Lecture 15
Pipelined Datapath and Control

School of Computer Science and Engineering
Soongsil University

4. The Processor

Pipelined Version of the Datapath

- **Pipeline register**
  - Separation of the two stages
  - Hold information produced in previous cycle

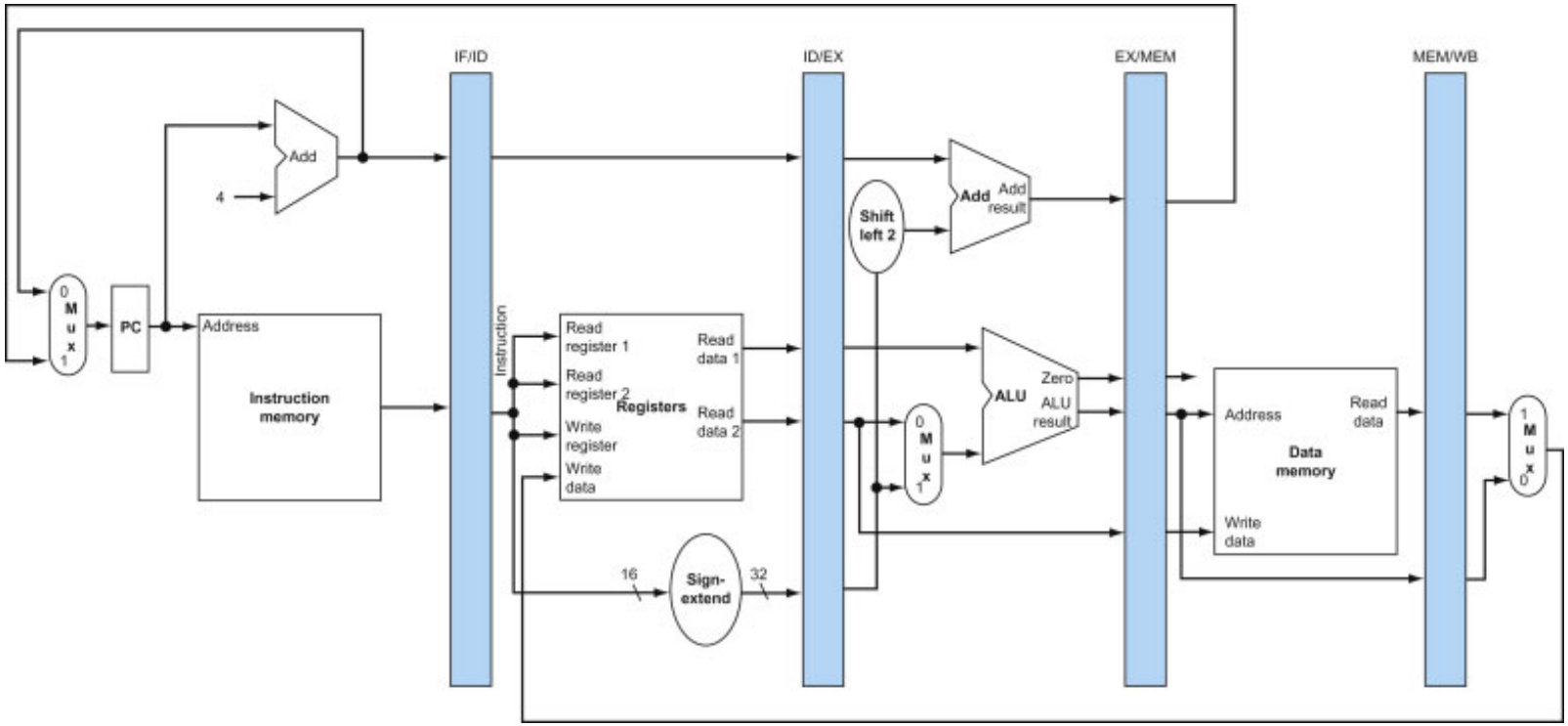

Figure 4.35

lw Instruction in IF Stage

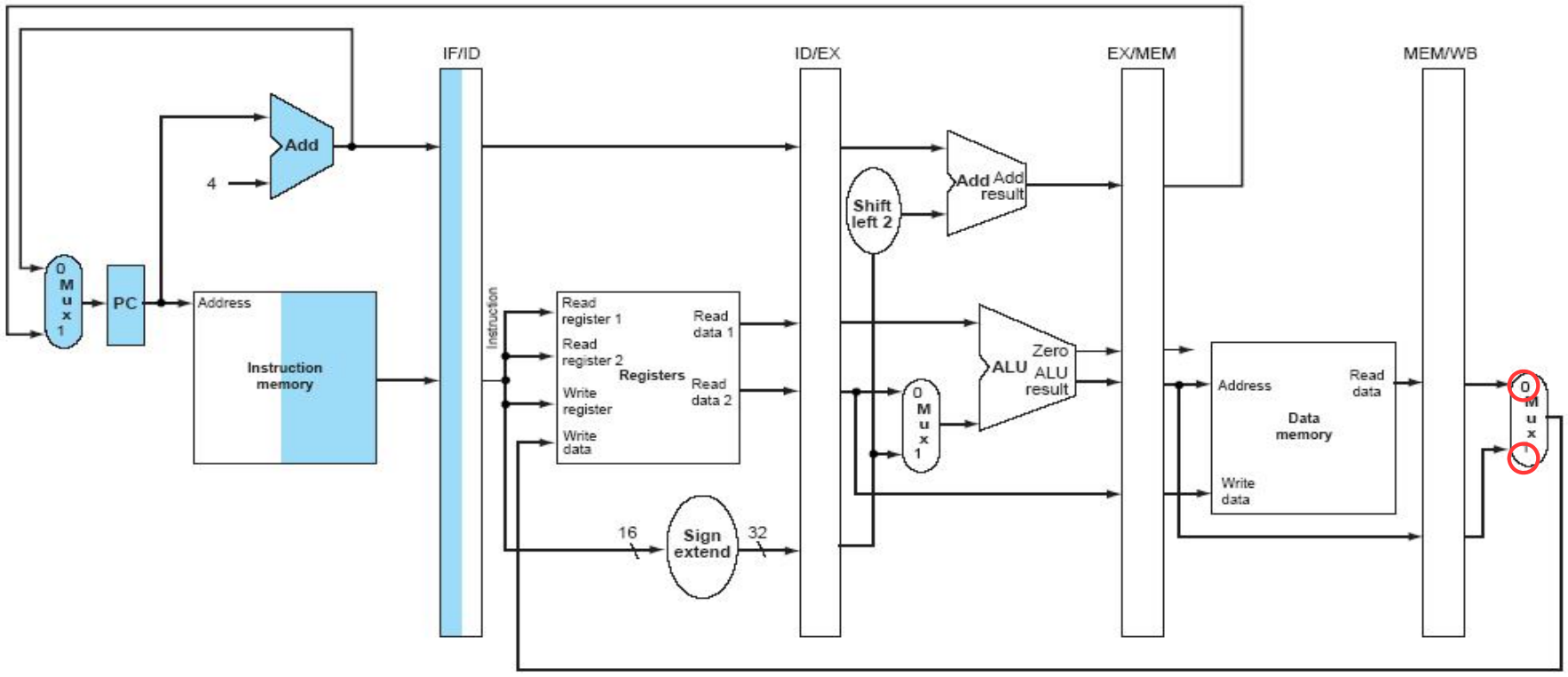

Figure 4.36(a)

lw Instruction in ID Stage

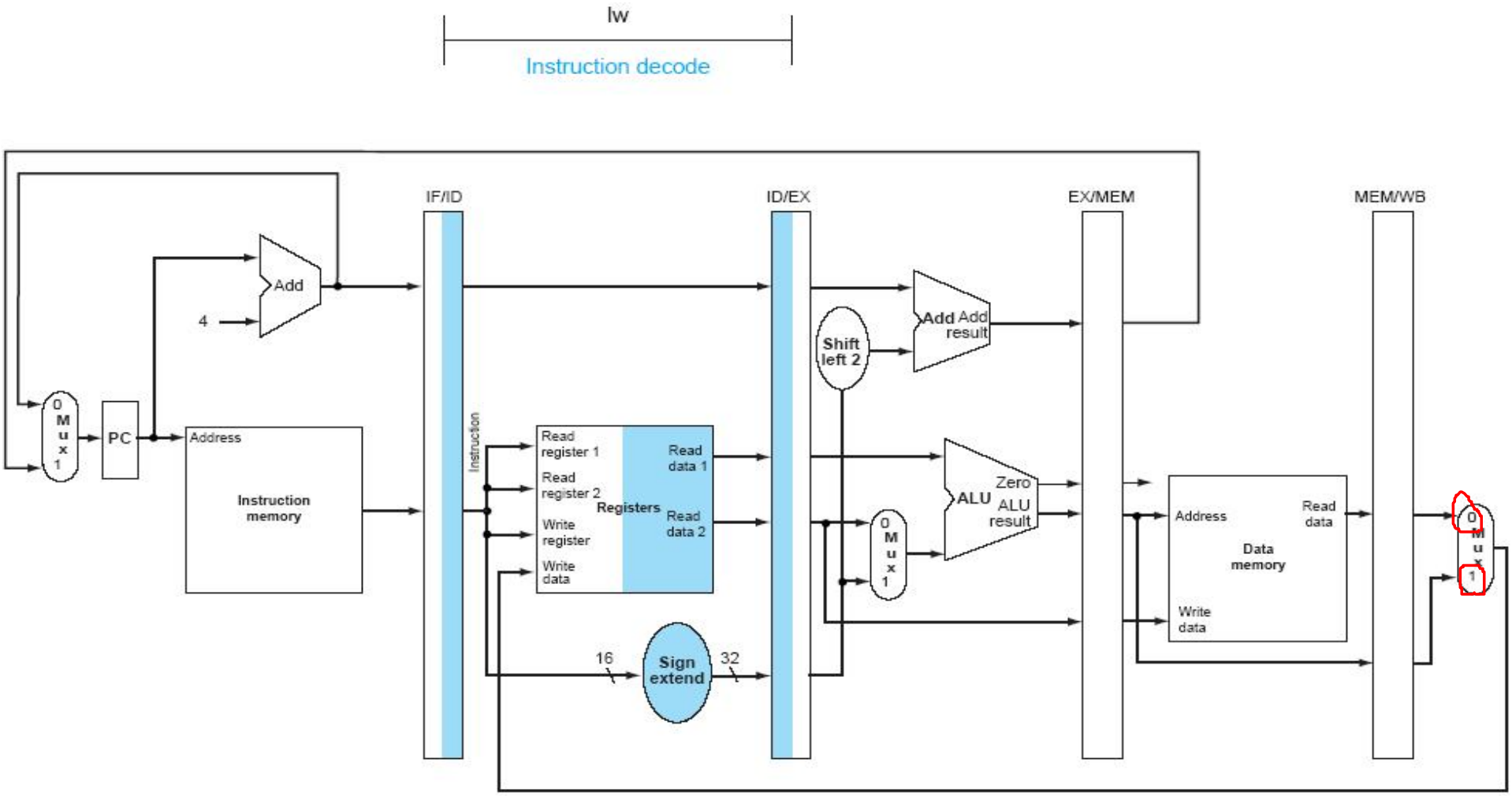

Figure 4.36(b)

lw Instruction in EX Stage

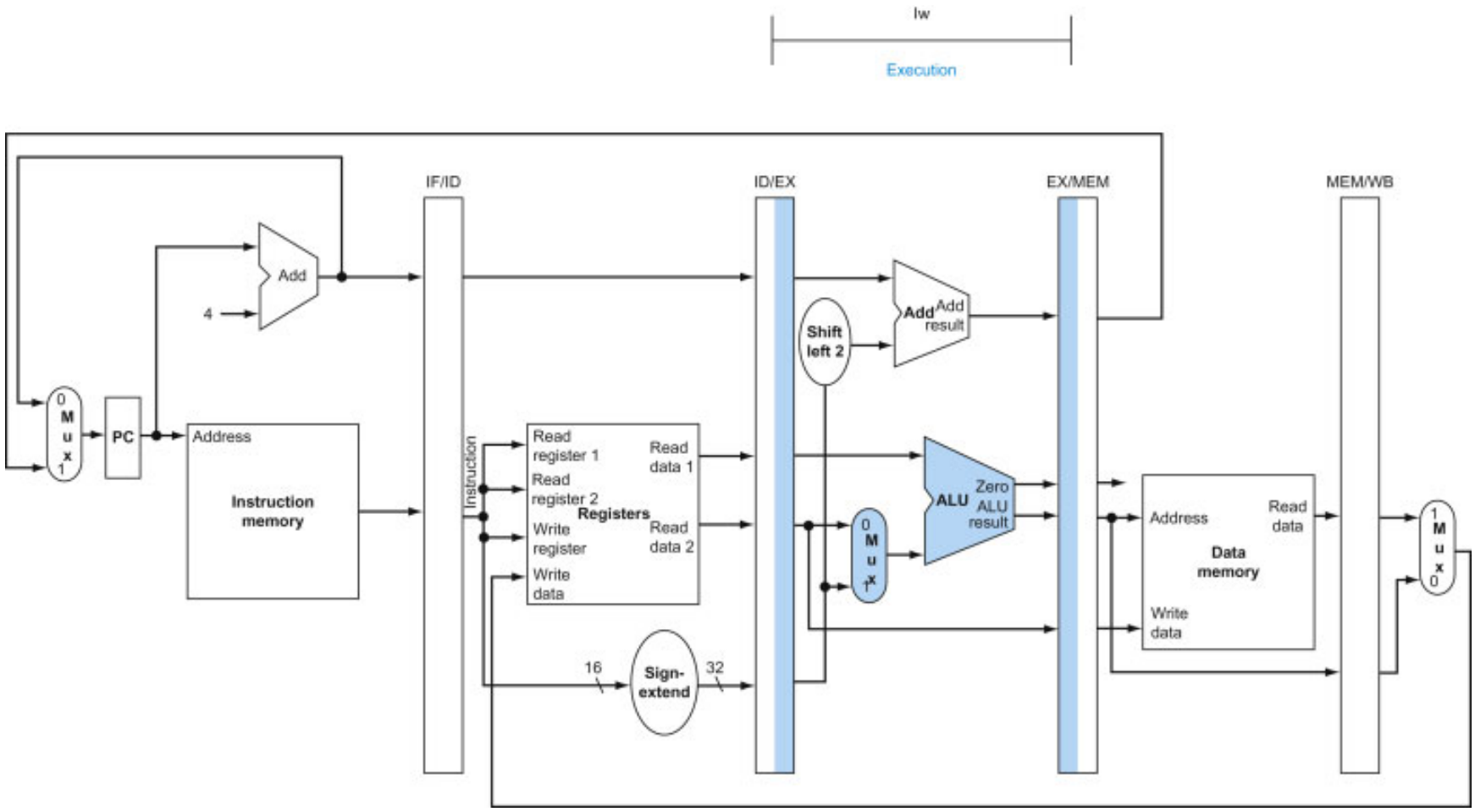

Figure 4.37

lw Instruction in MEM Stage

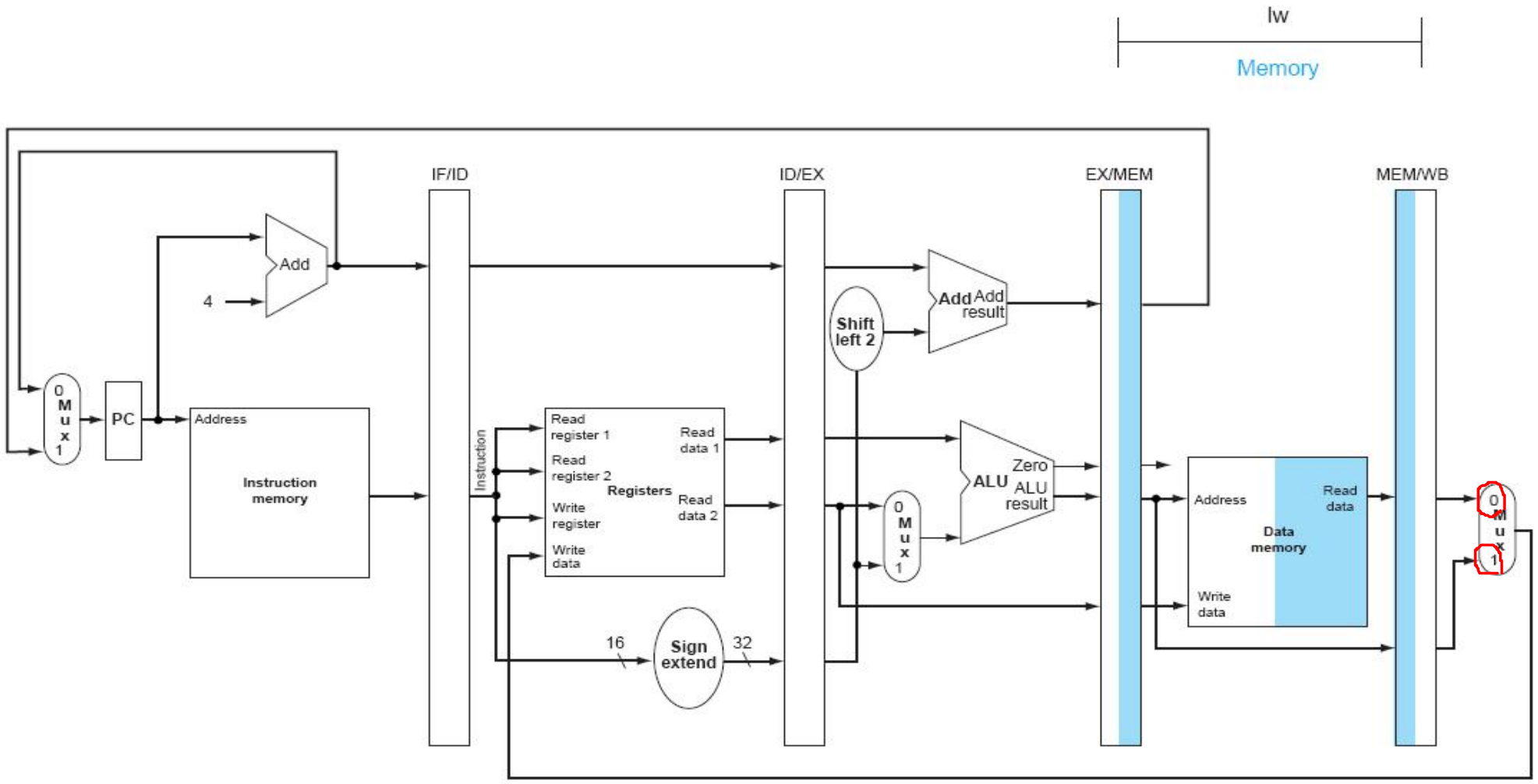

Figure 4.38(a)

lw Instruction in WB Stage

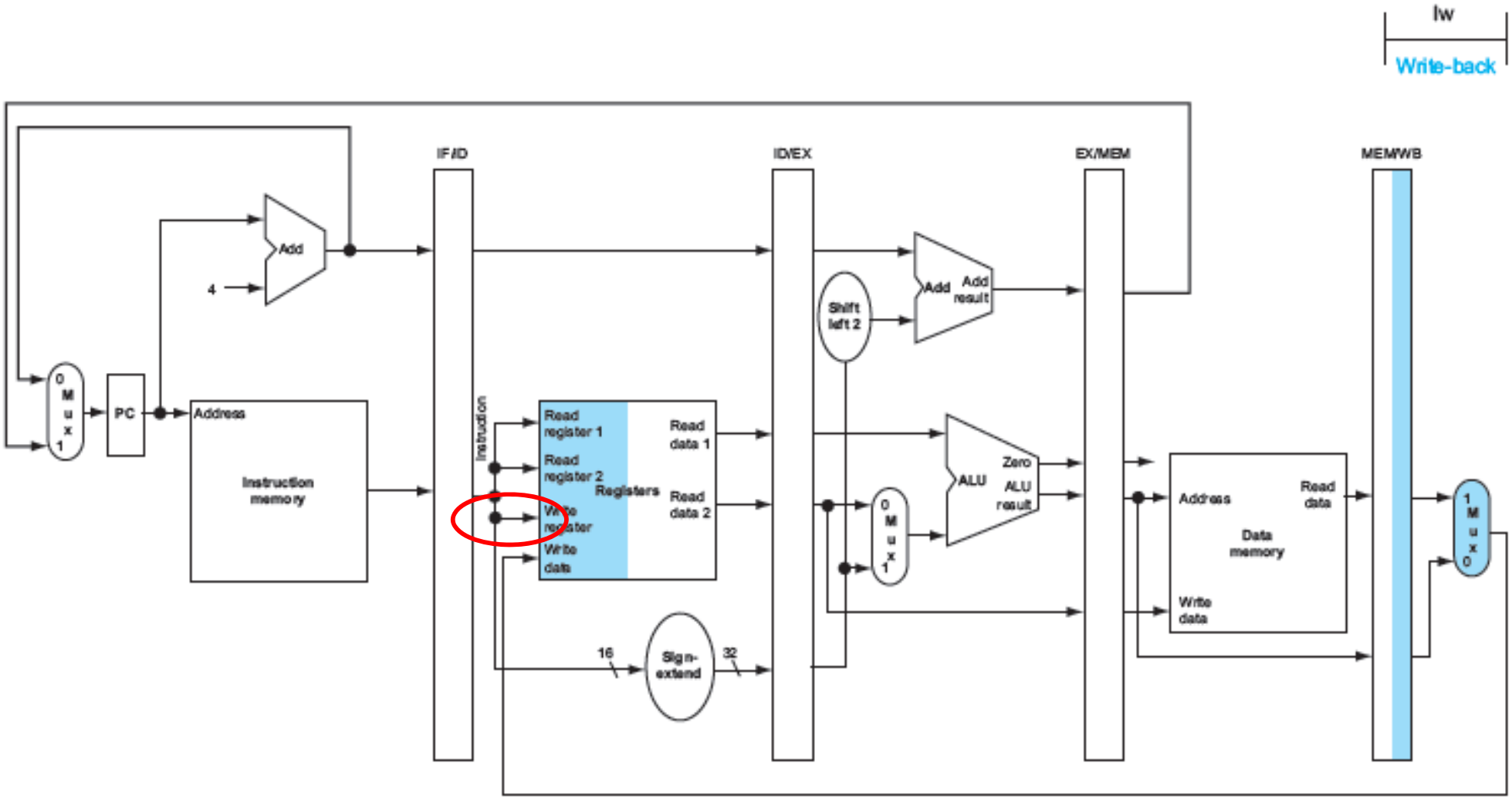

Figure 4.38(b)

Corrected Pipelined Datapath

- **`load` instruction**
  - Write register number: Should be preserved until WB stage

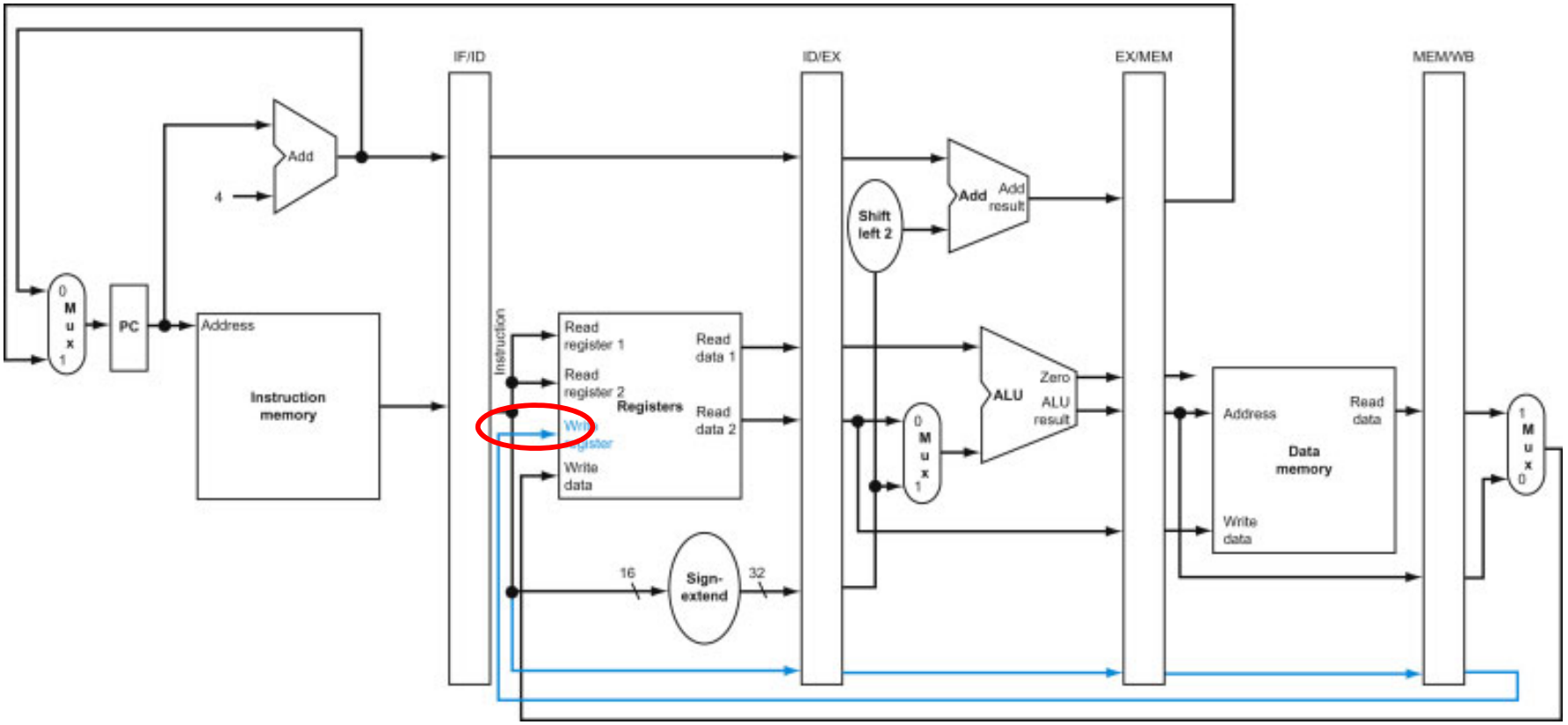

Figure 4.41

`sw` Instruction in EX Stage

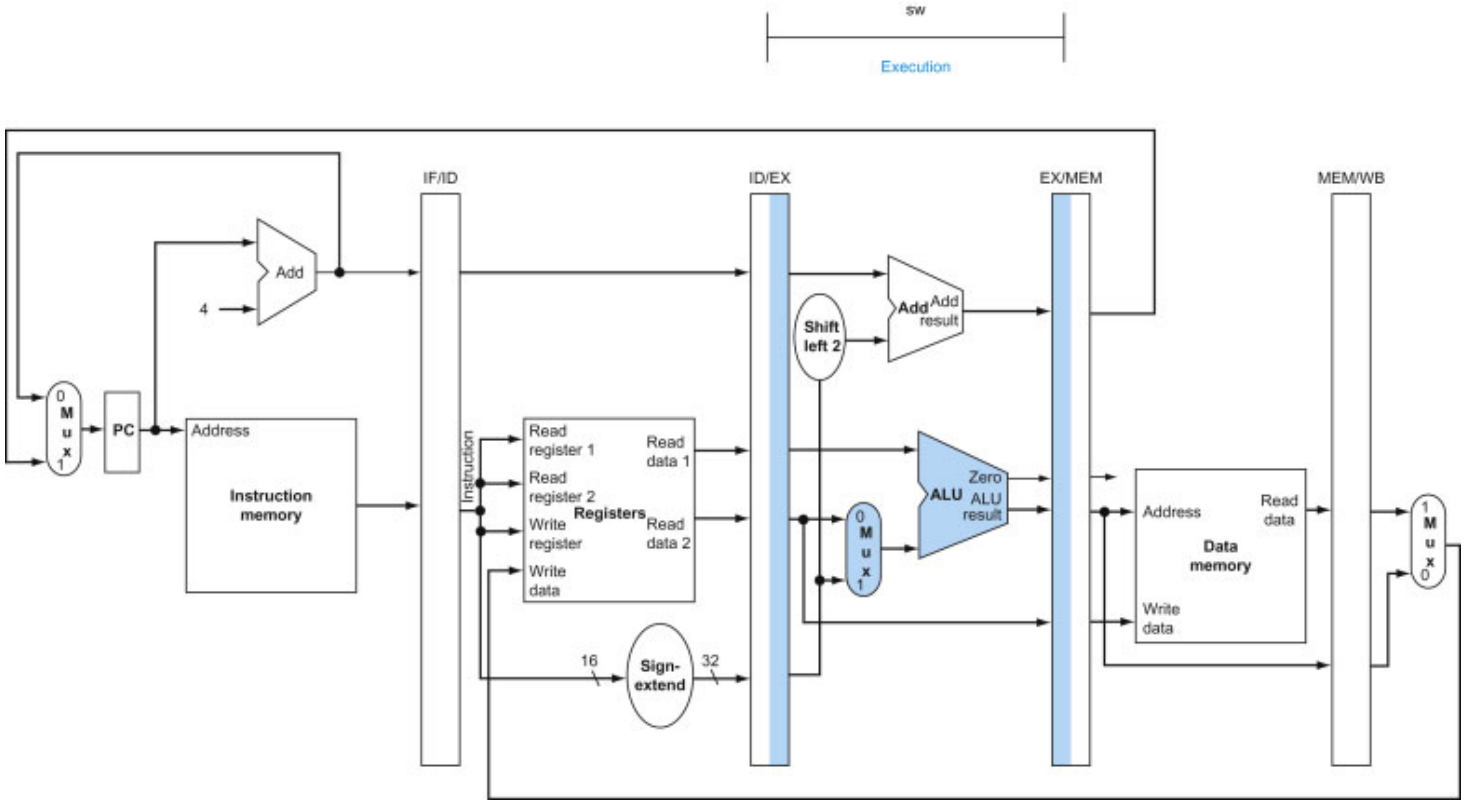

Figure 4.39

sw Instruction in MEM Stage

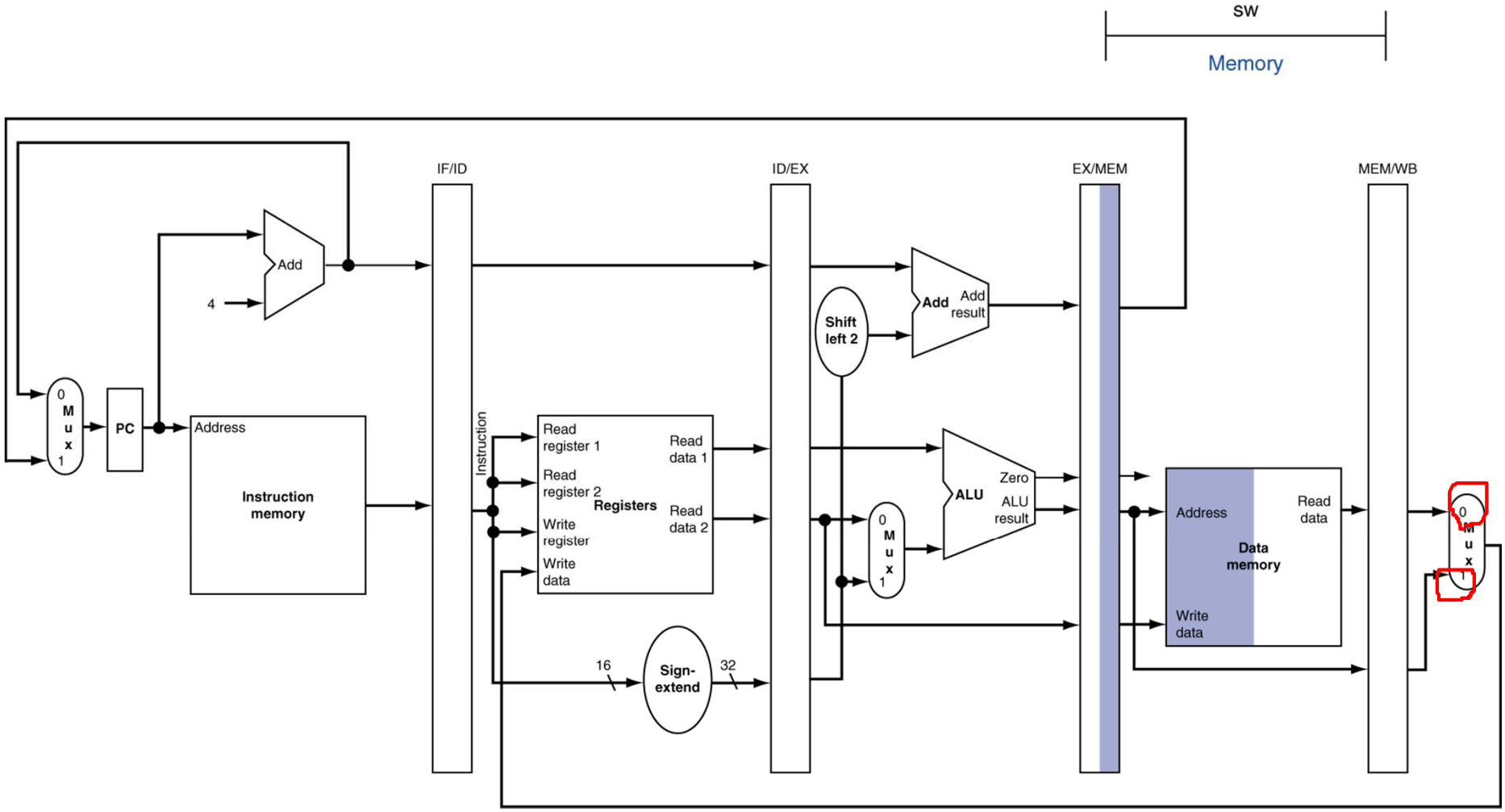

Figure 4.40 (a)

`sw` Instruction in WB Stage

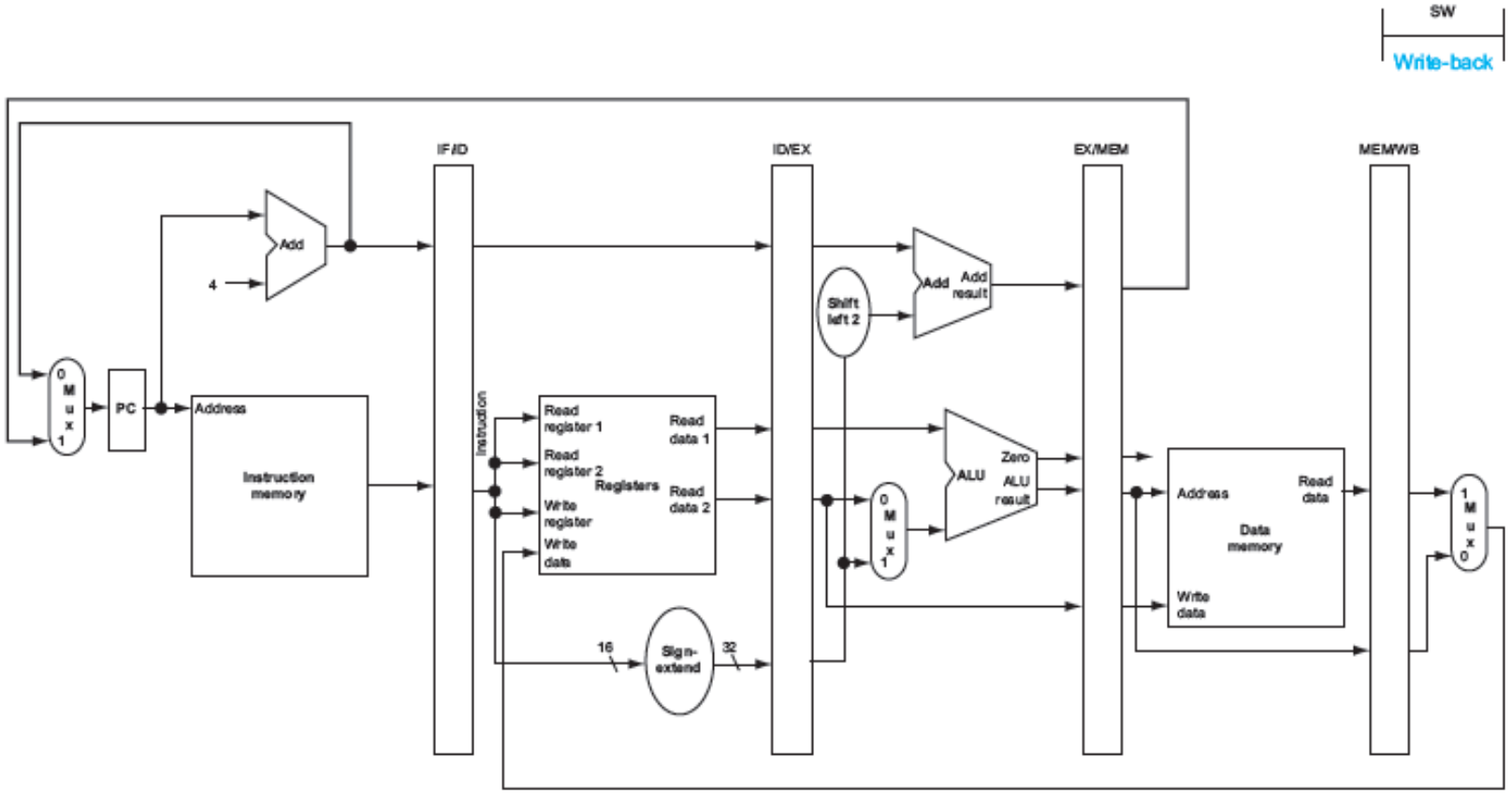

Figure 4.40 (b)

Pipelined Control

- **Pipelined datapath with control signals**

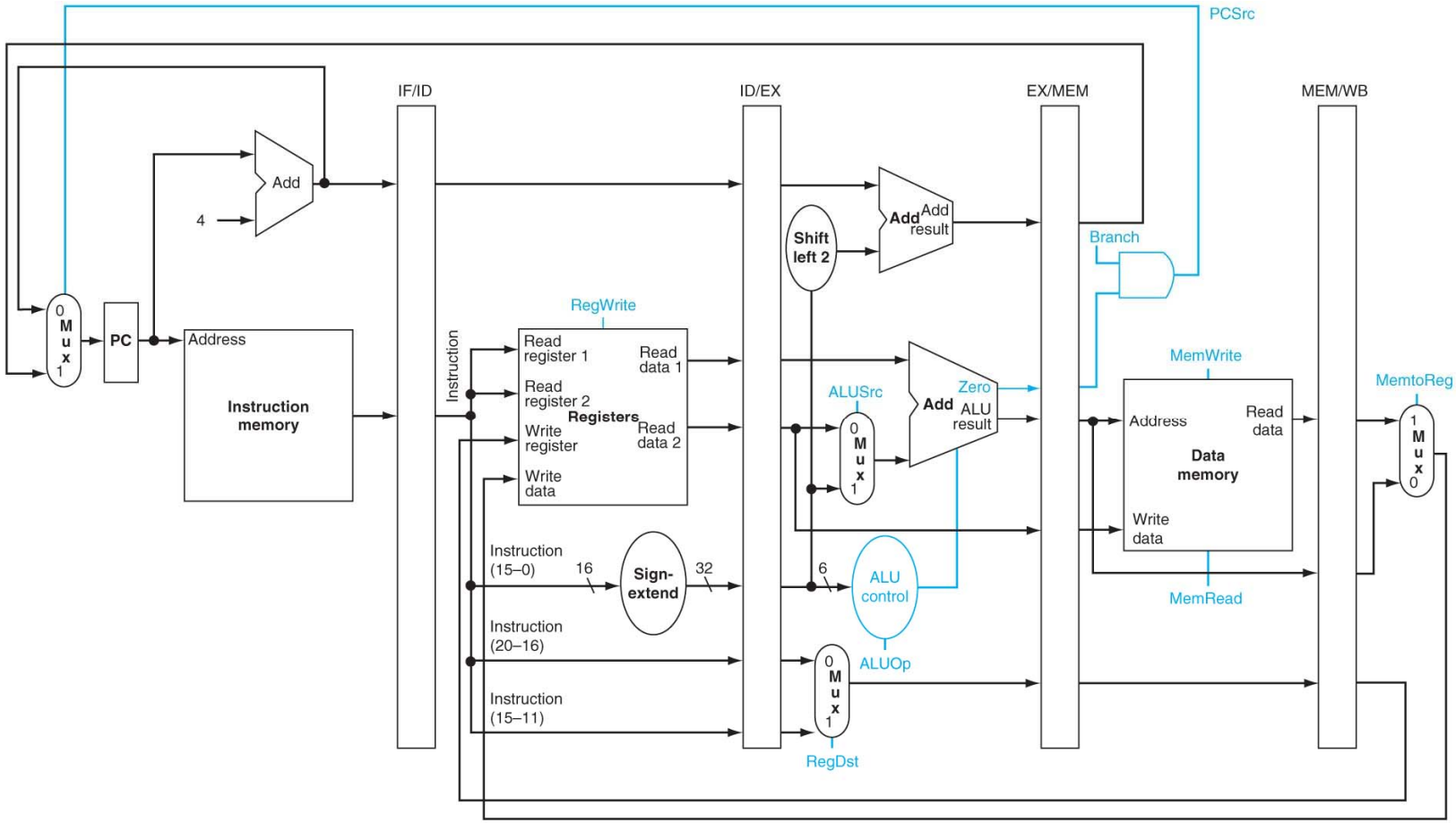

Figure 4.46

Control Signals

1. IF : nothing (the same thing happens at every clock cycle)
2. ID : nothing
3. EX : RegDst, ALUOp, ALUSrc
4. MEM : Branch(for `beq`), MemRead(for `lw`), MemWrite(for `sw`)
5. WB : MemtoReg, RegWrite

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
`lw`	0	0	0	1	0	1	0	1	1
`sw`	X	0	0	1	0	0	1	0	X
`beq`	X	0	1	0	1	0	0	0	X

Figure 4.49

Control Signals for Each Stage

- Control signals derived from instruction

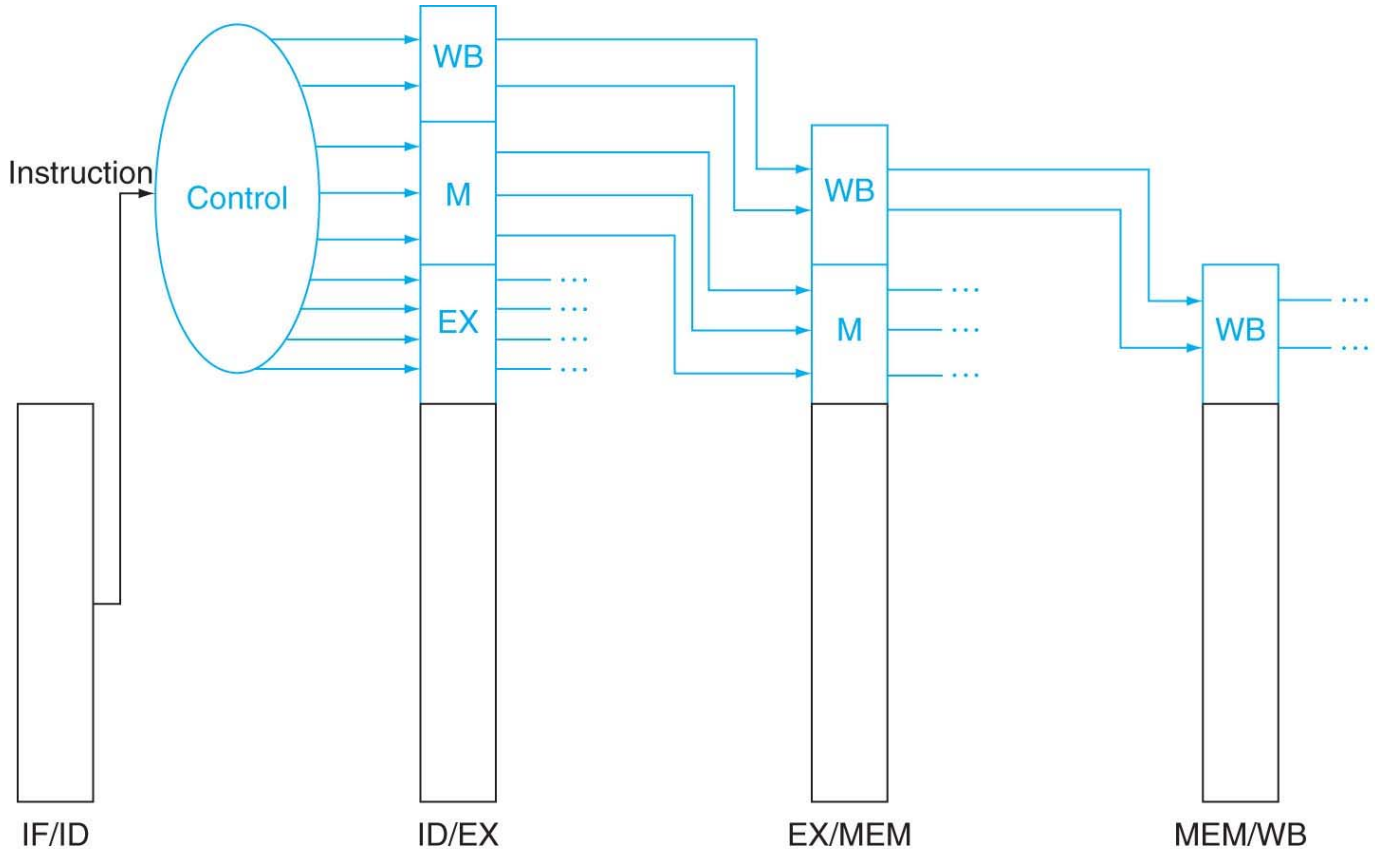

Figure 4.50

Complete Pipelined Datapath

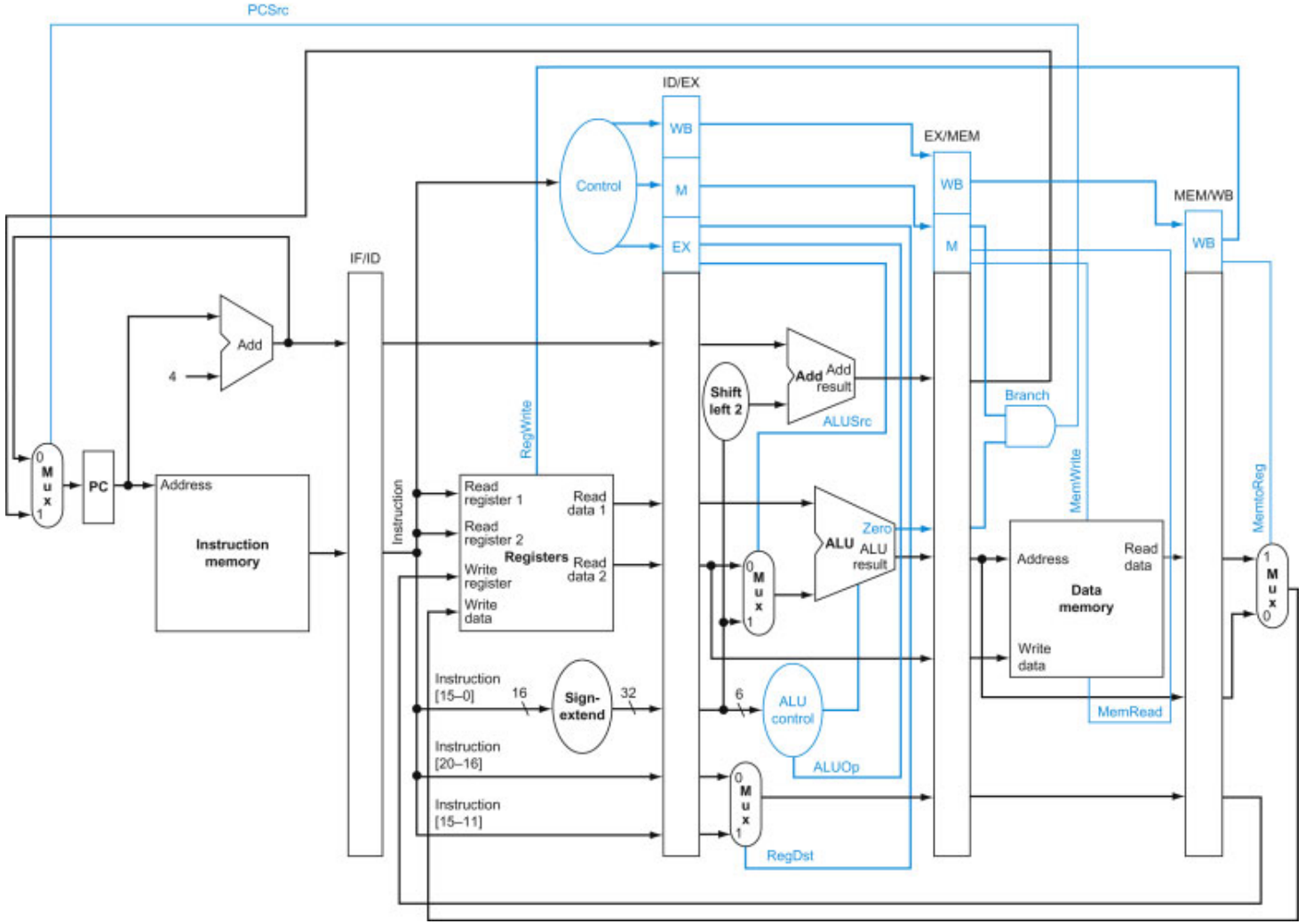

Figure 4.51

Example: Complete Pipelined Datapath

```
1000: lw $1, 81($2)
1004: add $3,$4,$5
1008: sub $6,$7,$8
1012: slt $9,$10,$11
1016: beq $12,$13, XX

$r = r+1
M[x] = x*2
```

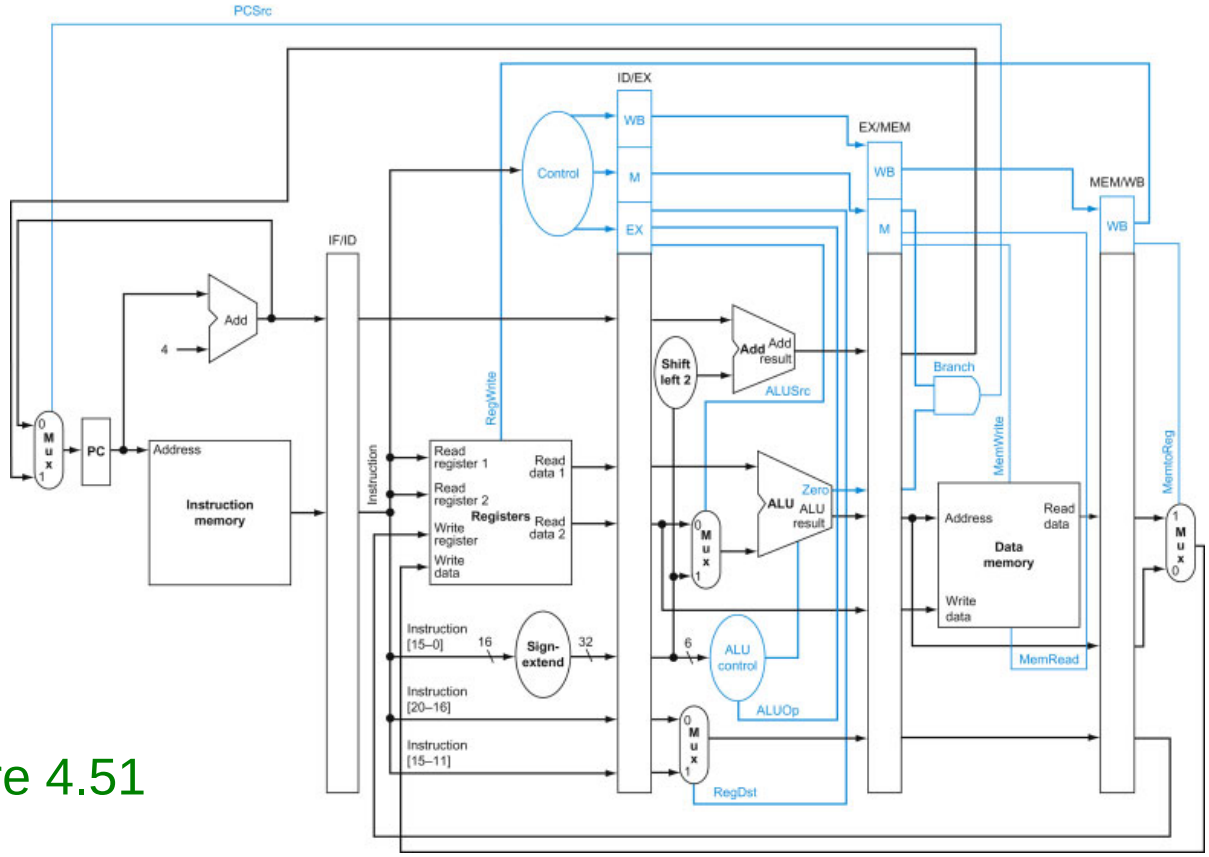

Figure 4.51

Structural Hazards

- Conflict for use of a resource
- In MIPS pipeline with a single memory
  - Load/store requires data access
  - Instruction fetch would have to stall for that cycle
    - Would cause a pipeline “bubble”
- Hence, pipelined datapaths require separate instruction/data memories
  - Or separate instruction/data caches