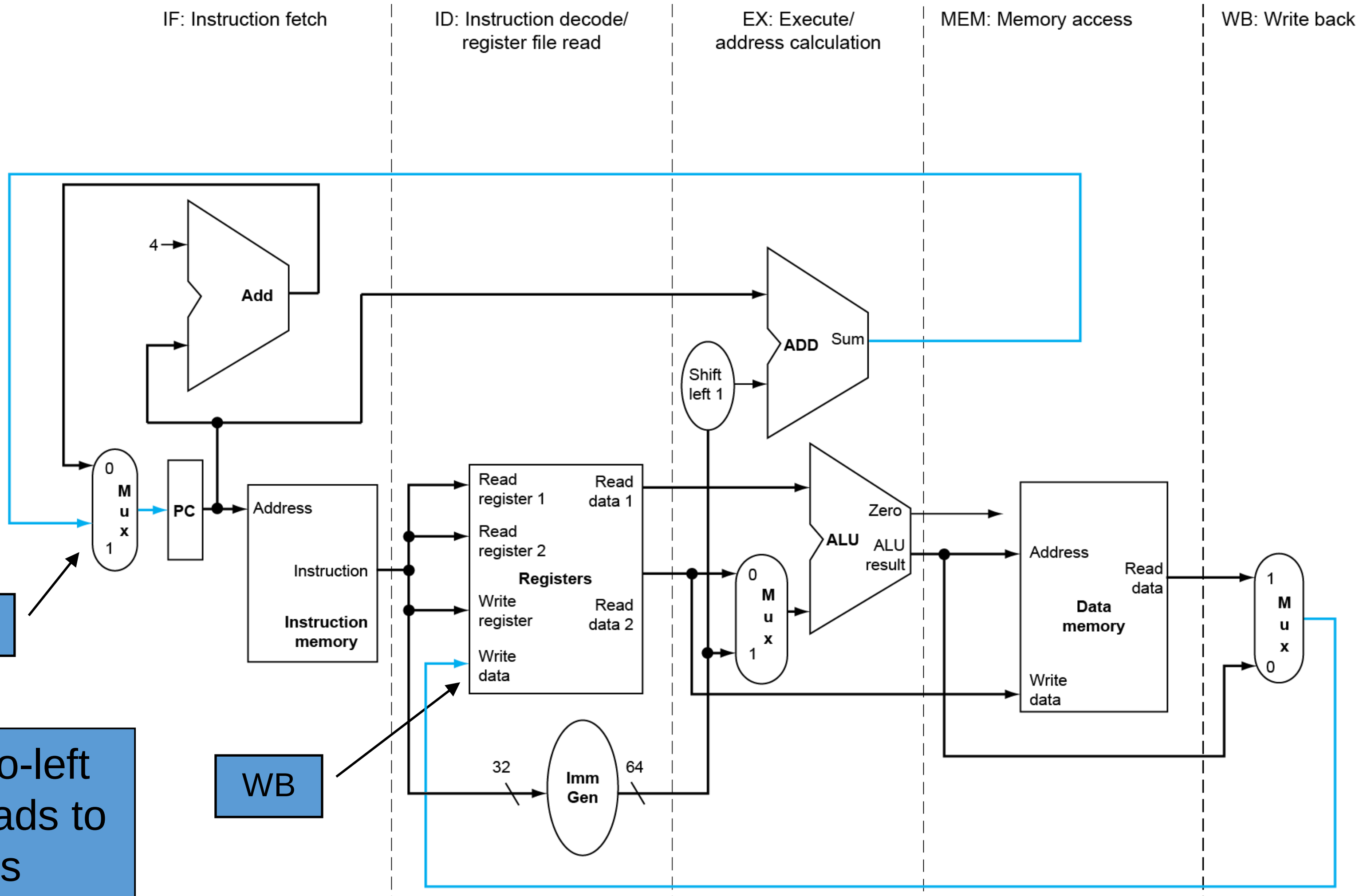
COMPUTER ORGANIZATION

Lecture 9 – Part2
Pipeline Datapath

2025 Spring

baiyh@sustech.edu.cn

RISC-V Pipelined Datapath



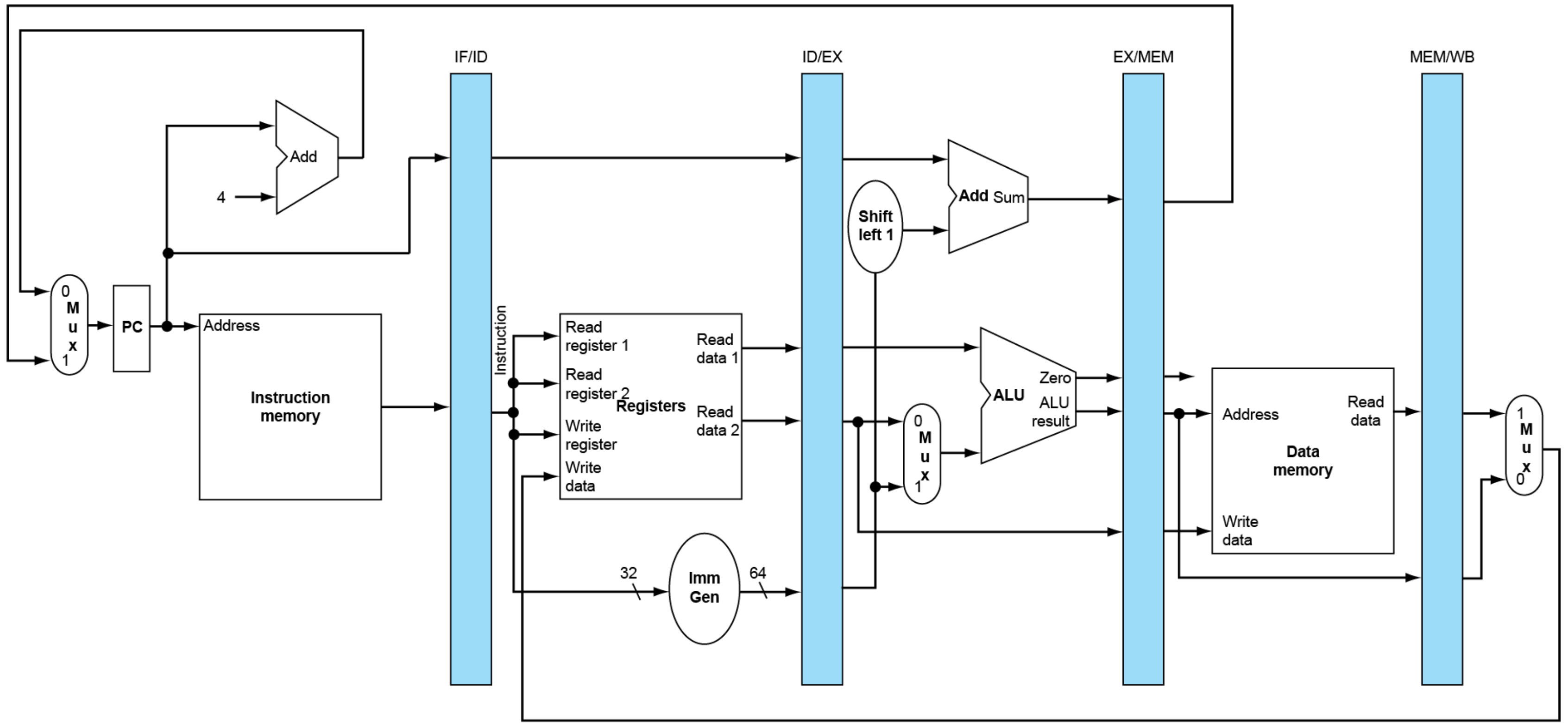
MEM

WB

Right-to-left flow leads to hazards

Pipeline registers

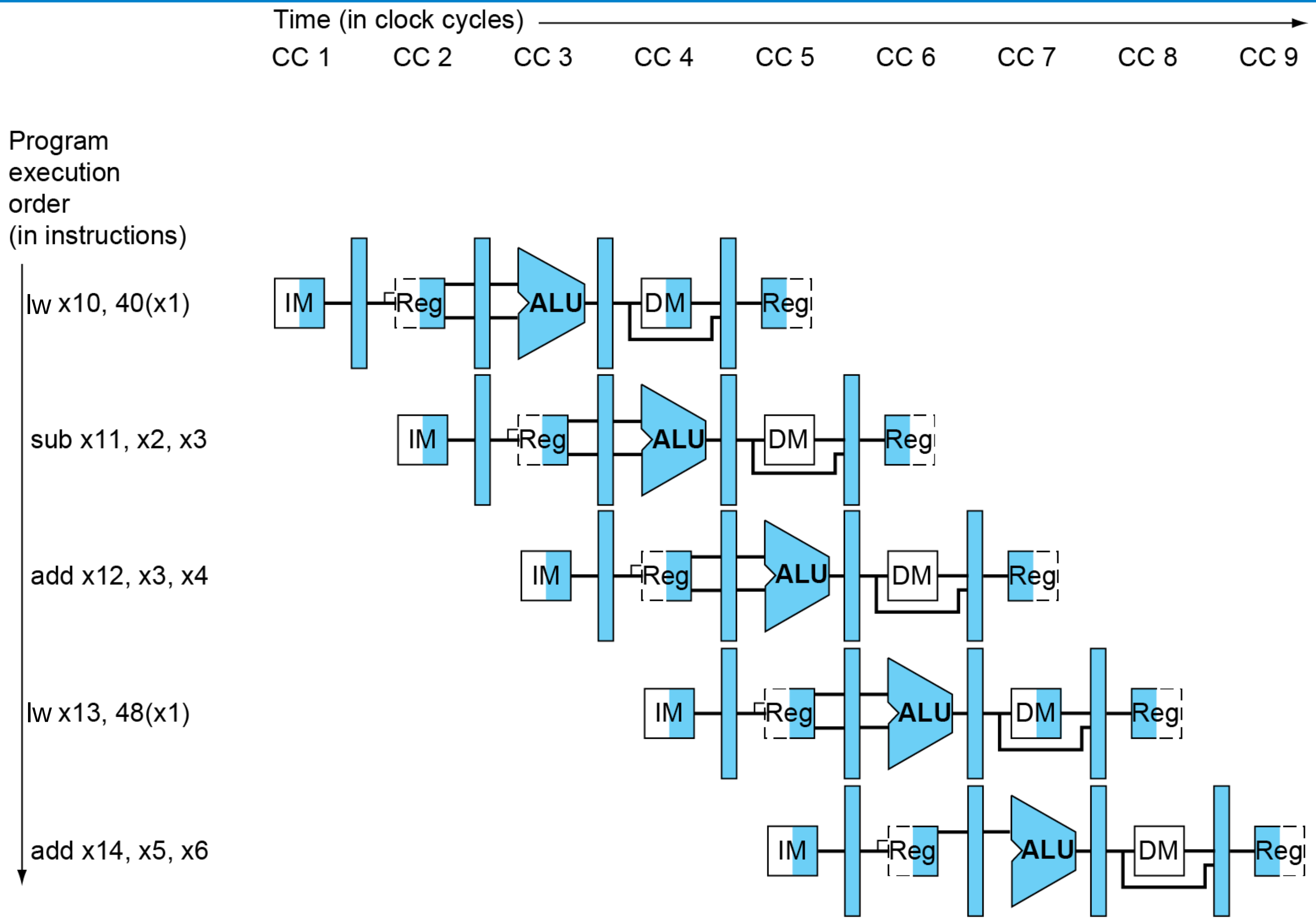
- Need registers between stages
  - To hold information produced in previous cycle



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

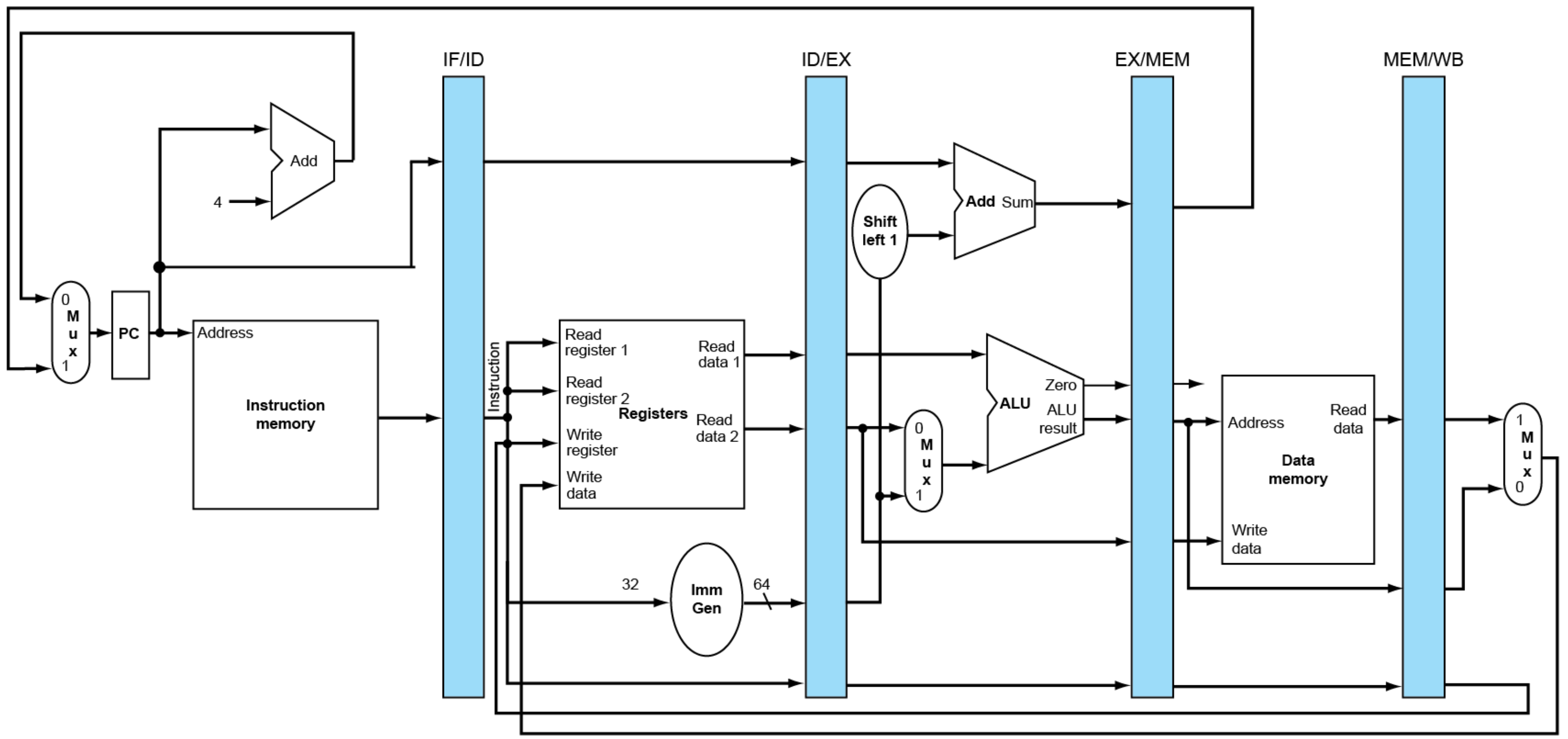
Multi-Cycle Pipeline Diagram



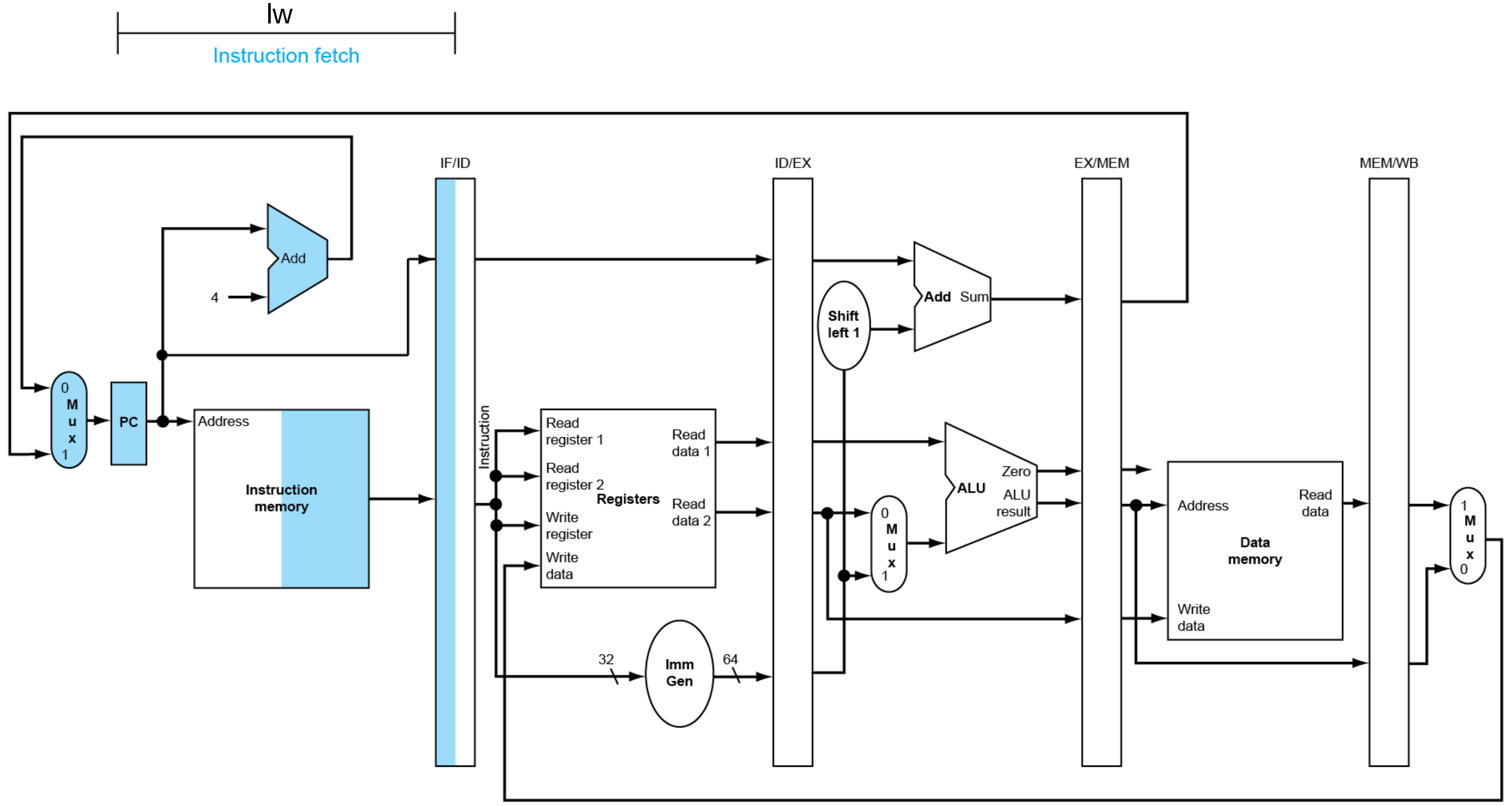
Single-Cycle Pipeline Diagram

- State of pipeline in a given cycle
  - We’ll look at “single-clock-cycle” diagrams for load & store

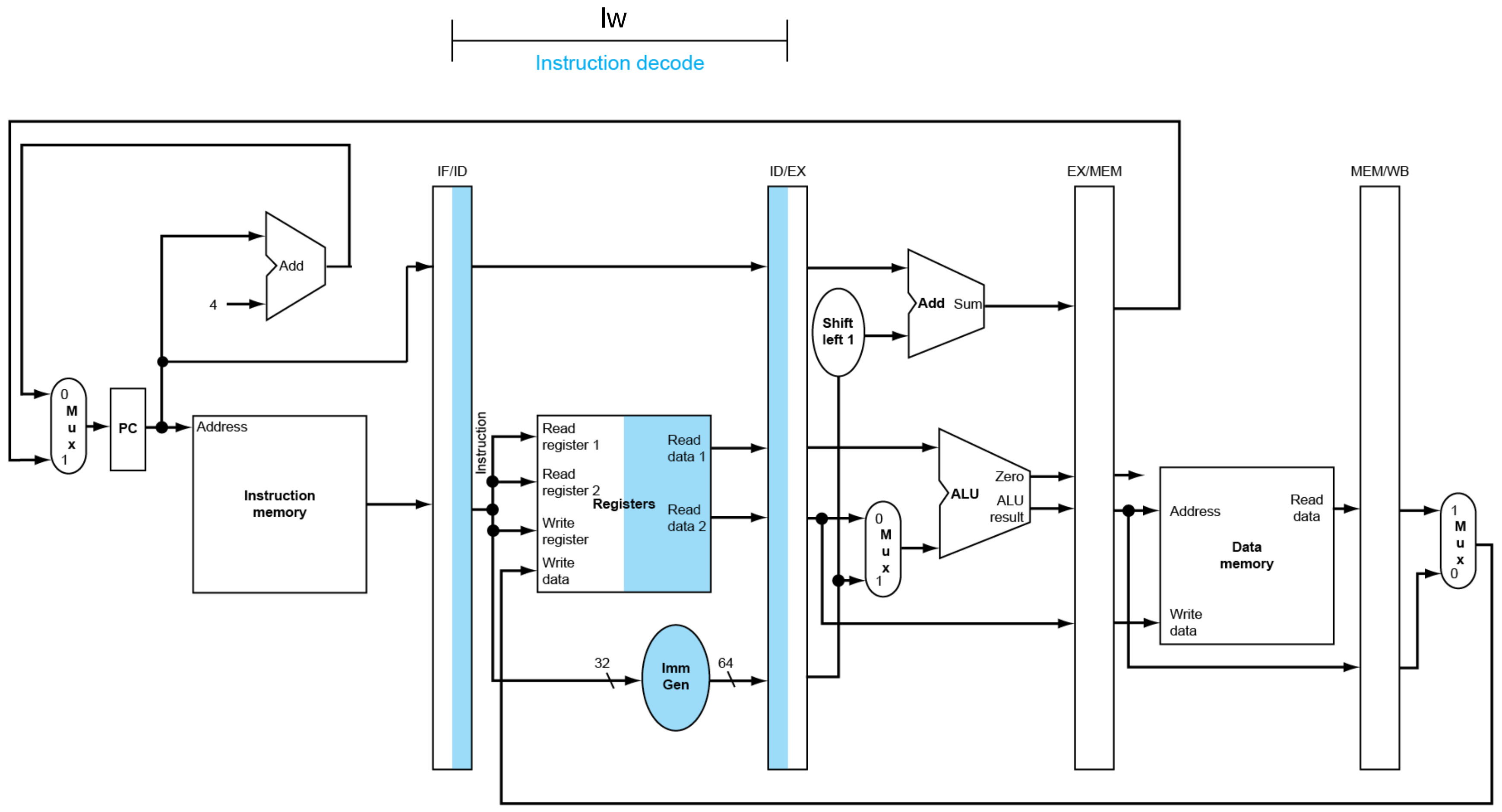
add x14, x5, x6	lw x13, 48(x1)	add x12, x3, x4	sub x11, x2, x3	lw x10, 40(x1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back



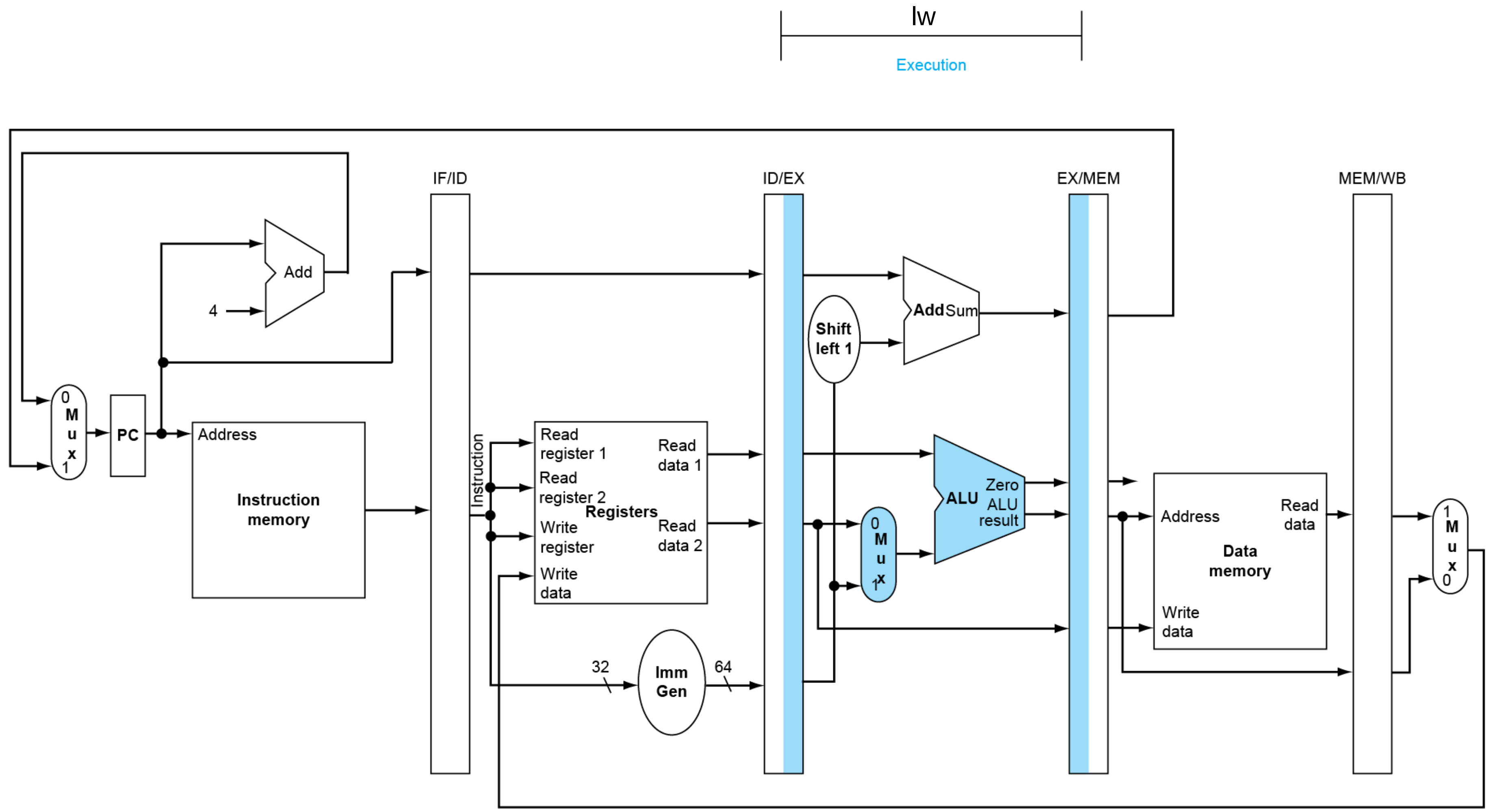
IF for Load, Store, …



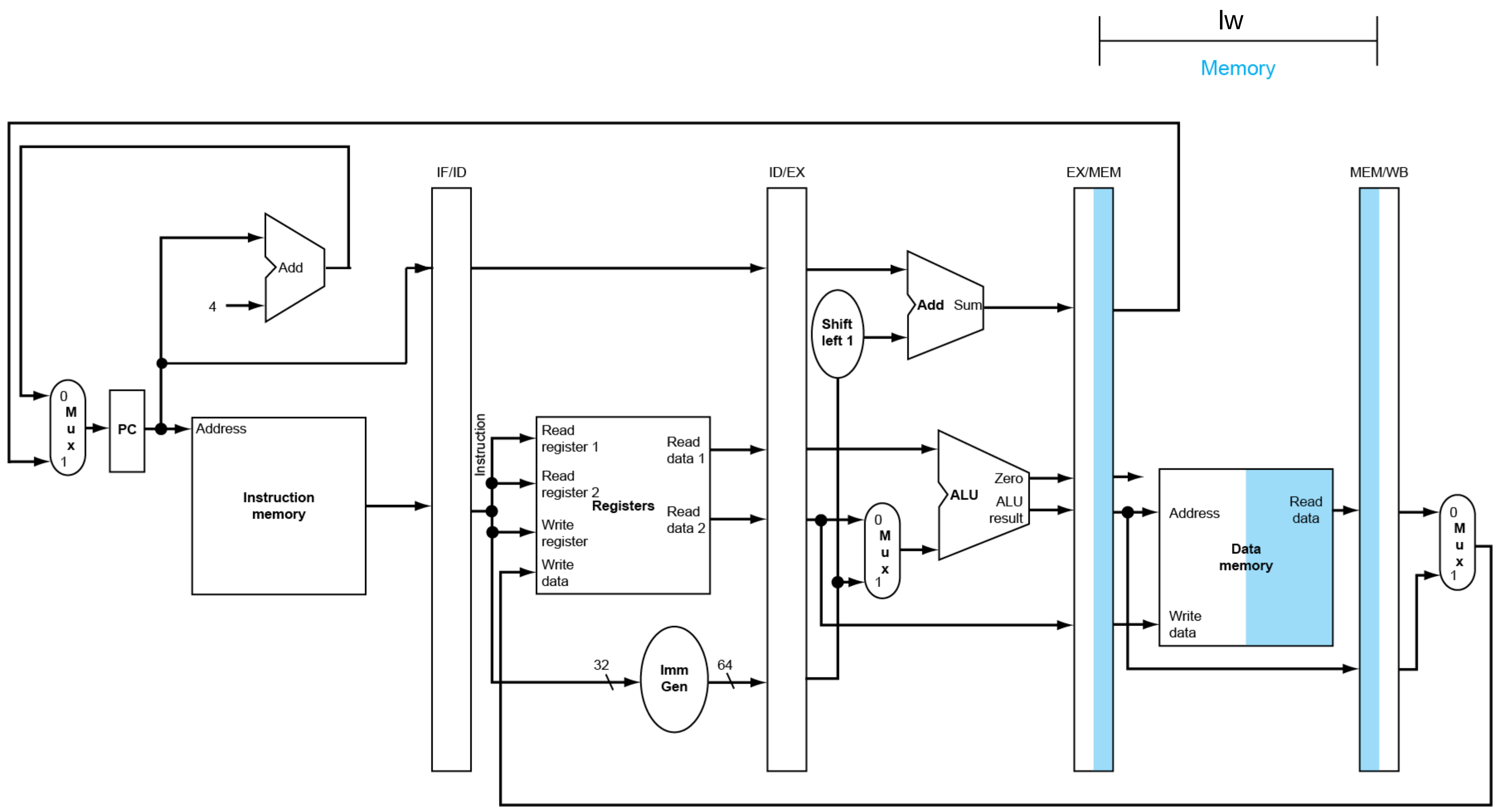
ID for Load, Store, …



EX for Load

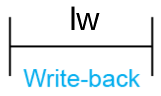

lw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

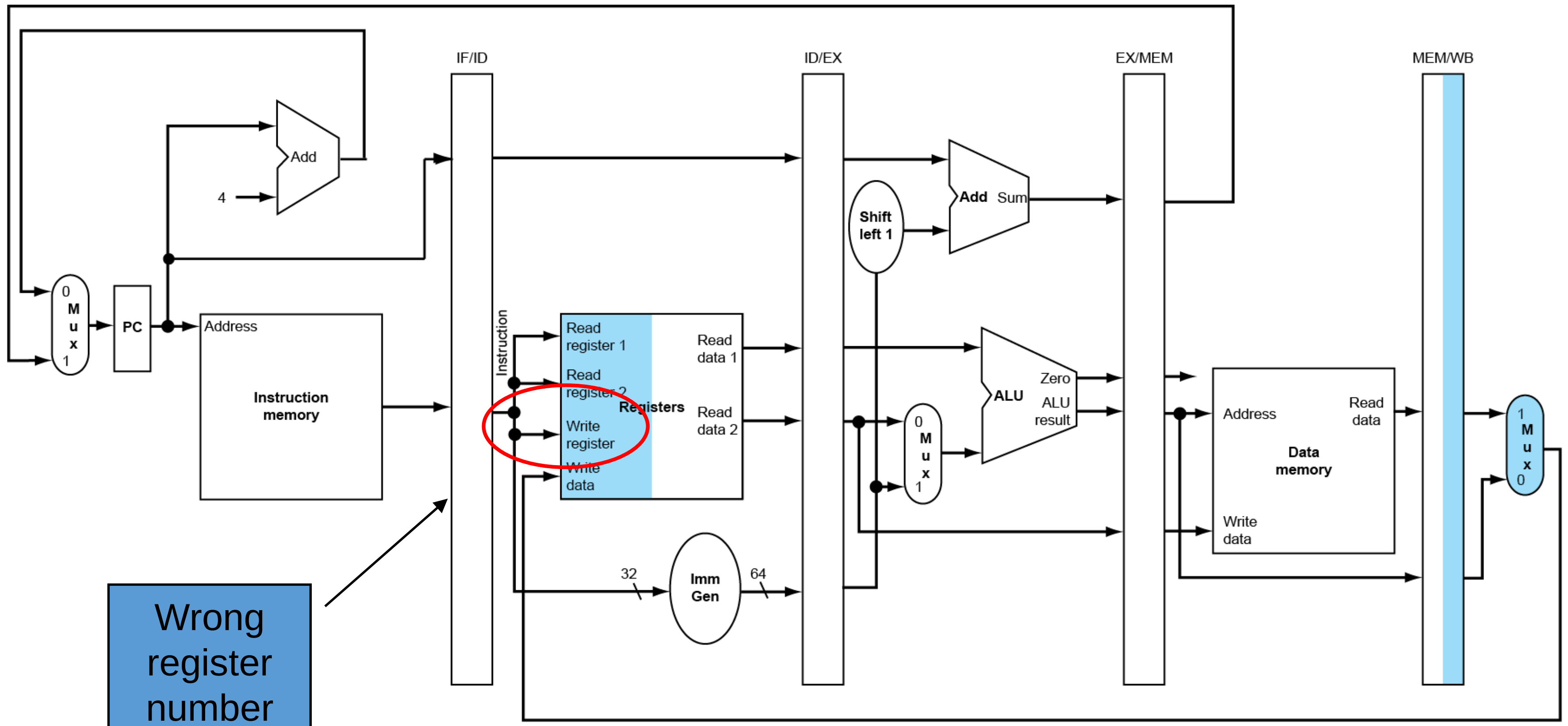
MEM for Load



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
32
Imm Gen
64
Shift left 1
Add Sum
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

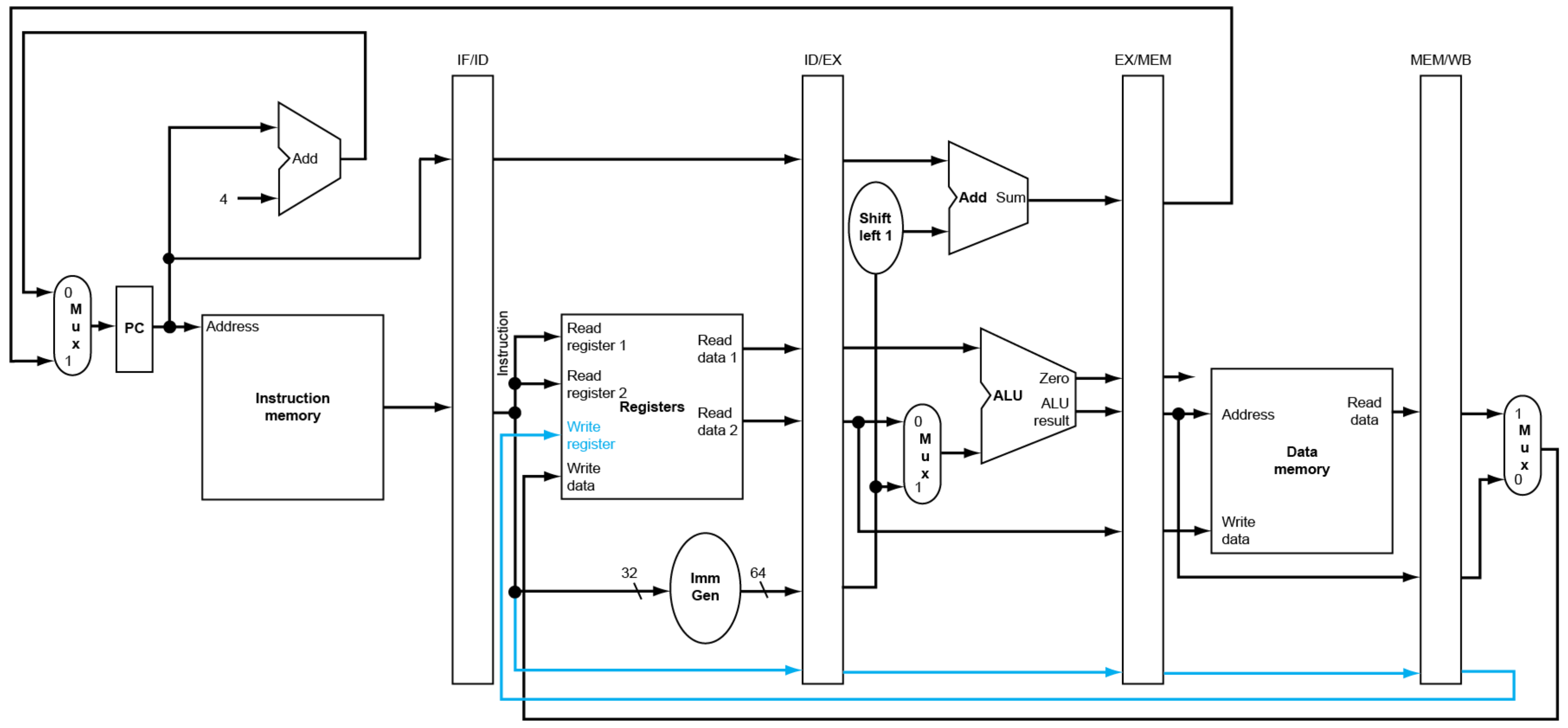
WB for Load





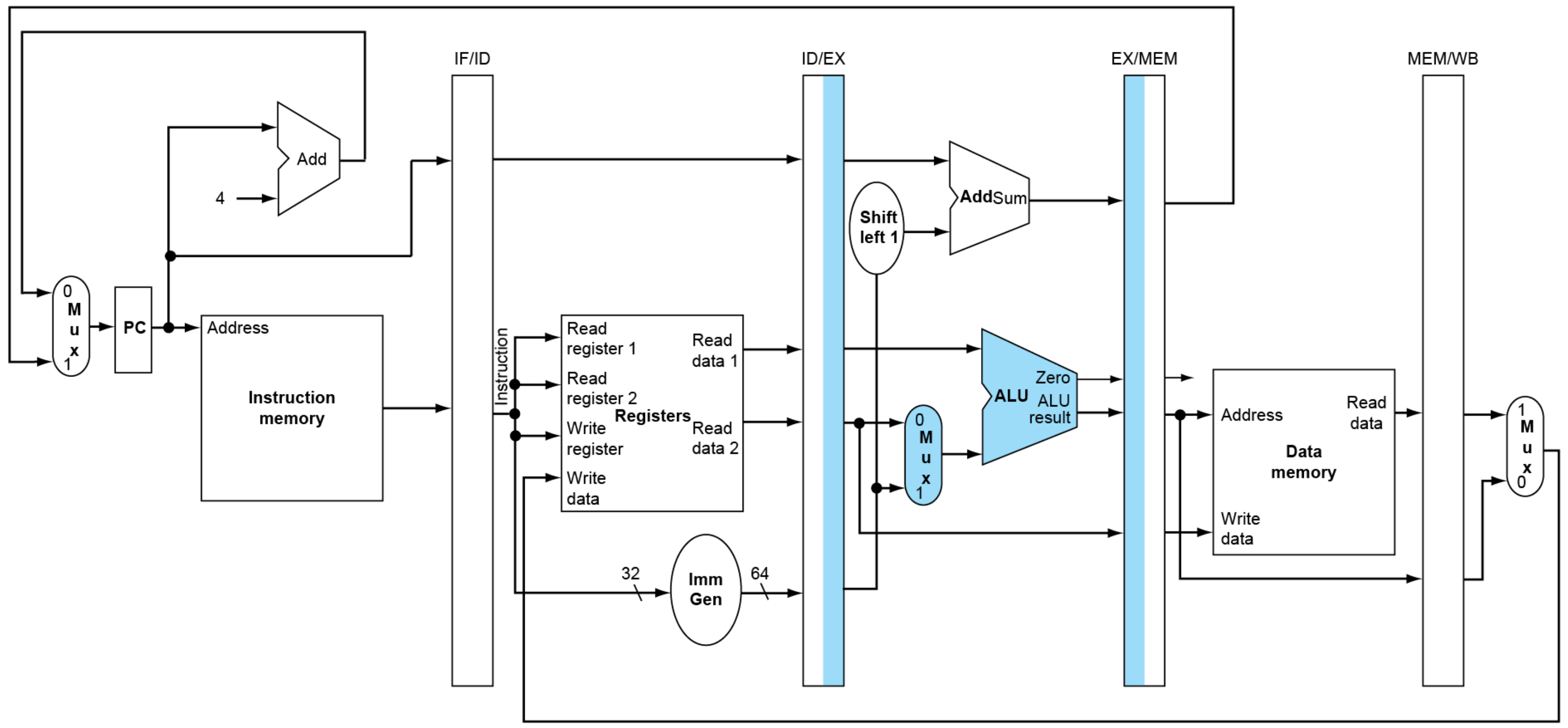
Wrong register number

Corrected Datapath for Load

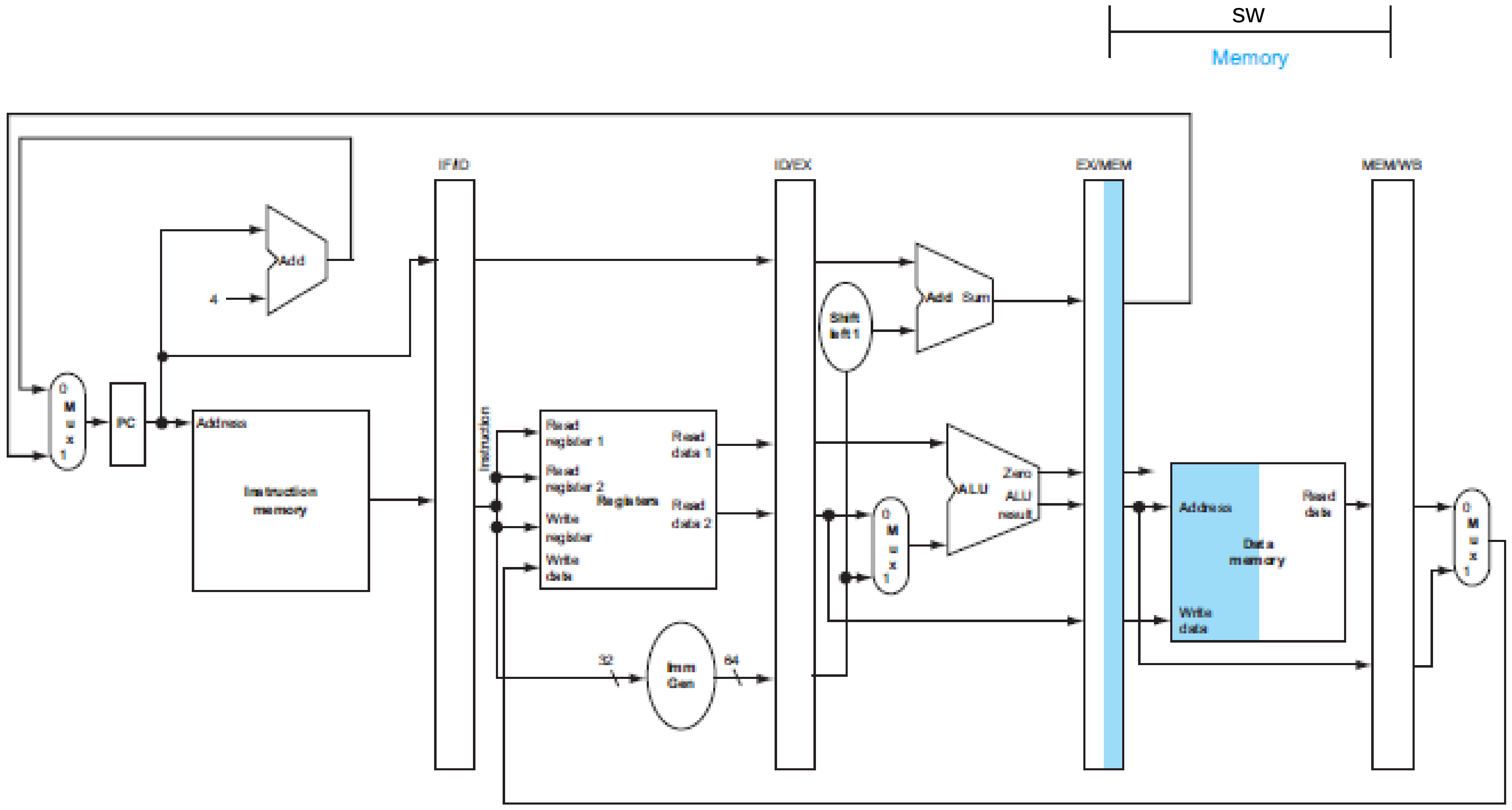


EX for Store

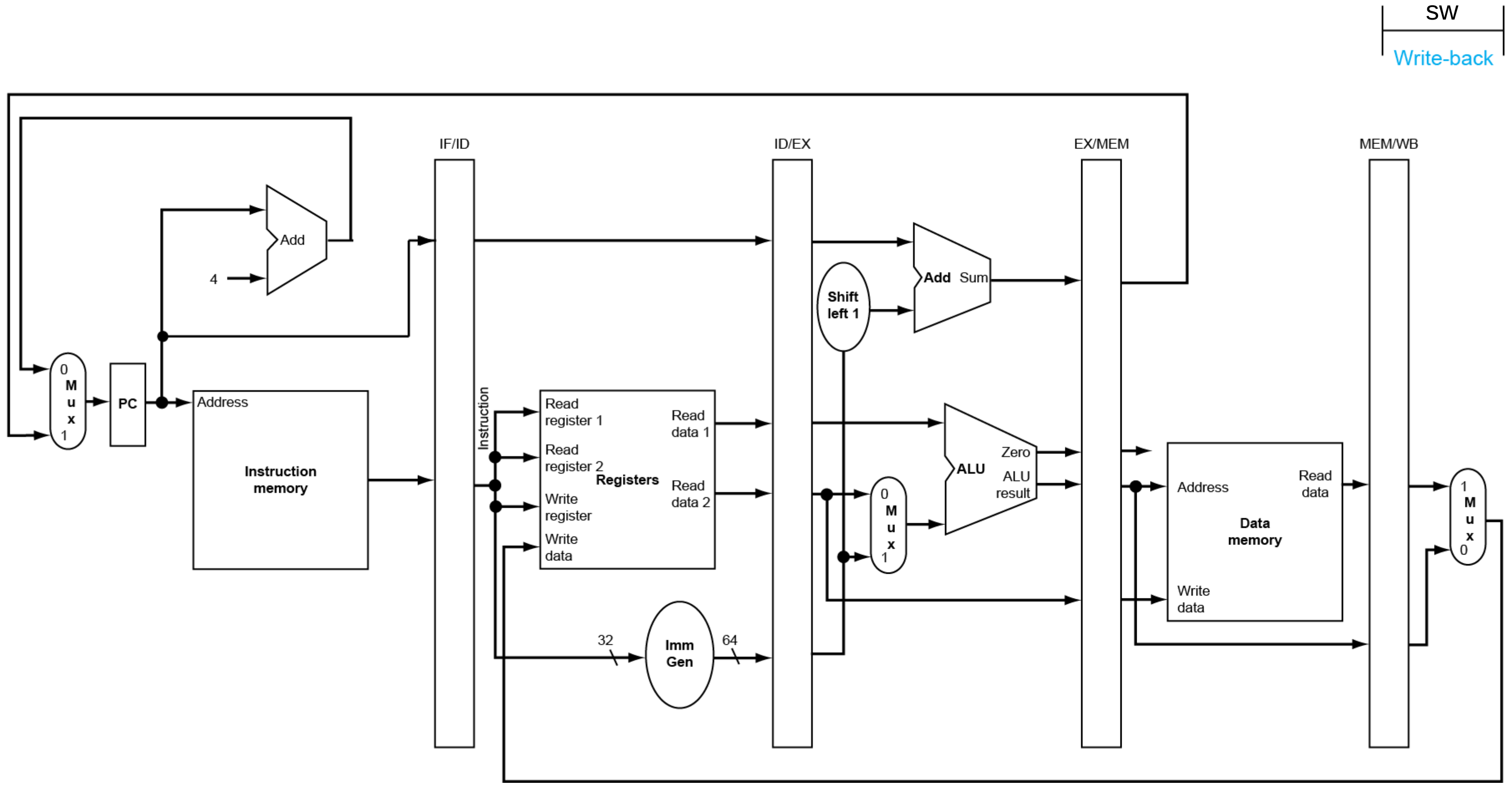
SW
Execution


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
Write register
Write data
Registers
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

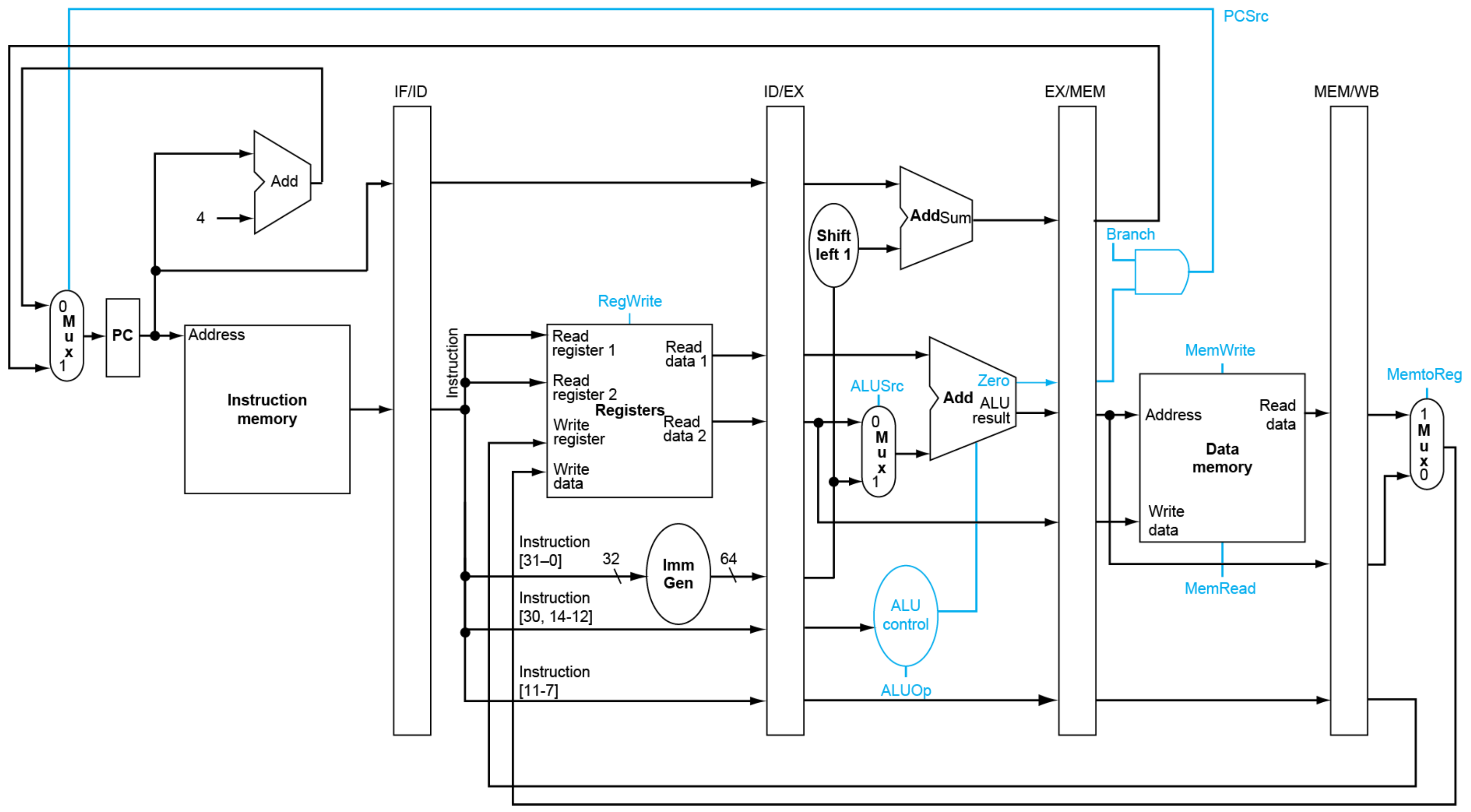
MEM for Store



WB for Store

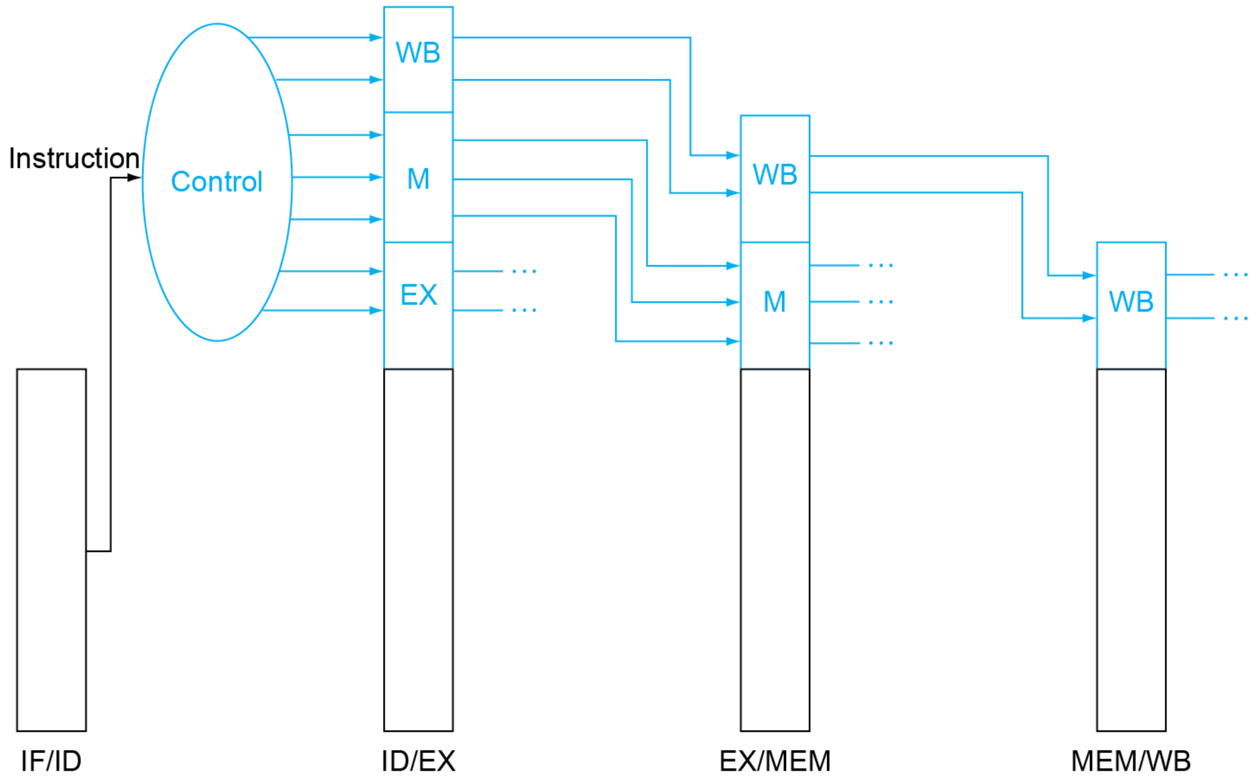

SW
Write-back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

Pipelined Control (Simplified)

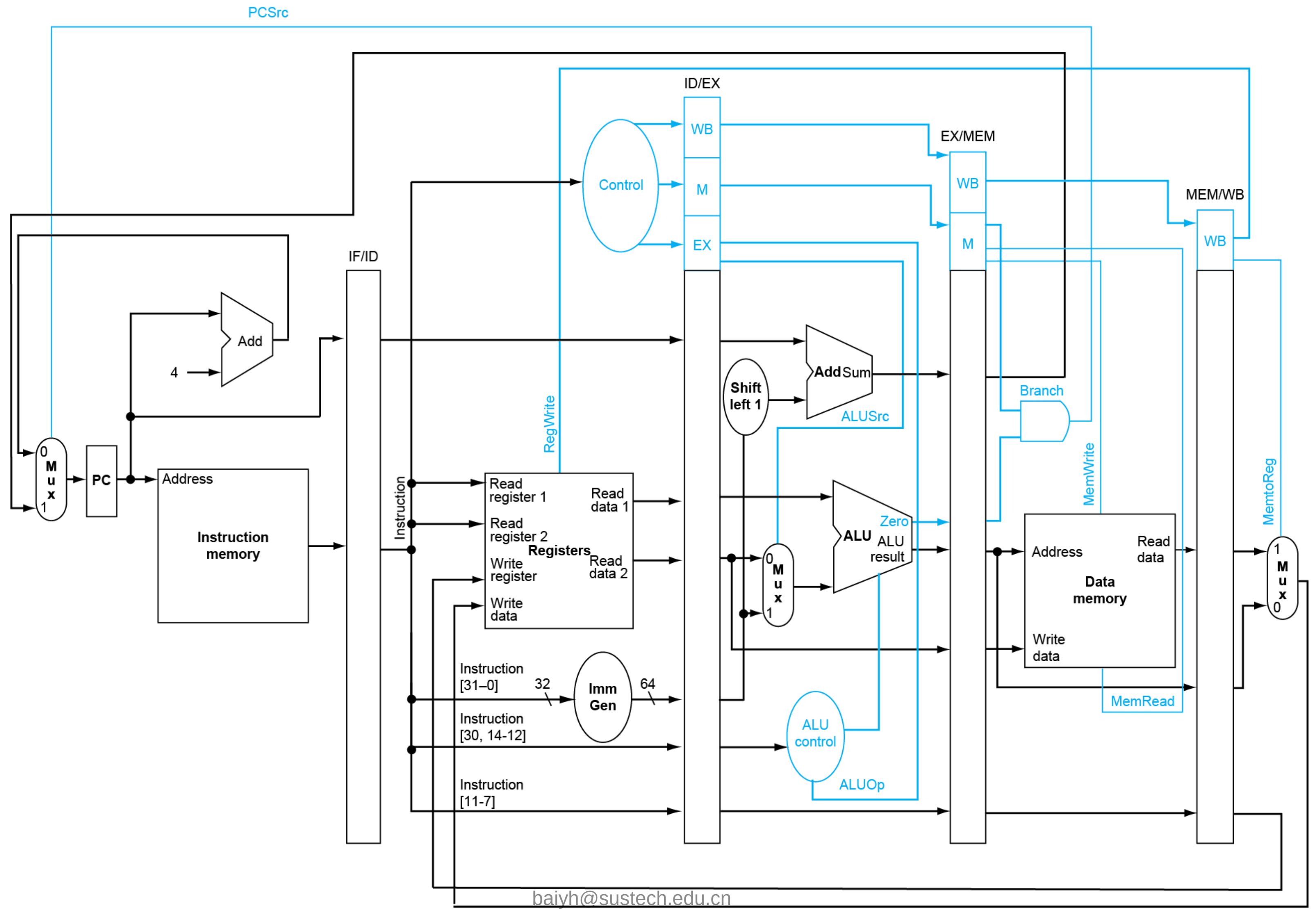


Pipelined Control

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



Pipelined Control



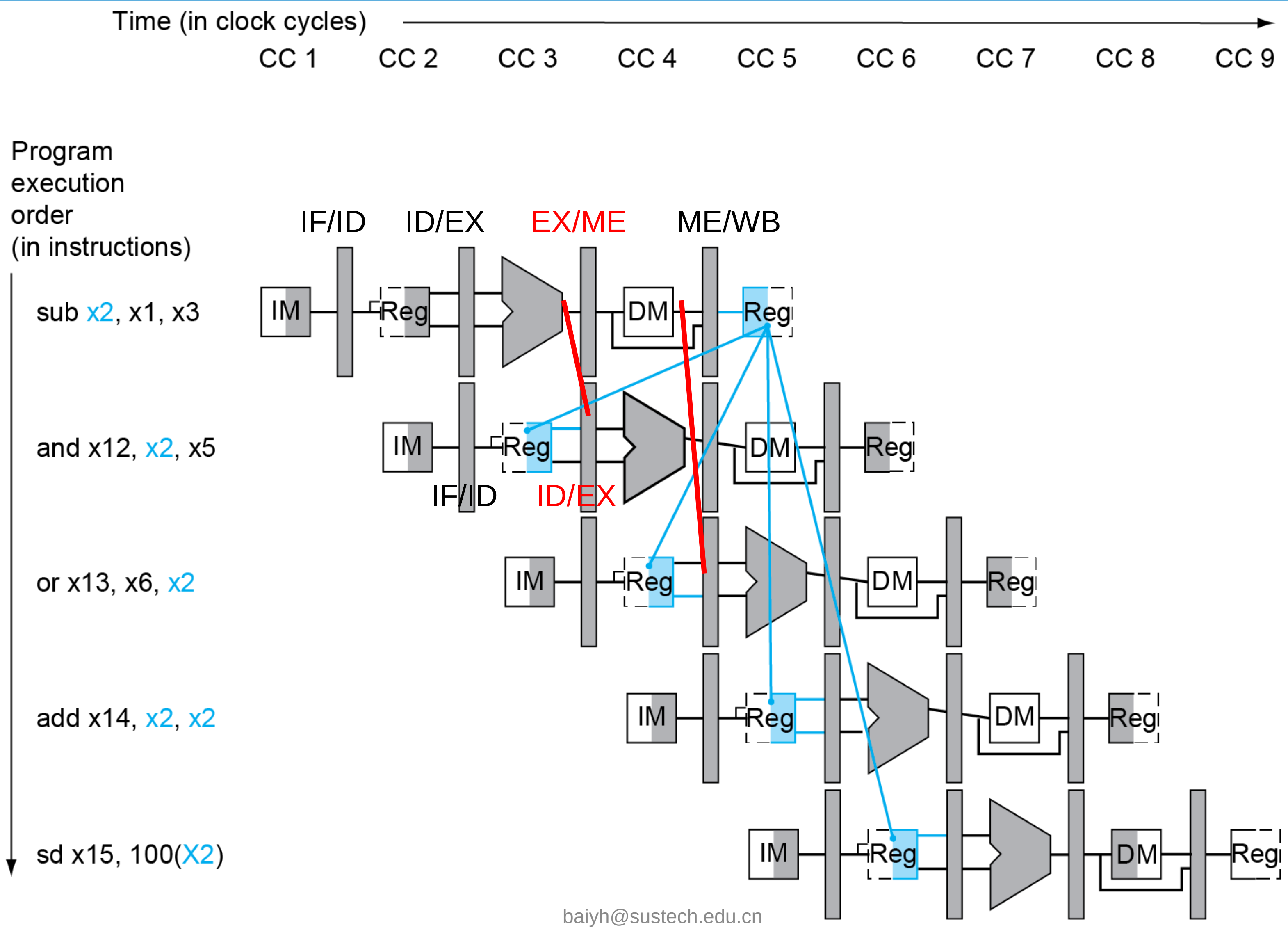
Data Hazards in ALU Instructions

- Consider this sequence:

```
sub   x2, x1,x3
and   x12,x2,x5
or    x13,x6,x2
add   x14,x2,x2
sd    x15,100(x2)
```

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

Dependencies & Forwarding



Detecting the Need to Forward

- Pass register numbers along pipeline
  - e.g., ID/EX.RegisterRs1 = register number for Rs1 sitting in ID/EX pipeline register
- ALU operand register numbers in EX stage are given by
  - ID/EX.RegisterRs1, ID/EX.RegisterRs2
- Data hazards when

  1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
  1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
  2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
  2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2

Fwd from EX/MEM pipeline reg

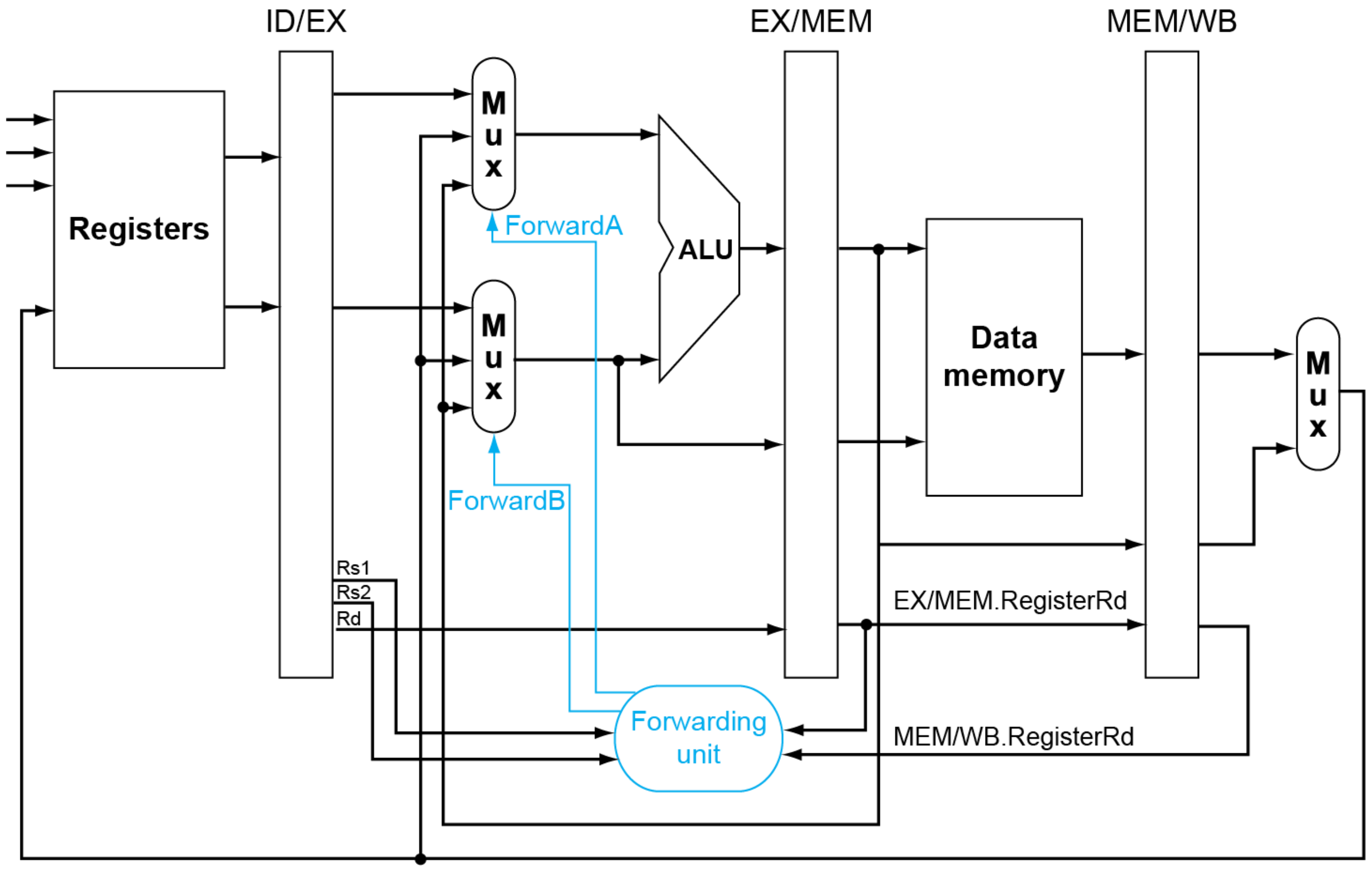
Fwd from MEM/WB pipeline reg

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 x0
  - 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.RegisterRs1))
    - ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)and (EX/MEM.RegisterRd = ID/EX.RegisterRs2))
    - ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)and (MEM/WB.RegisterRd = ID/EX.RegisterRs1))
    - ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)and (MEM/WB.RegisterRd = ID/EX.RegisterRs2))
    - ForwardB = 01