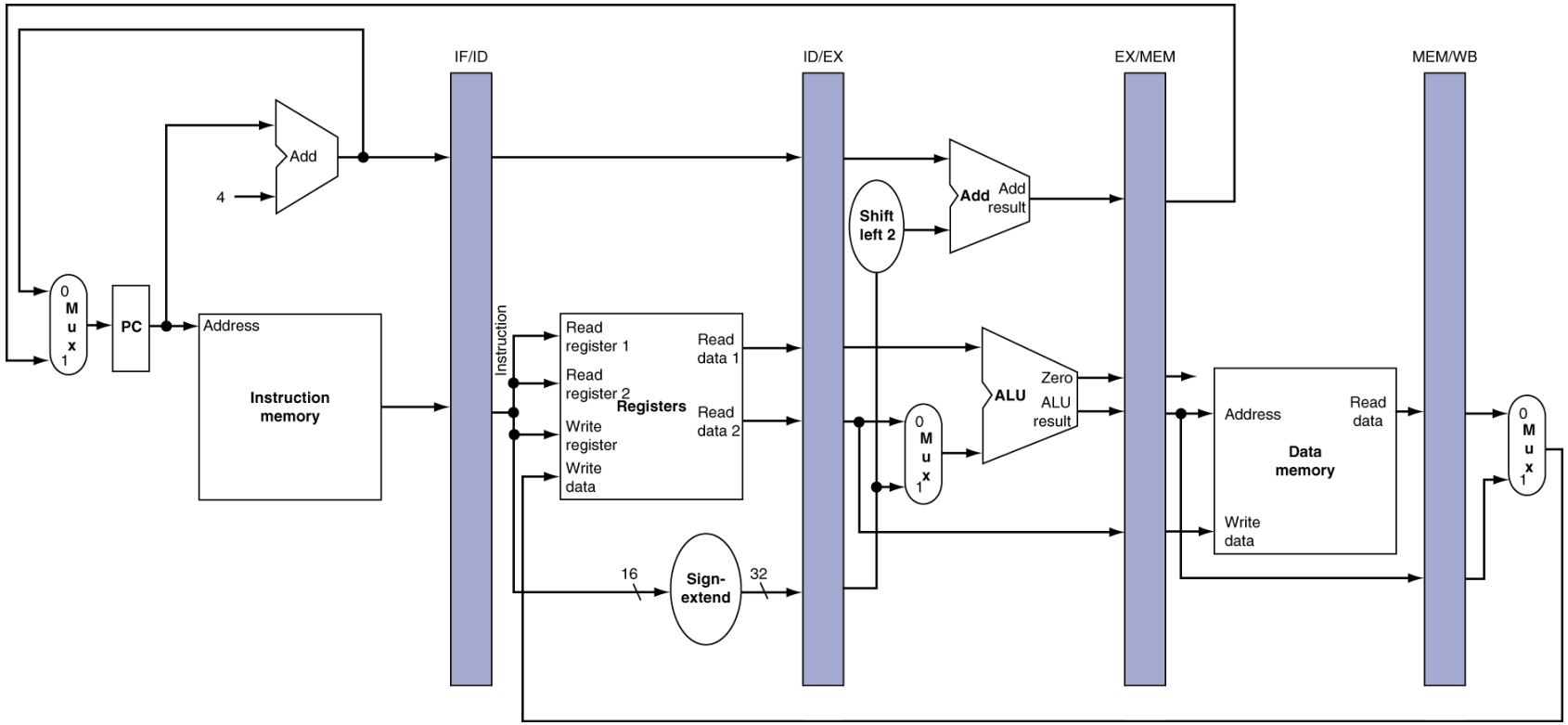


MIPS Pipelined Datapath



Pipeline registers

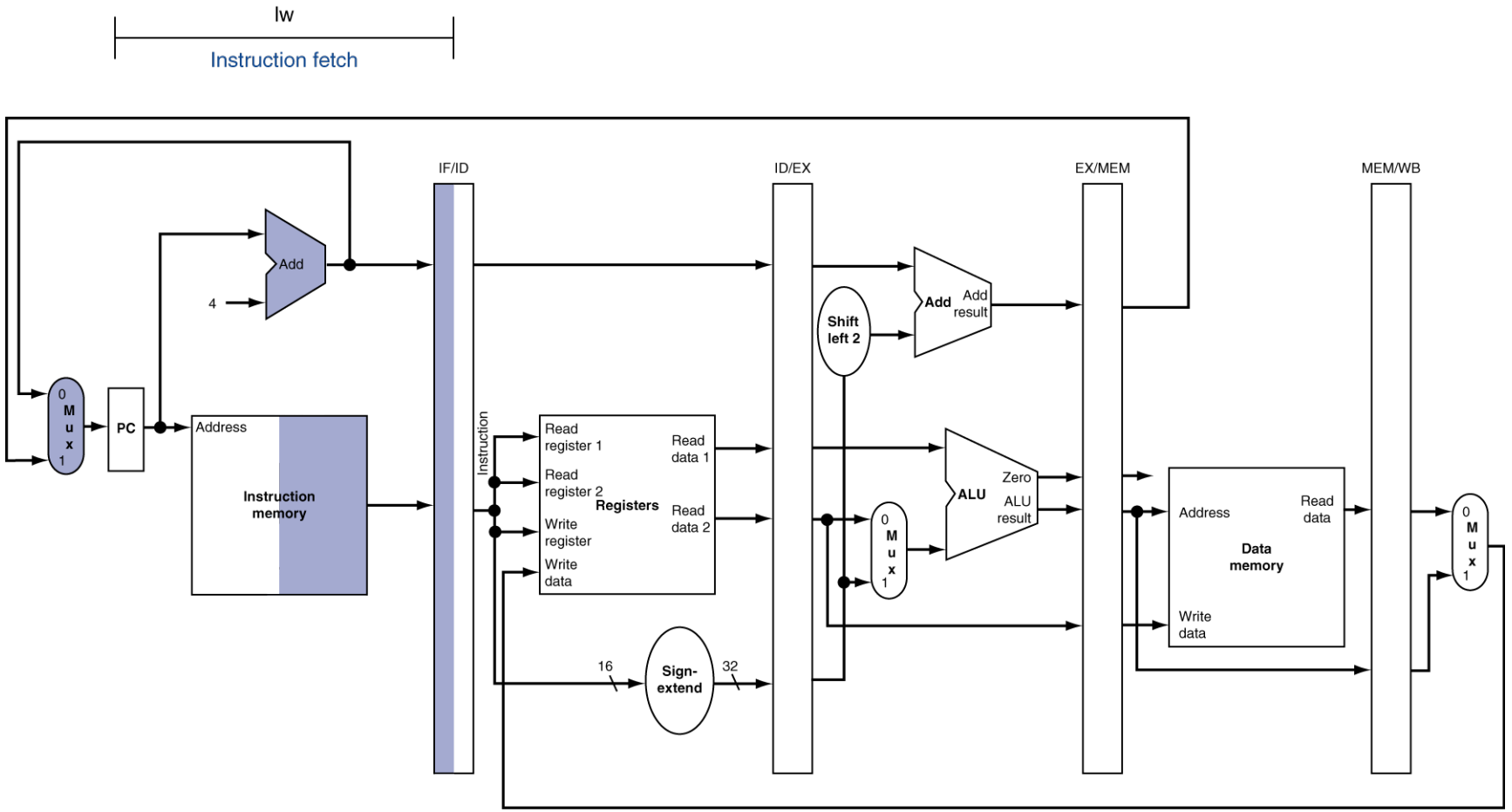
- **Need registers between stages**
  - To hold information produced in previous cycle



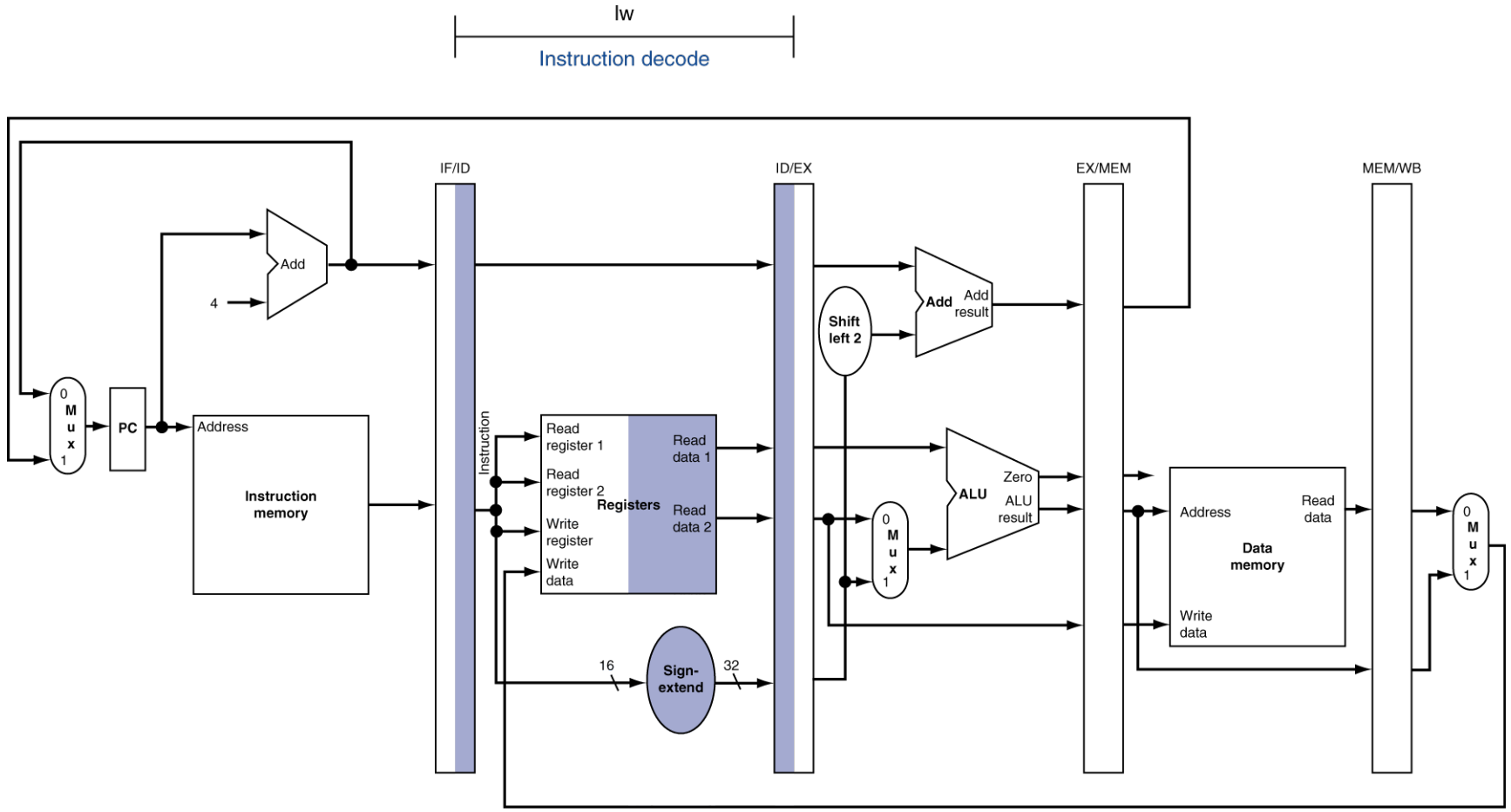
Pipeline Operation

- **Cycle-by-cycle flow of instructions through the pipelined datapath**
  - "Single-clock-cycle" pipeline diagram
    - » Shows pipeline usage in a single cycle
    - » Highlight resources used
  - c.f. "multi-clock-cycle" diagram
    - » Graph of operation over time
- **We'll look at "single-clock-cycle" diagrams for load & store**

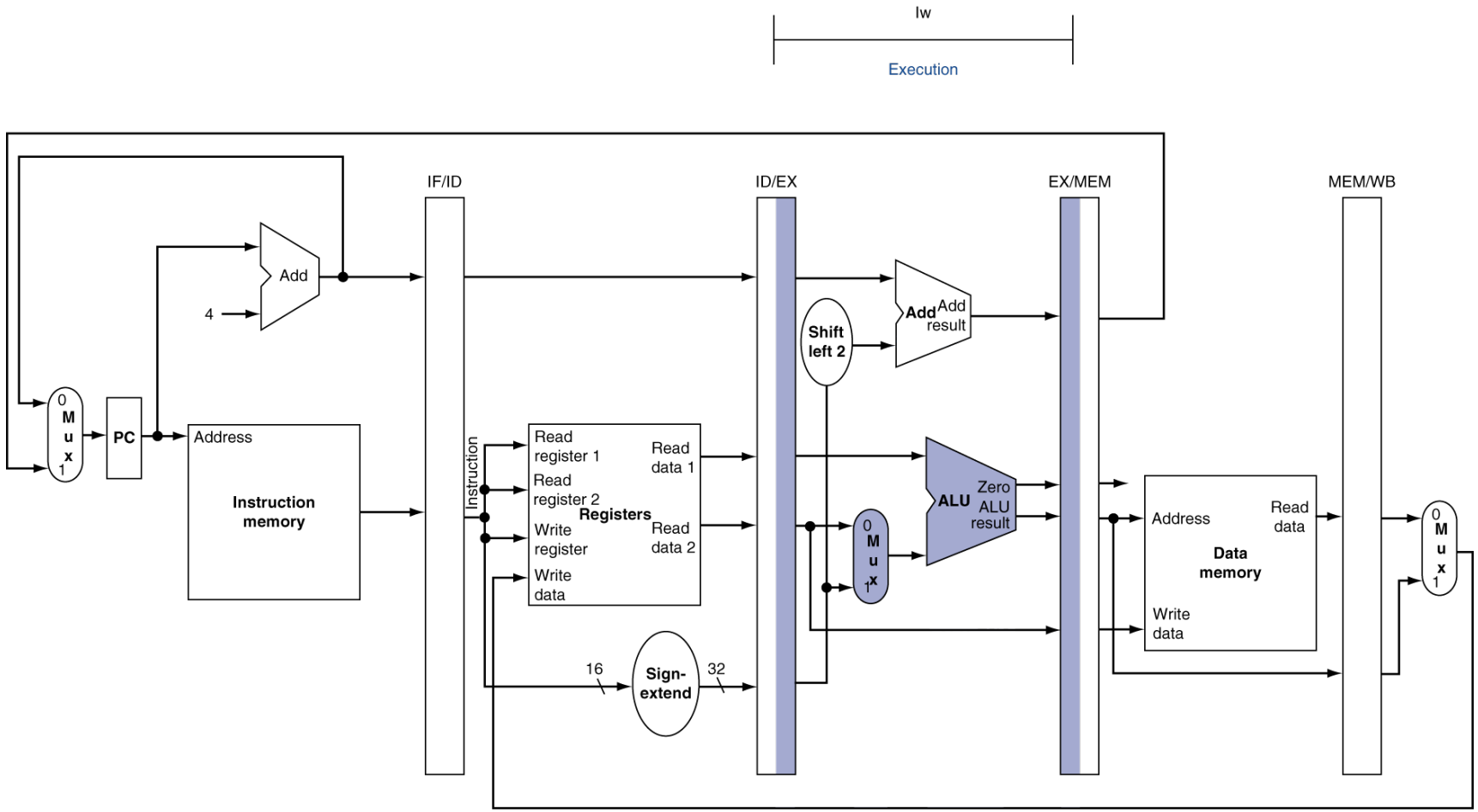
IF for Load, Store, …



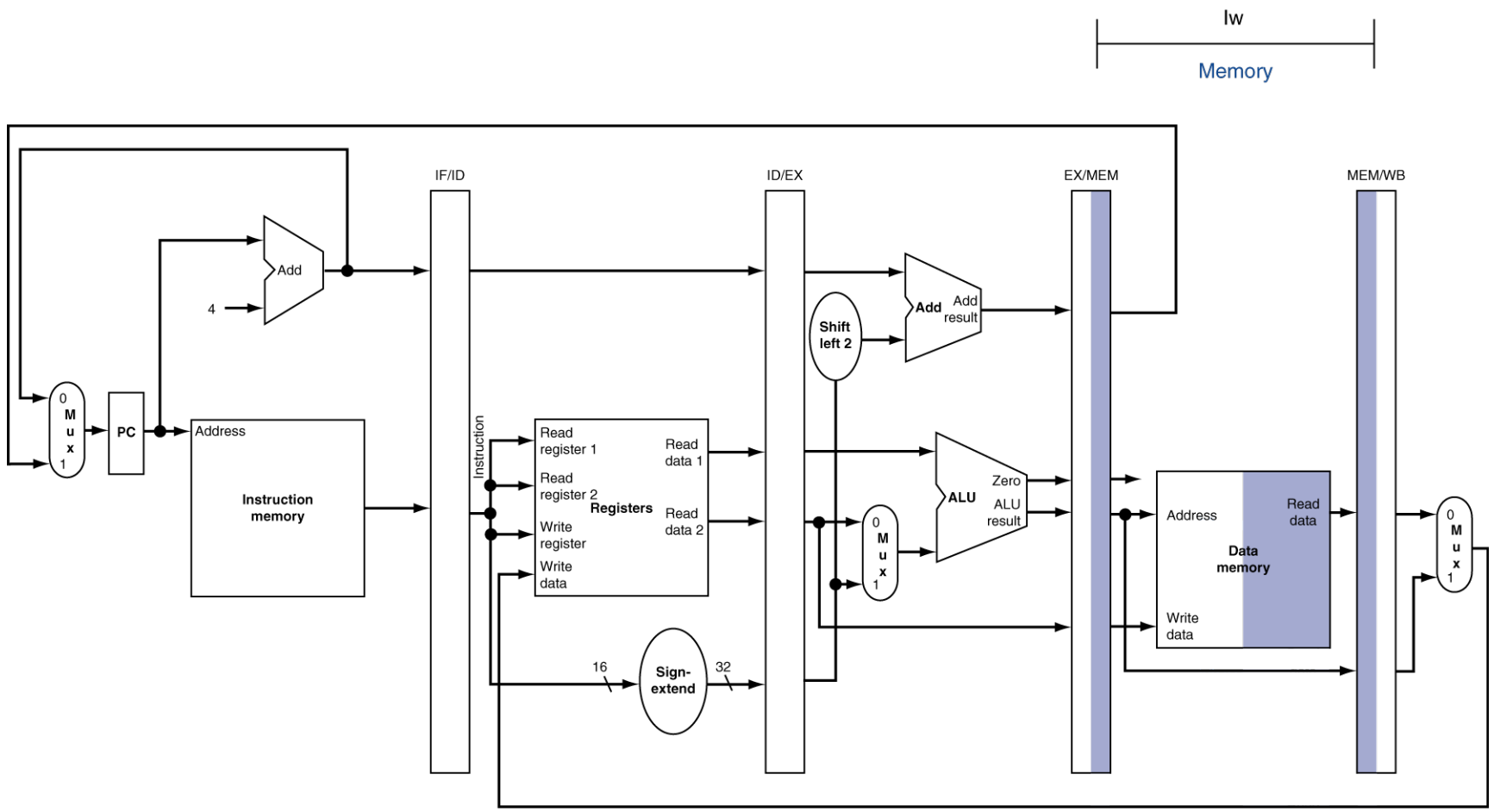
ID for Load, Store, …



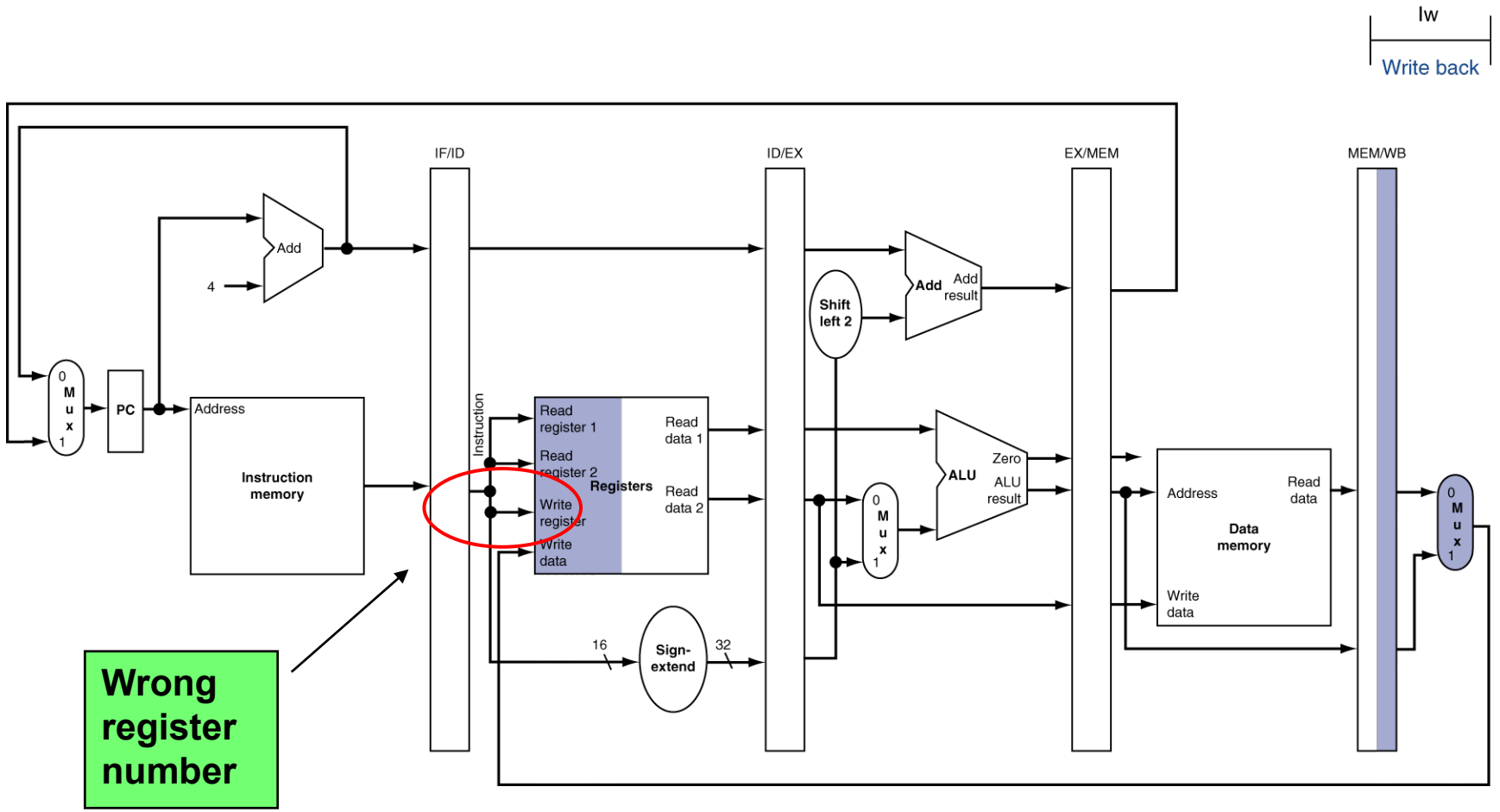
EX for Load


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

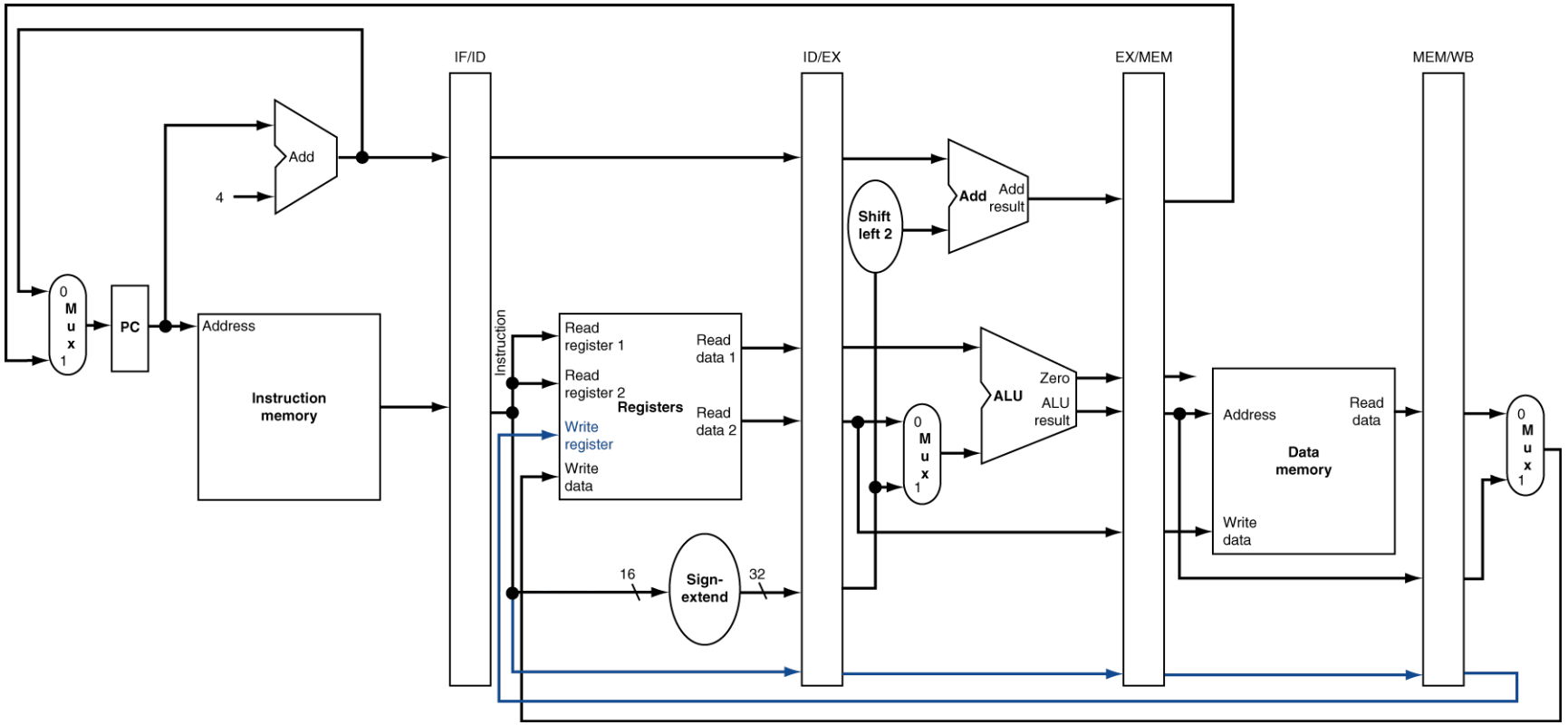
MEM for Load


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

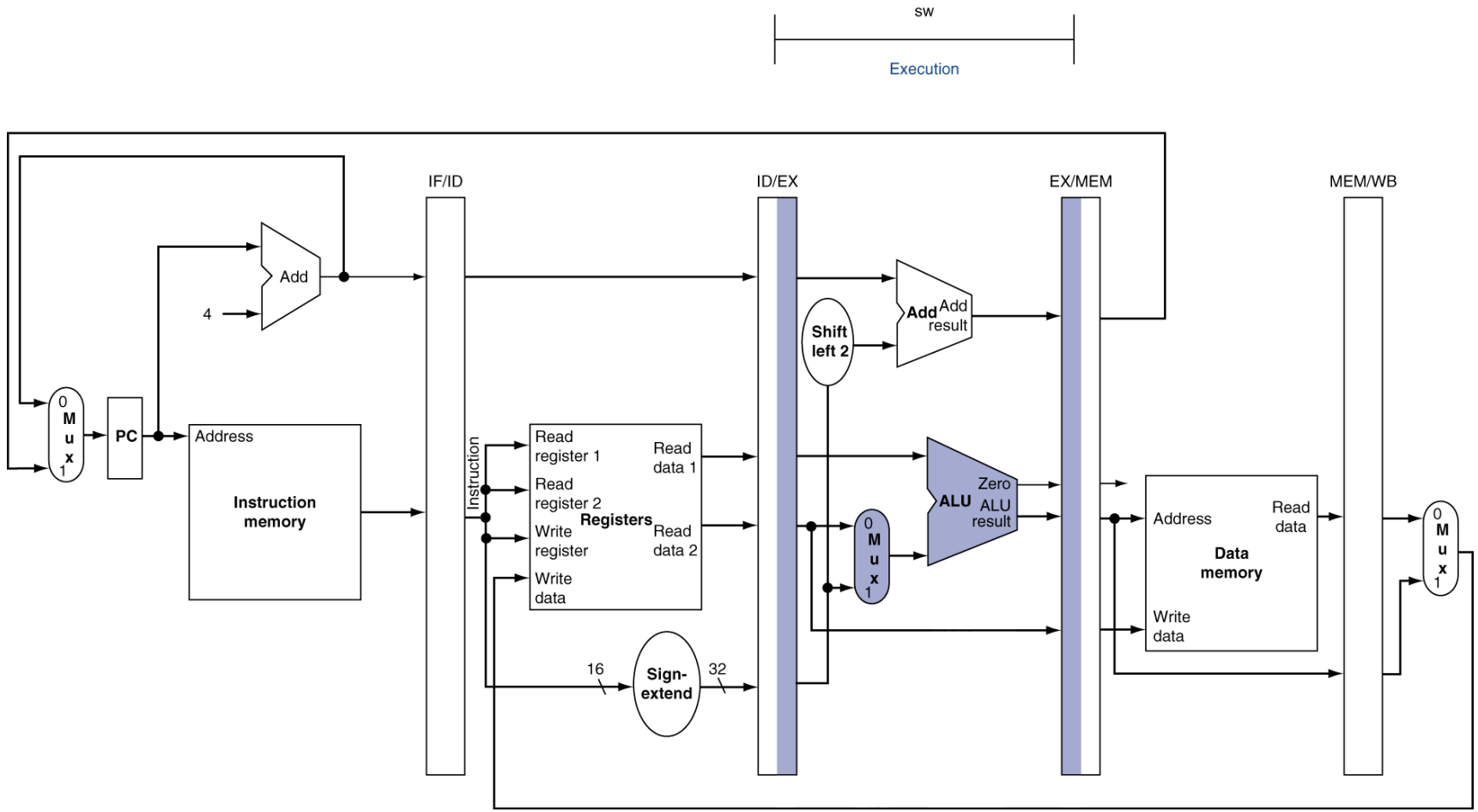
WB for Load


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
16
Sign-extend
32
Address
Data memory
Read data
Write data
Wrong register number

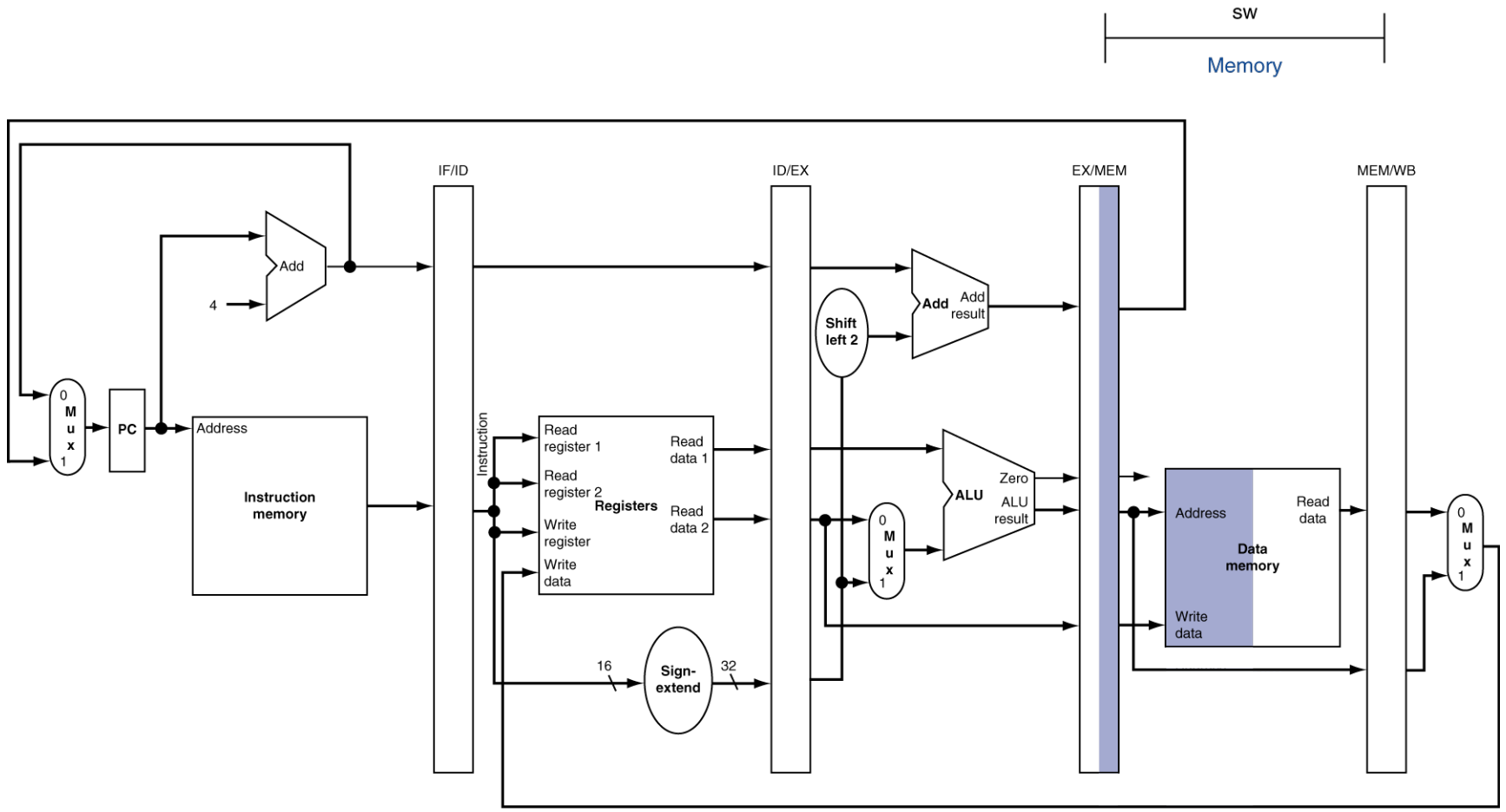
Corrected Datapath for Load



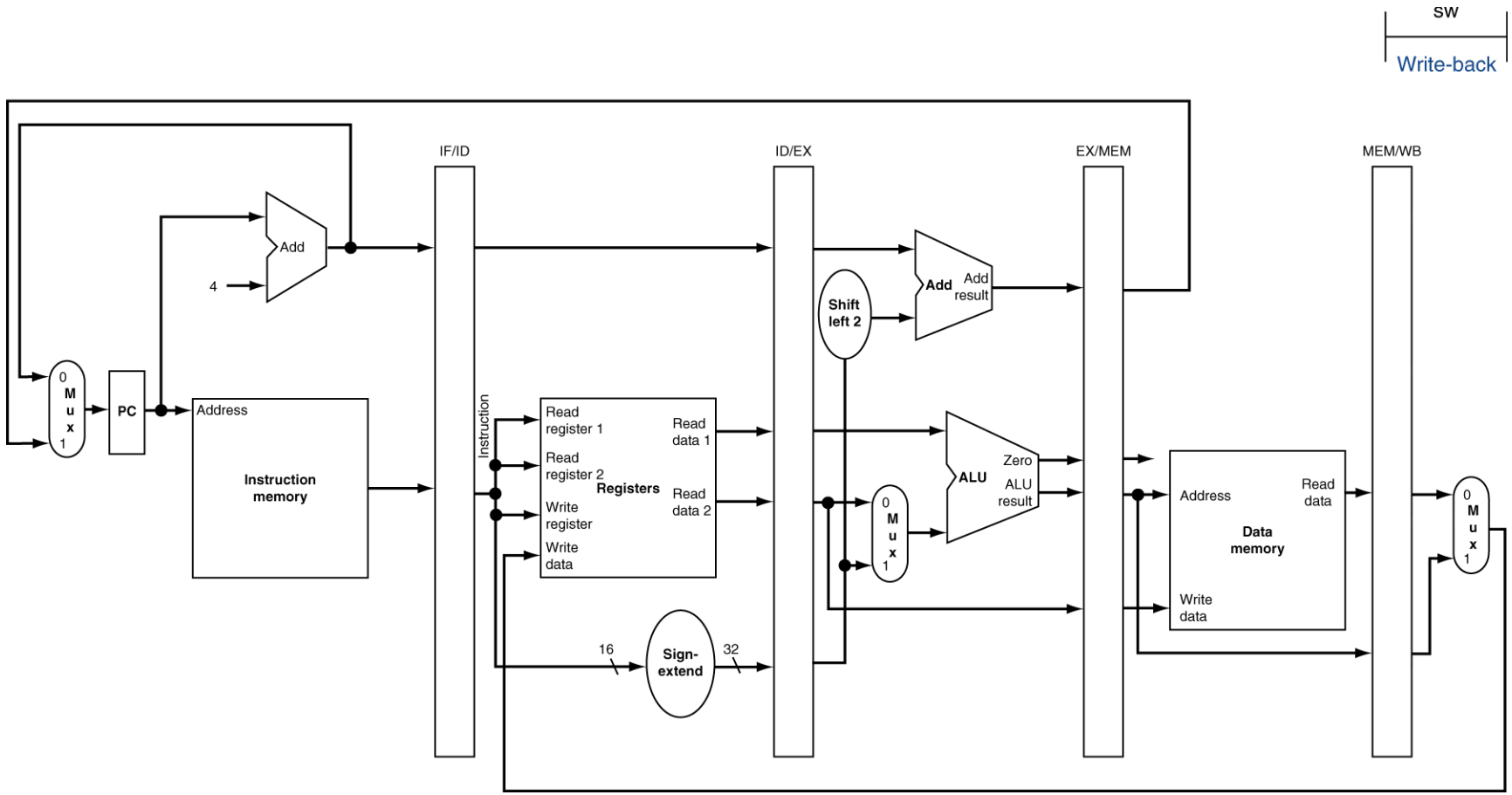
EX for Store


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

MEM for Store



WB for Store


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