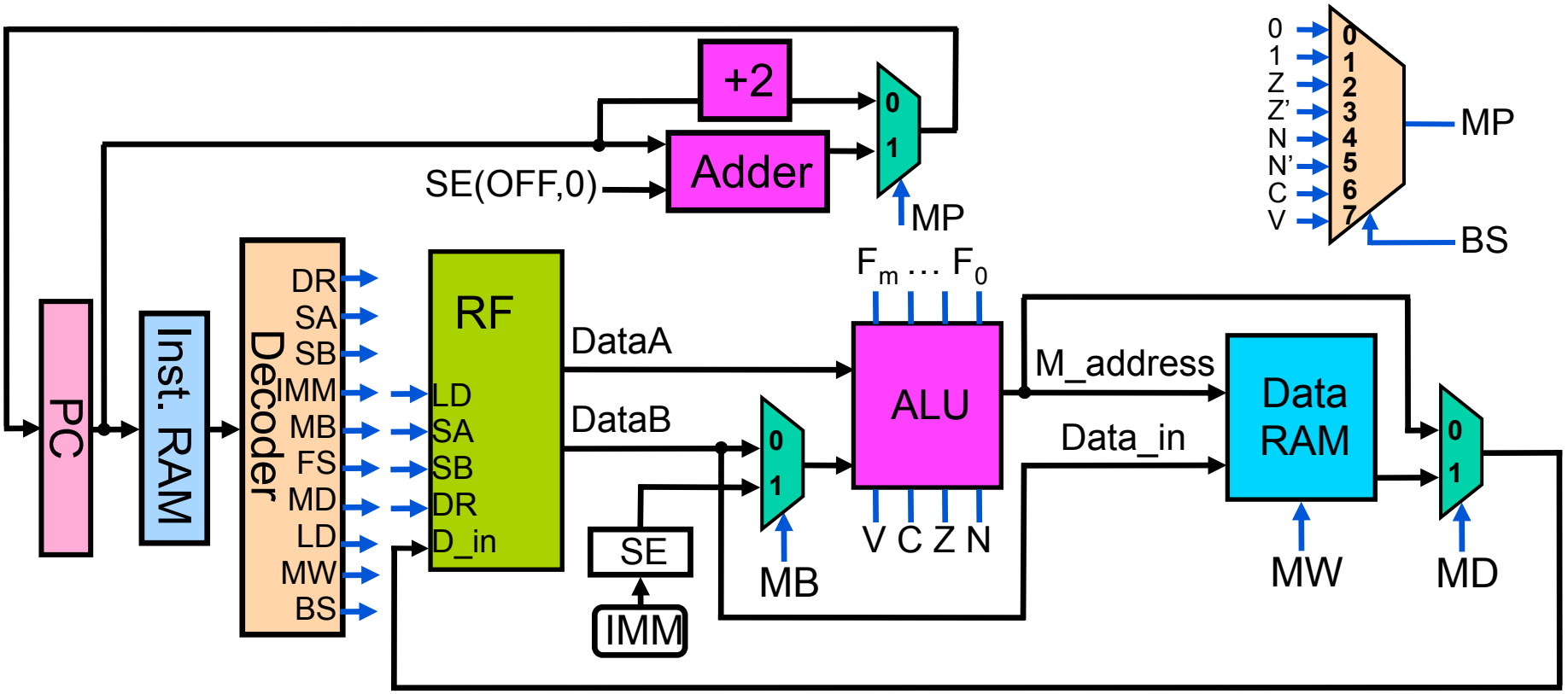
ECE 2300
Digital Logic & Computer Organization

Fall 2016

Pipelined Microprocessor

Cornell University

Programmable Processor



Single Cycle Microprocessor Performance

- Hardware steps: instruction fetch, register read, ALU, memory, register write
- Assume each step takes 1ns
- Different instructions use different steps

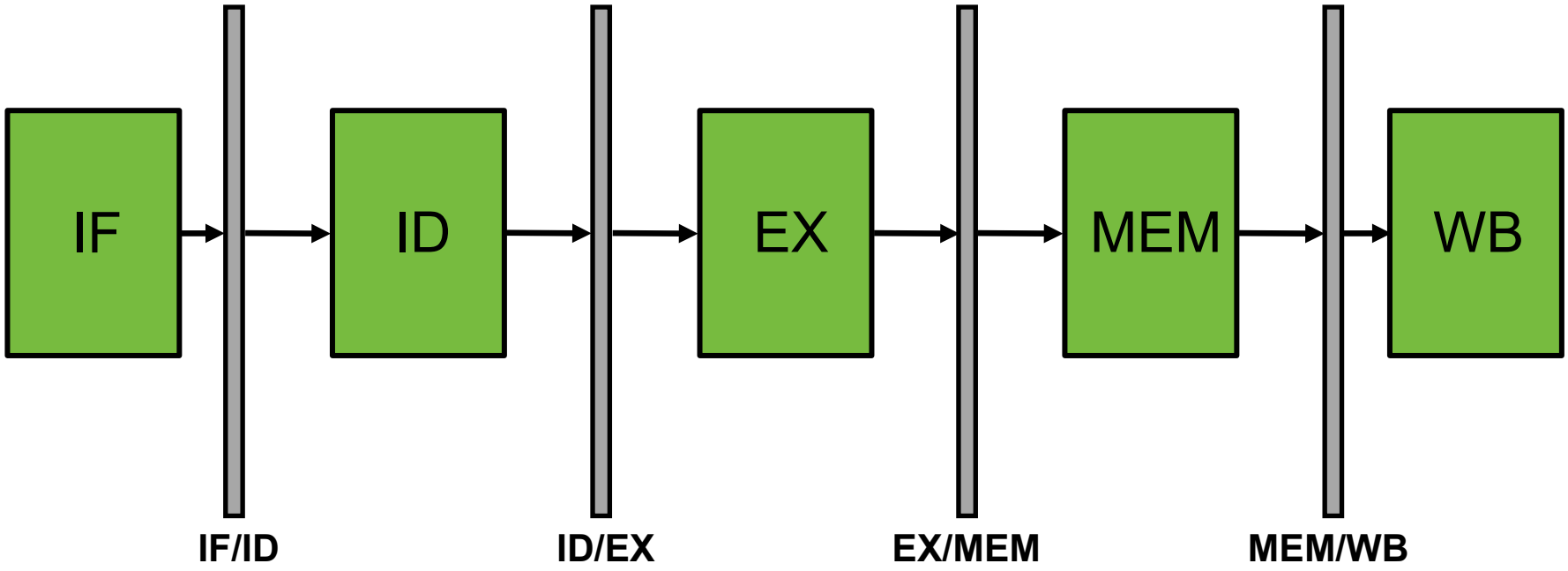
Instruction Type	IF	RR	ALU	MEM	RW	Total
Register to Register	1ns	1ns	1ns	-	1ns	4ns
Load from Memory	1ns	1ns	1ns	1ns	1ns	5ns
Store to Memory	1ns	1ns	1ns	1ns	-	4ns
Jump	1ns	1ns	-	-	-	2ns

- Clock period is limited by longest instruction (5ns)
- Pipelining: Overlap instruction execution by performing each step in successive clock cycles

5 Steps in Instruction Execution

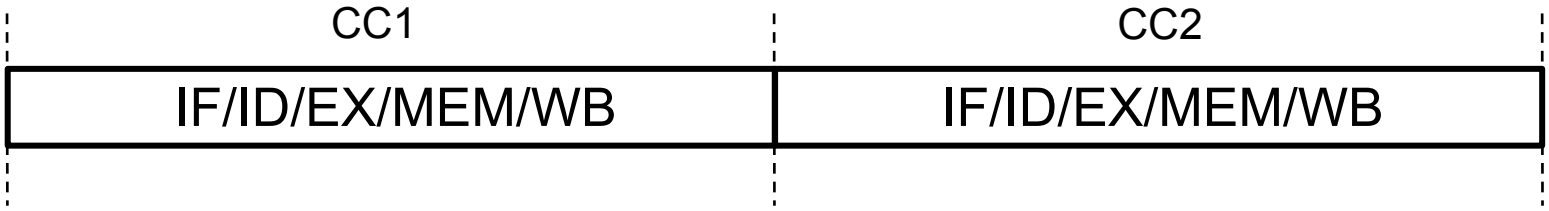
- Instruction Fetch (IF)
  - Update PC; Fetch instruction
- Instruction Decode (ID)
  - Decode instruction; Read register file
- Execute (EX)
  - Perform ALU operation
- Memory (MEM)
  - Perform memory operation
- Write Back (WB)
  - Store result into register file

Pipelining: Basic Idea

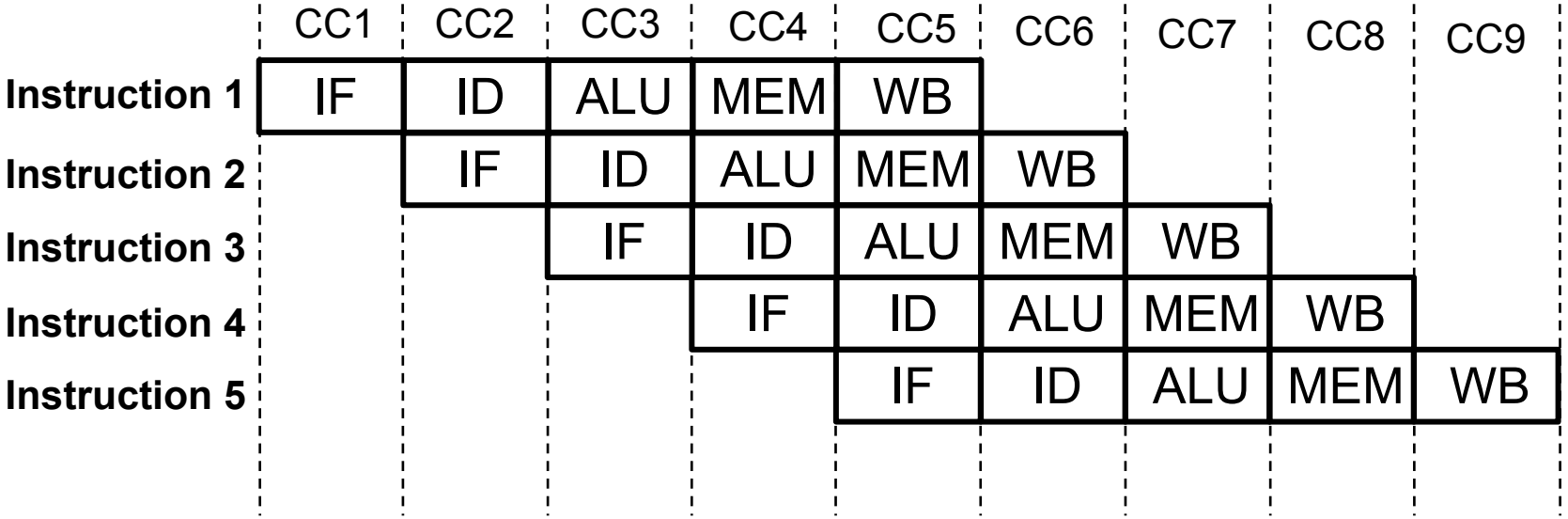


Pipelining: Overlapped Instructions

- **Single-cycle execution**



- **Pipelined execution**

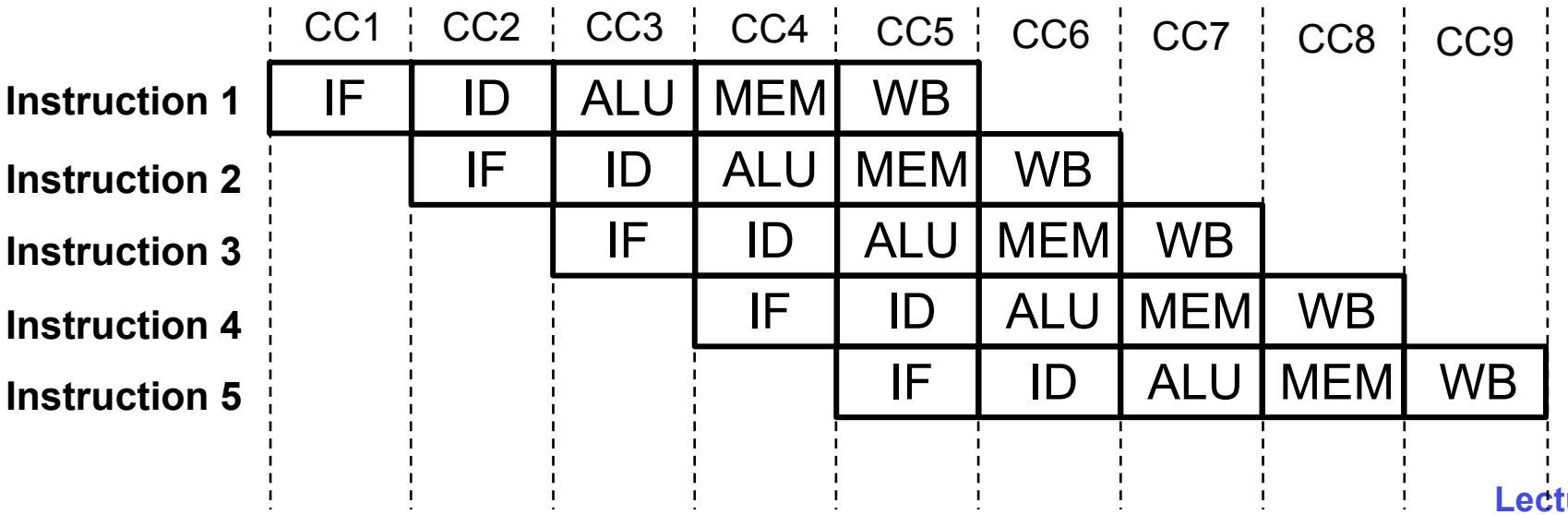


Pipelining: Performance

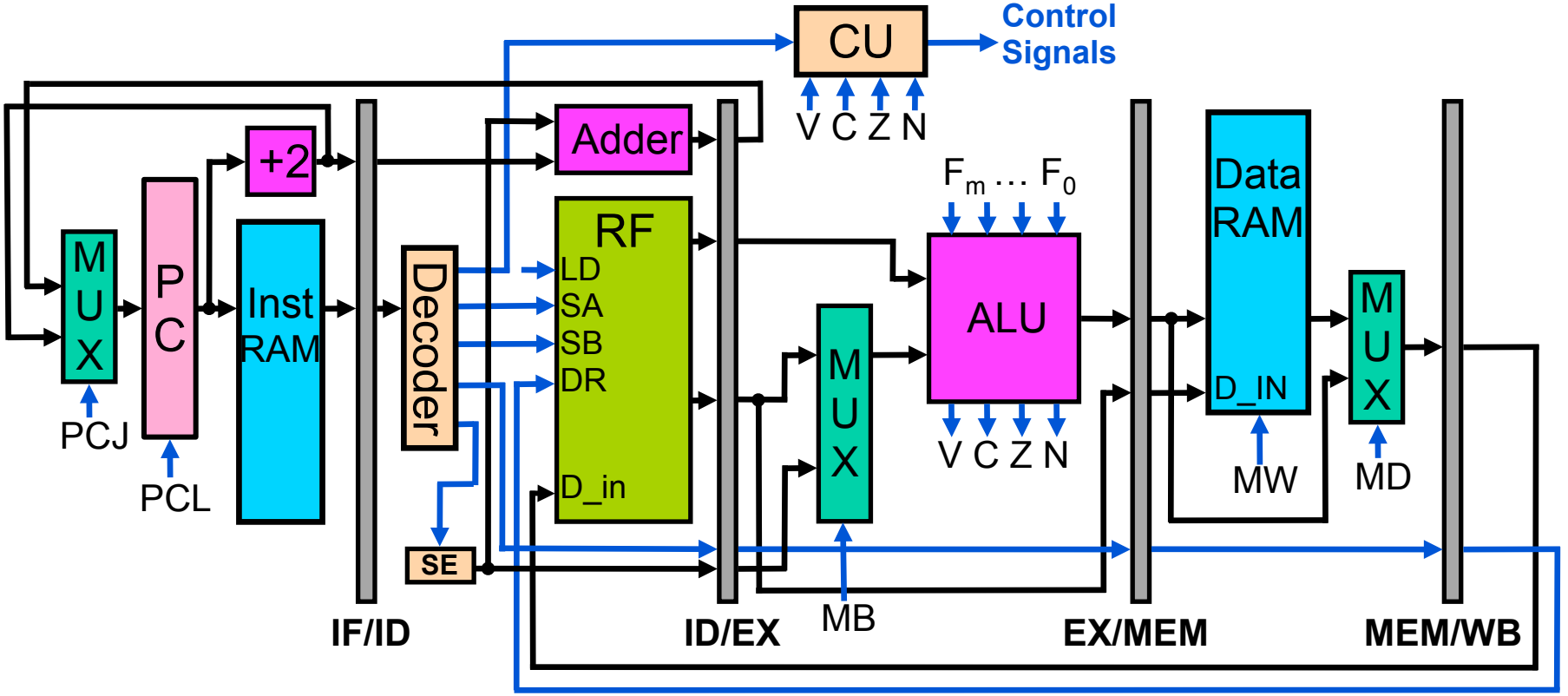
- Faster clock frequency than single cycle processor
- Each instruction takes 5 cycles
- Average number of cycles per instruction (CPI)

$$\frac{\text{Number of cycles for } N \text{ instructions}}{N} = \frac{N+4}{N} \approx 1$$

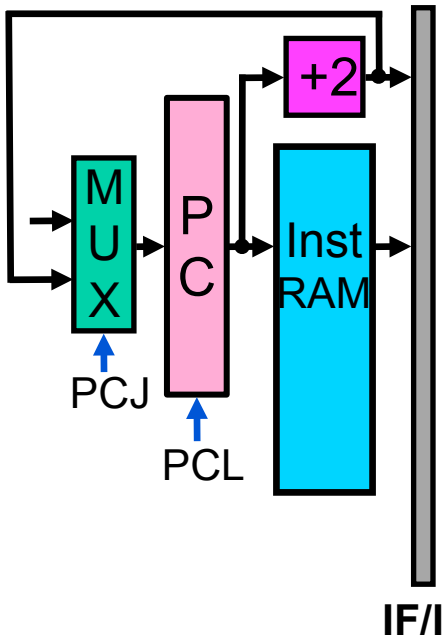
- ~1 instruction completed every cycle (ideally)



Pipelined Processor

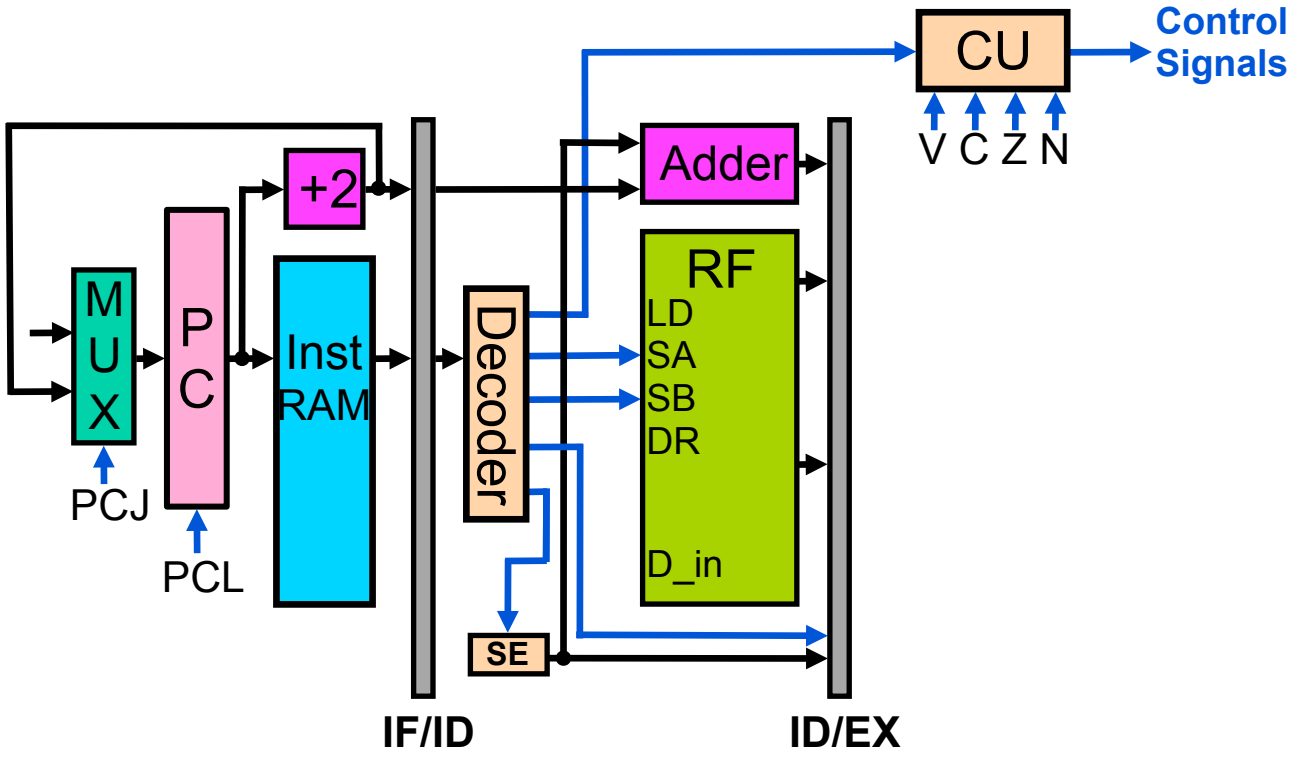


Instruction Fetch Stage



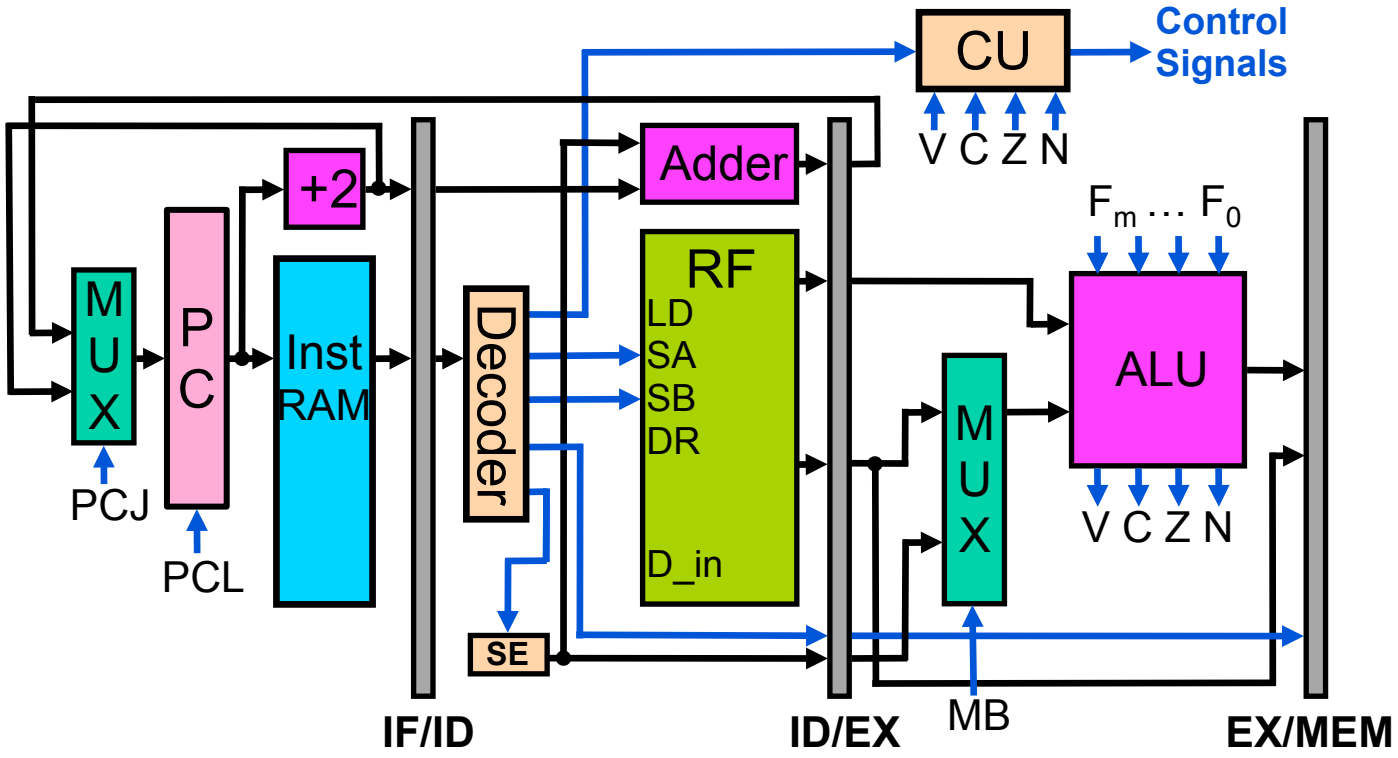
- Fetch the instruction into IF/ID
- Load PC+2 into PC
- Place PC+2 into IF/ID

Instruction Decode Stage



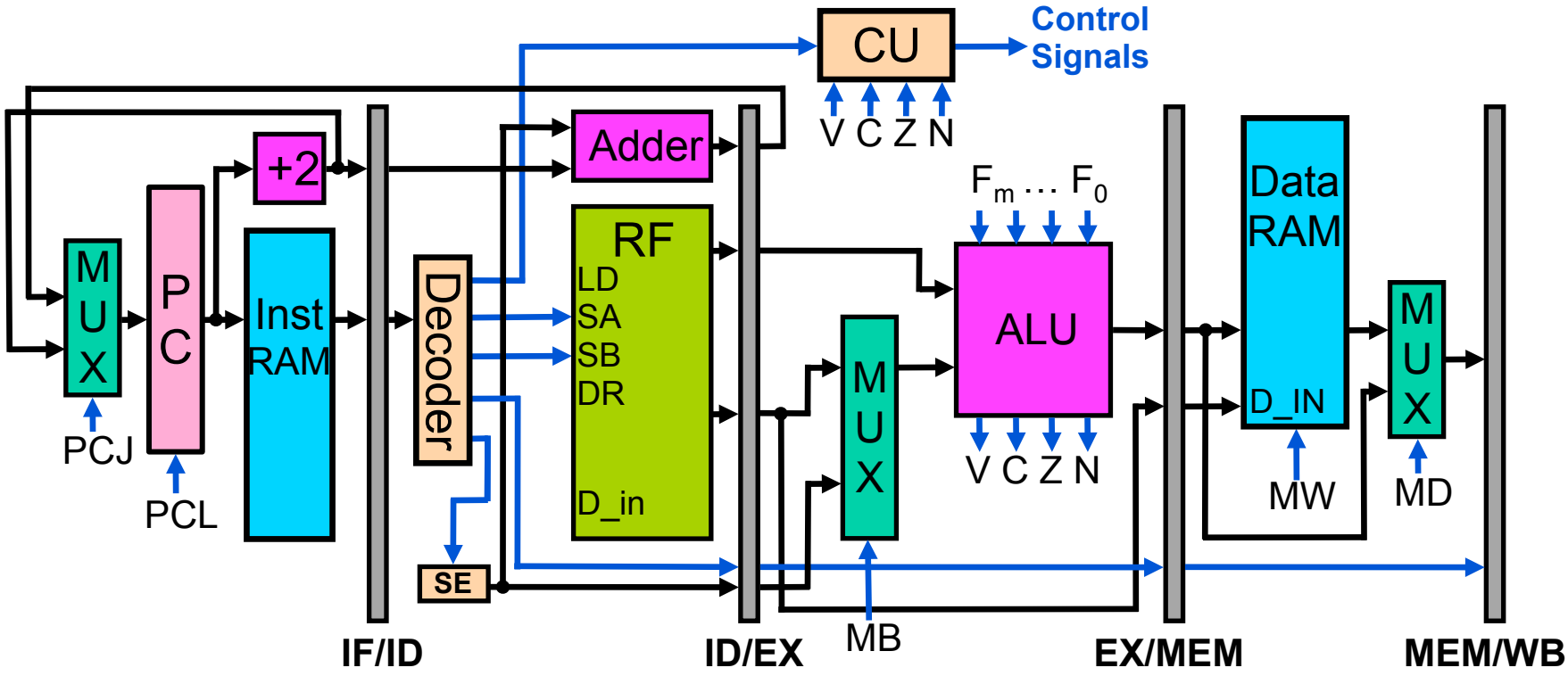
- Read source operands from RF into ID/EX
- Place SE(IMM) into ID/EX
- Place SE(IMM)+(PC+2) into ID/EX
- Place DR into ID/EX

Execute Stage



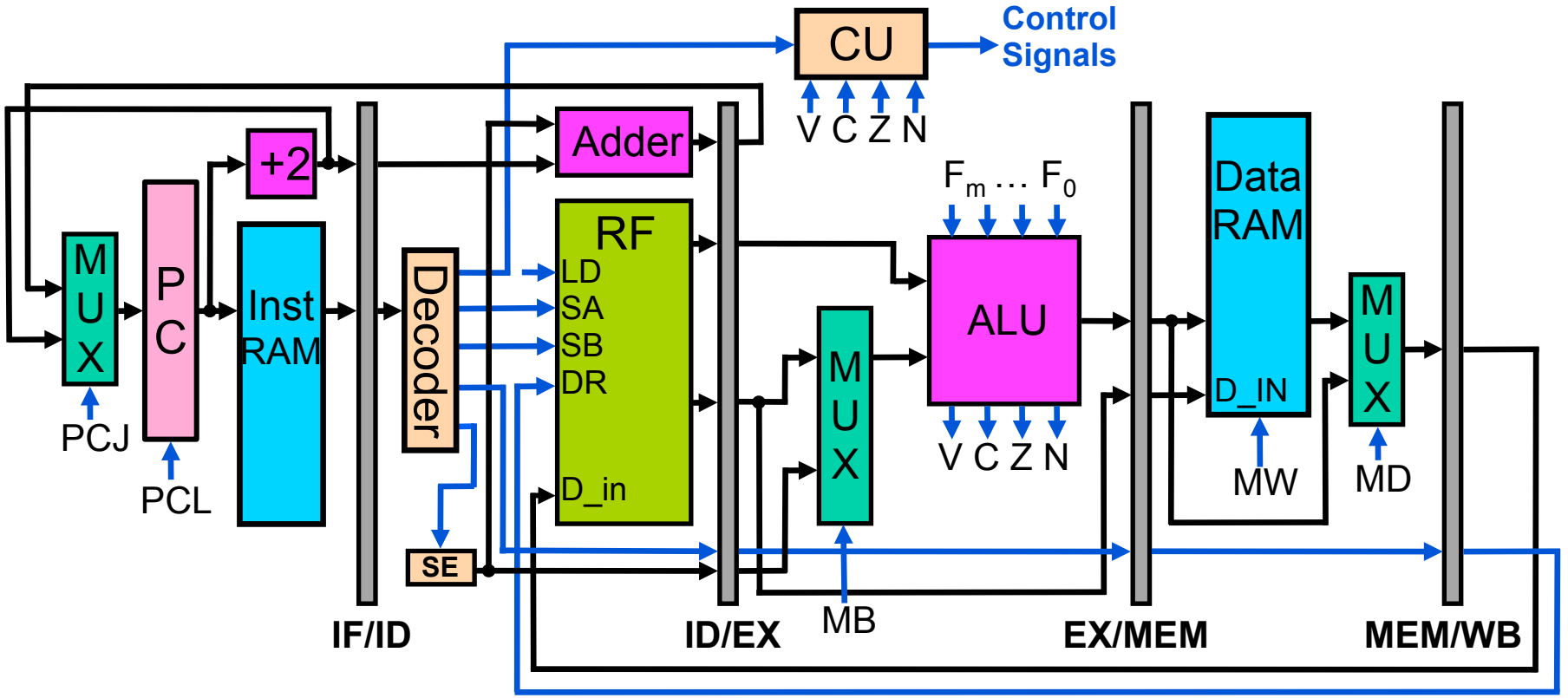
- Perform ALU operation and place into EX/MEM
- Pass DataB from the RF to EX/MEM
- Pass DR to EX/MEM
- If taken branch, update PC

Memory Stage



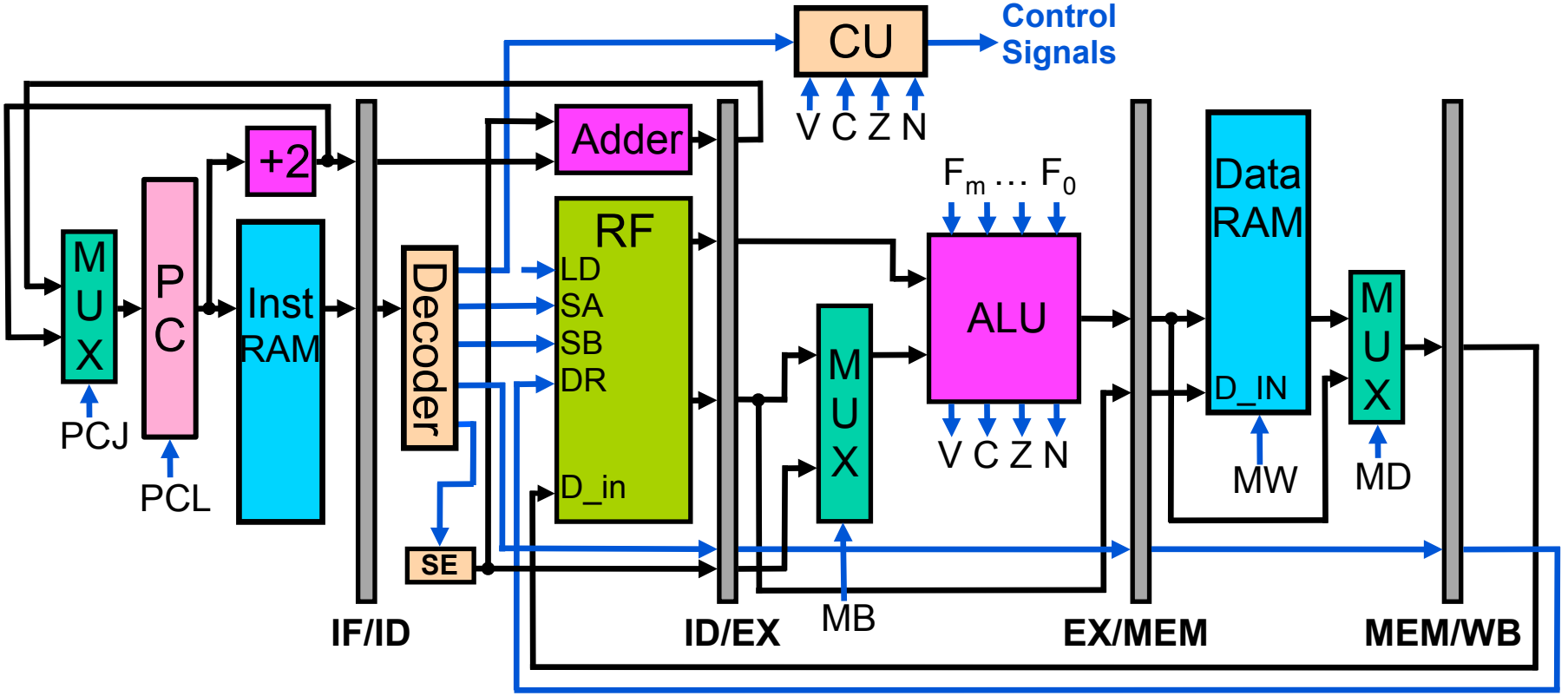
- **Store**: Write data into RAM
- **Load**: Read data from RAM into MEM/WB
- **ALU operation**: pass ALU result from EX/MEM to MEM/WB
- Pass DR to MEM/WB

Writeback Stage

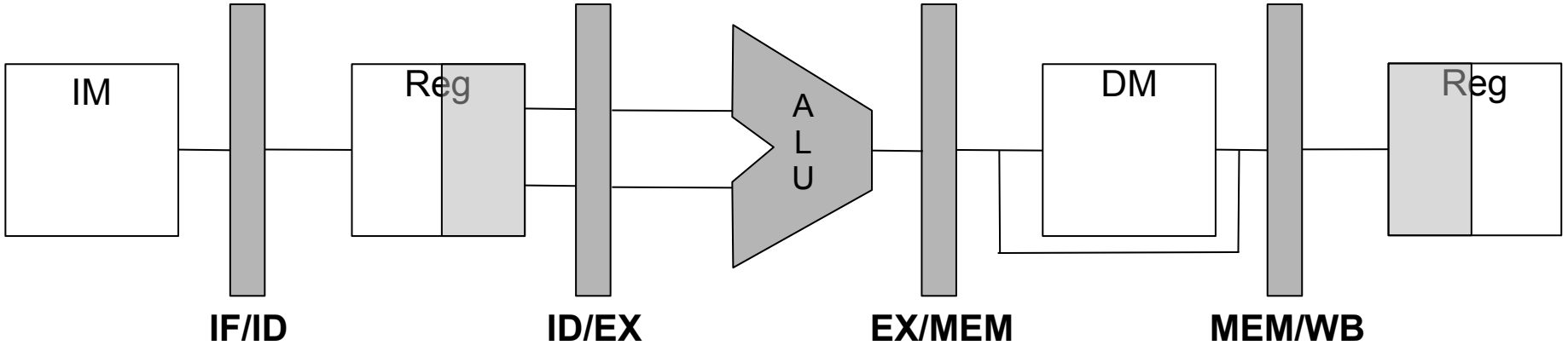


- Load or ALU operation: Write register file

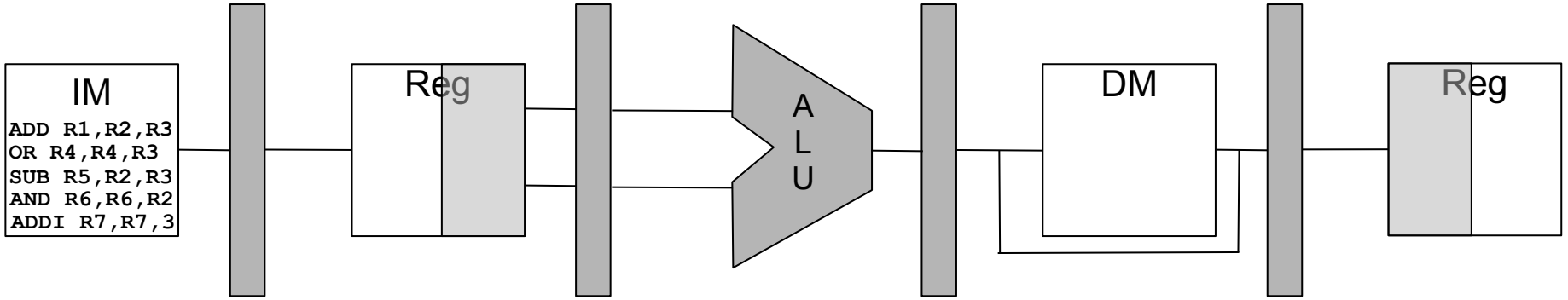
Pipelined Microprocessor



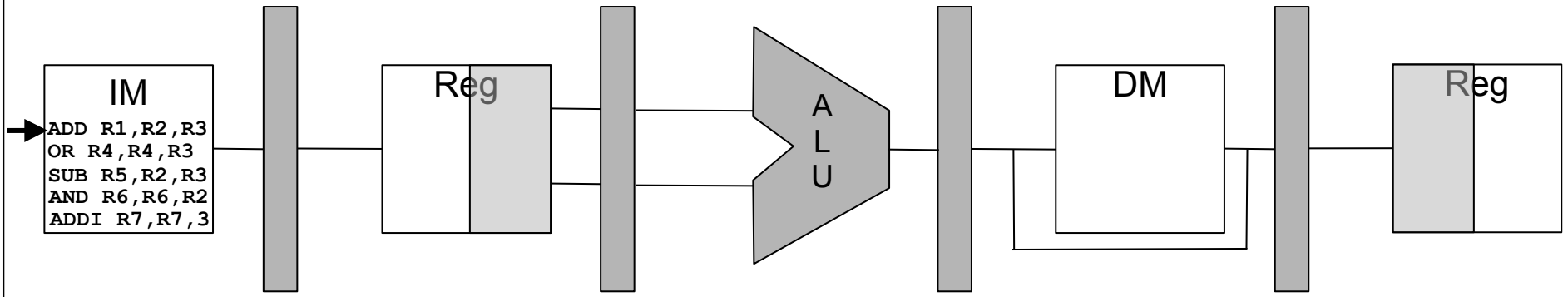
Abstract Representation



Example Instruction Sequence

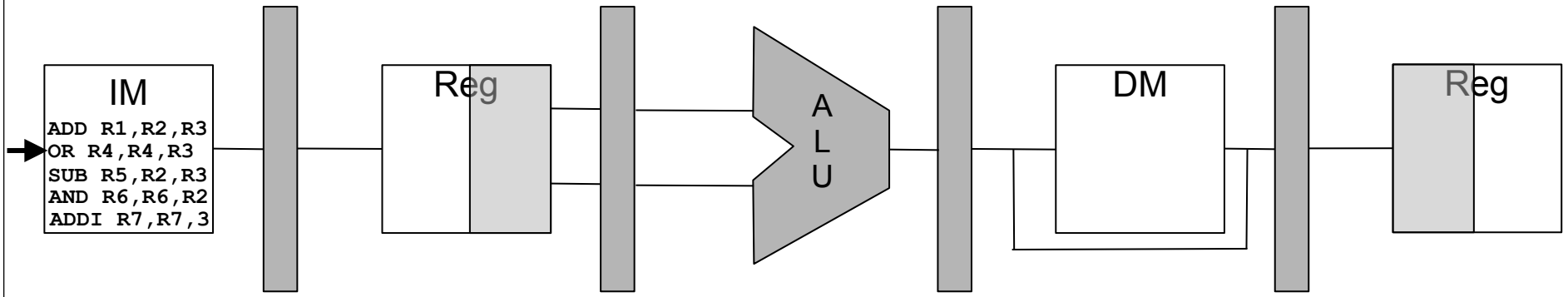


Example Instruction Sequence



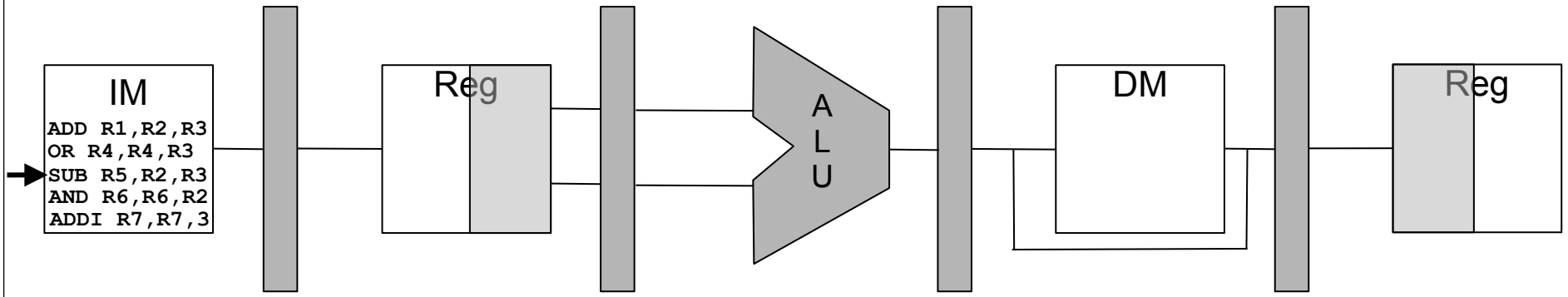
ADD R1,R2,R3

Example Instruction Sequence



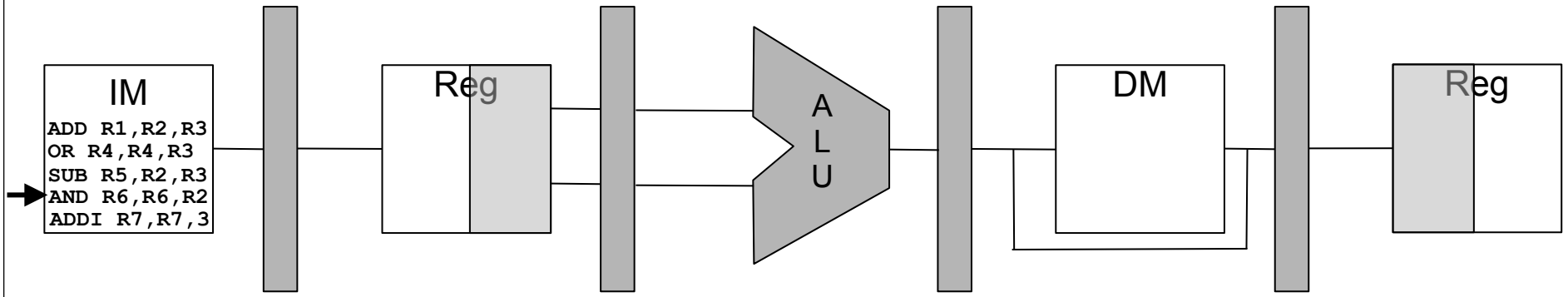
OR R4,R4,R3 **ADD R1,R2,R3**

Example Instruction Sequence



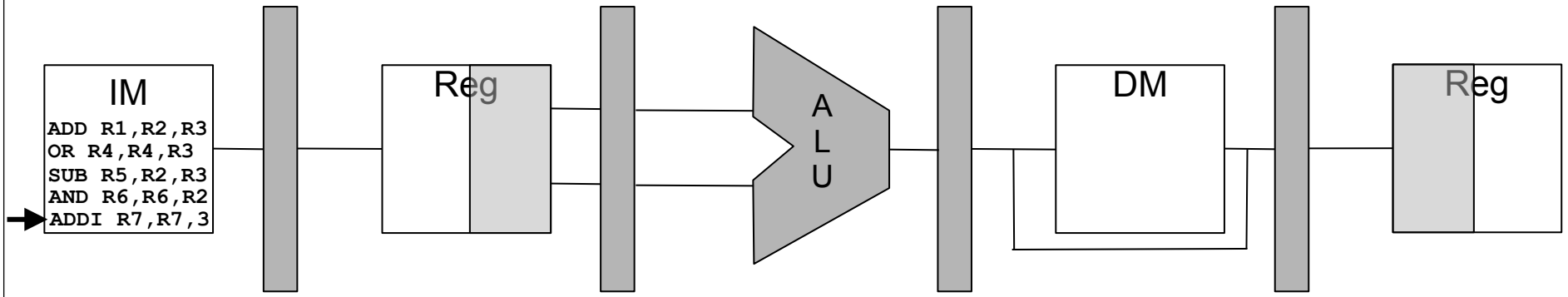
SUB R5,R2,R3 **OR R4,R4,R3** **ADD R1,R2,R3**

Example Instruction Sequence



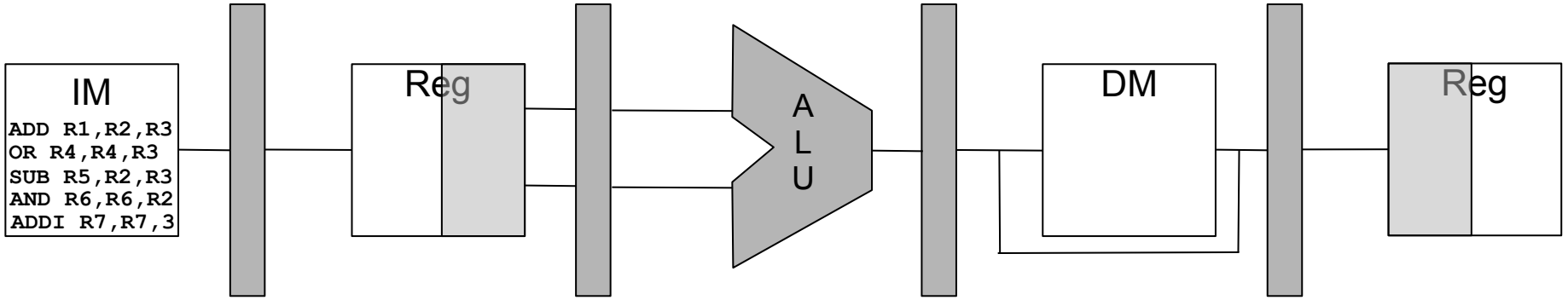
AND R6,R6,R2 SUB R5,R2,R3 OR R4,R4,R3 ADD R1,R2,R3

Example Instruction Sequence



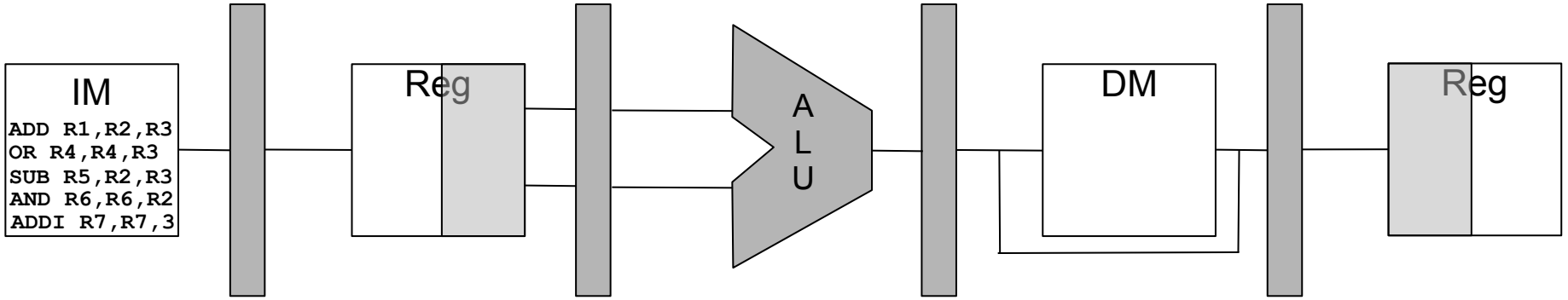
ADDI R7,R7,3 AND R6,R6,R2 SUB R5,R2,R3 OR R4,R4,R3 ADD R1,R2,R3

Example Instruction Sequence



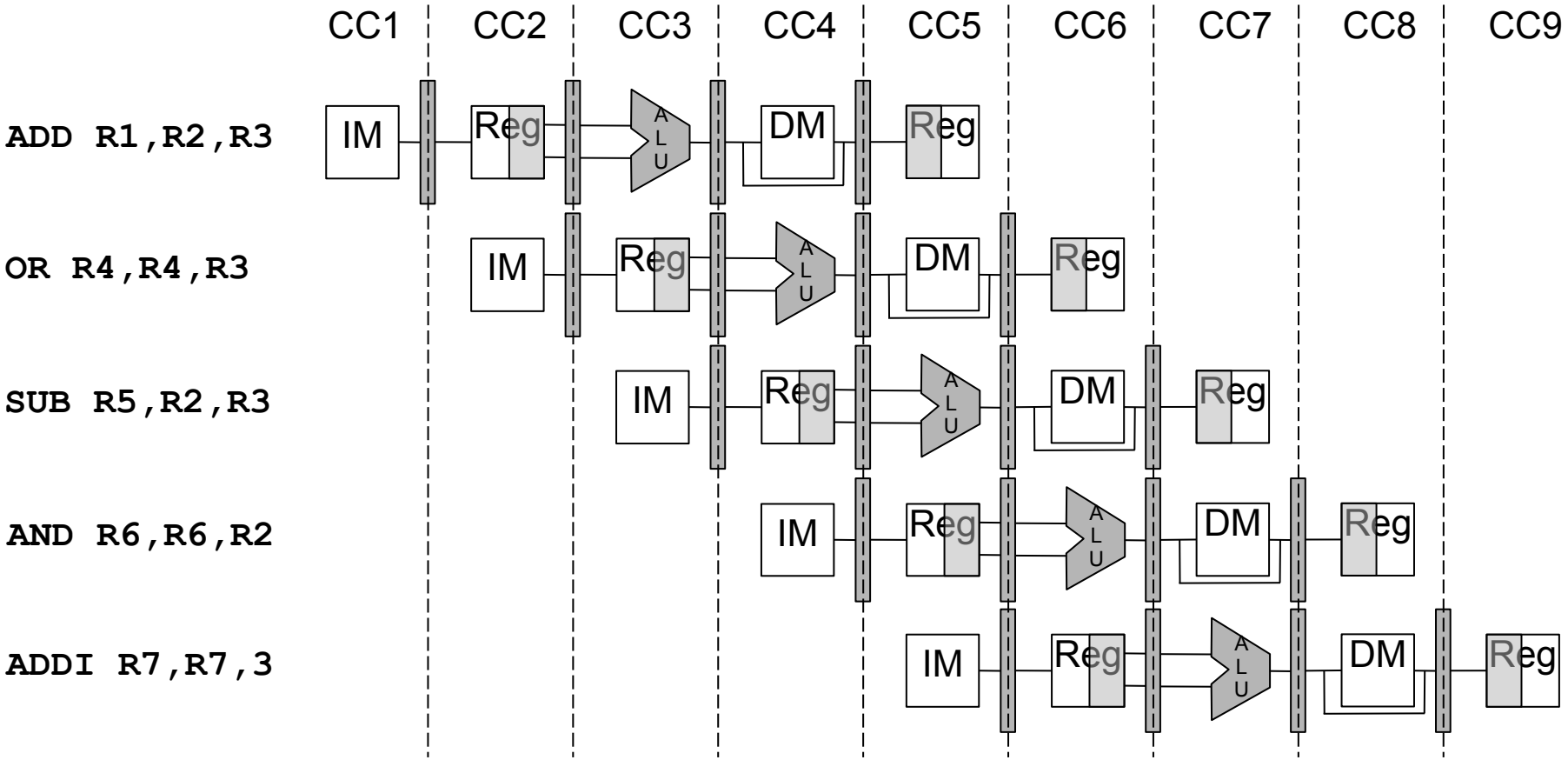
ADDI R7,R7,3 **AND R6,R6,R2** **SUB R5,R2,R3** **OR R4,R4,R3**

Example Instruction Sequence

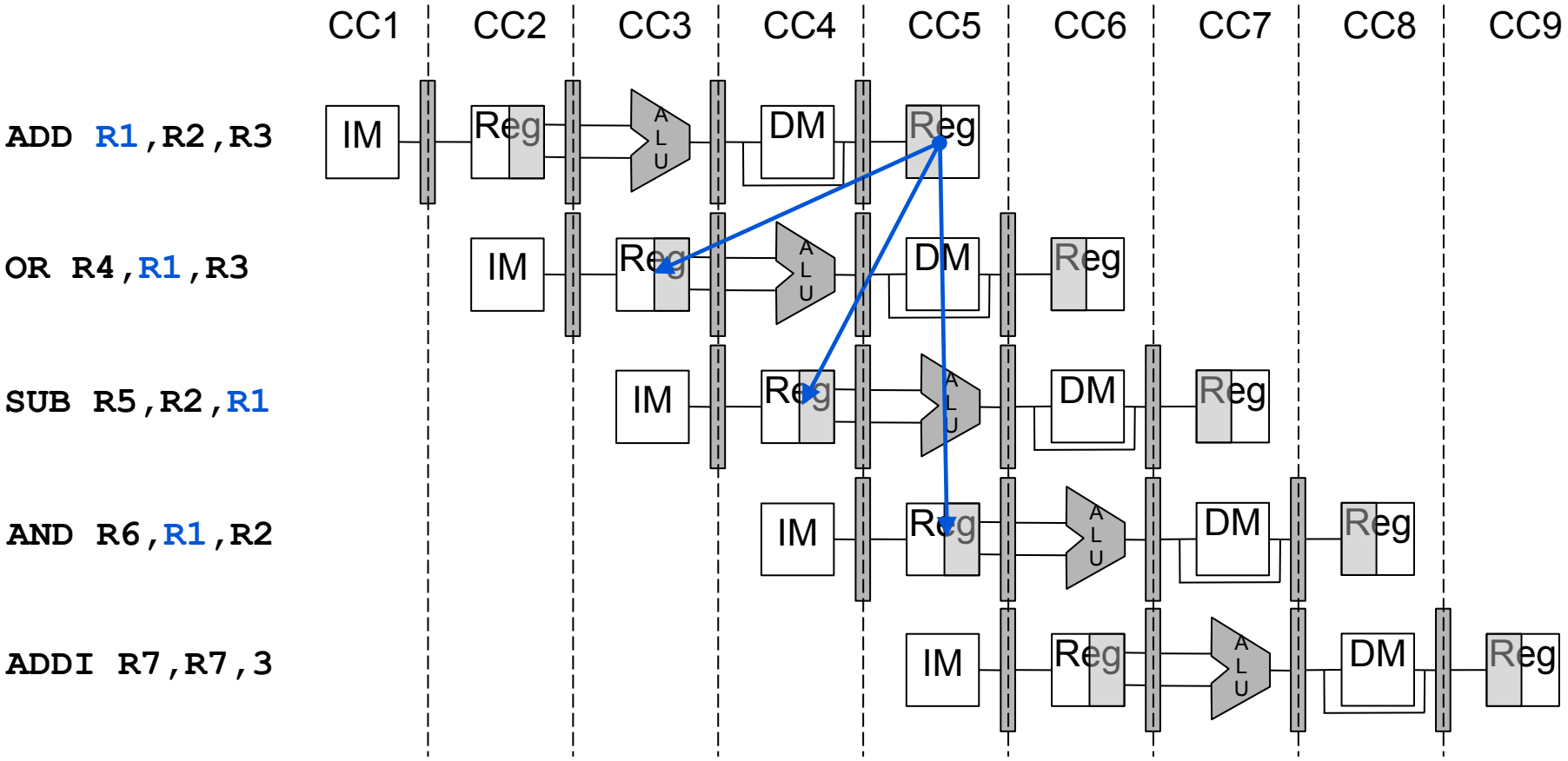


ADDI R7,R7,3 **AND R6,R6,R2** **SUB R5,R2,R3**

Example Instruction Sequence



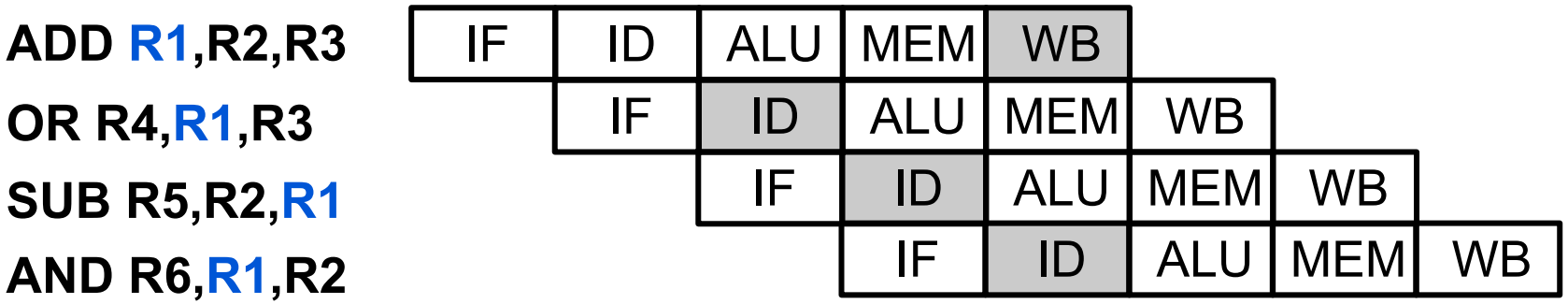
What About This Sequence?



The OR, SUB, and AND instructions are *data dependent* on the ADD instruction

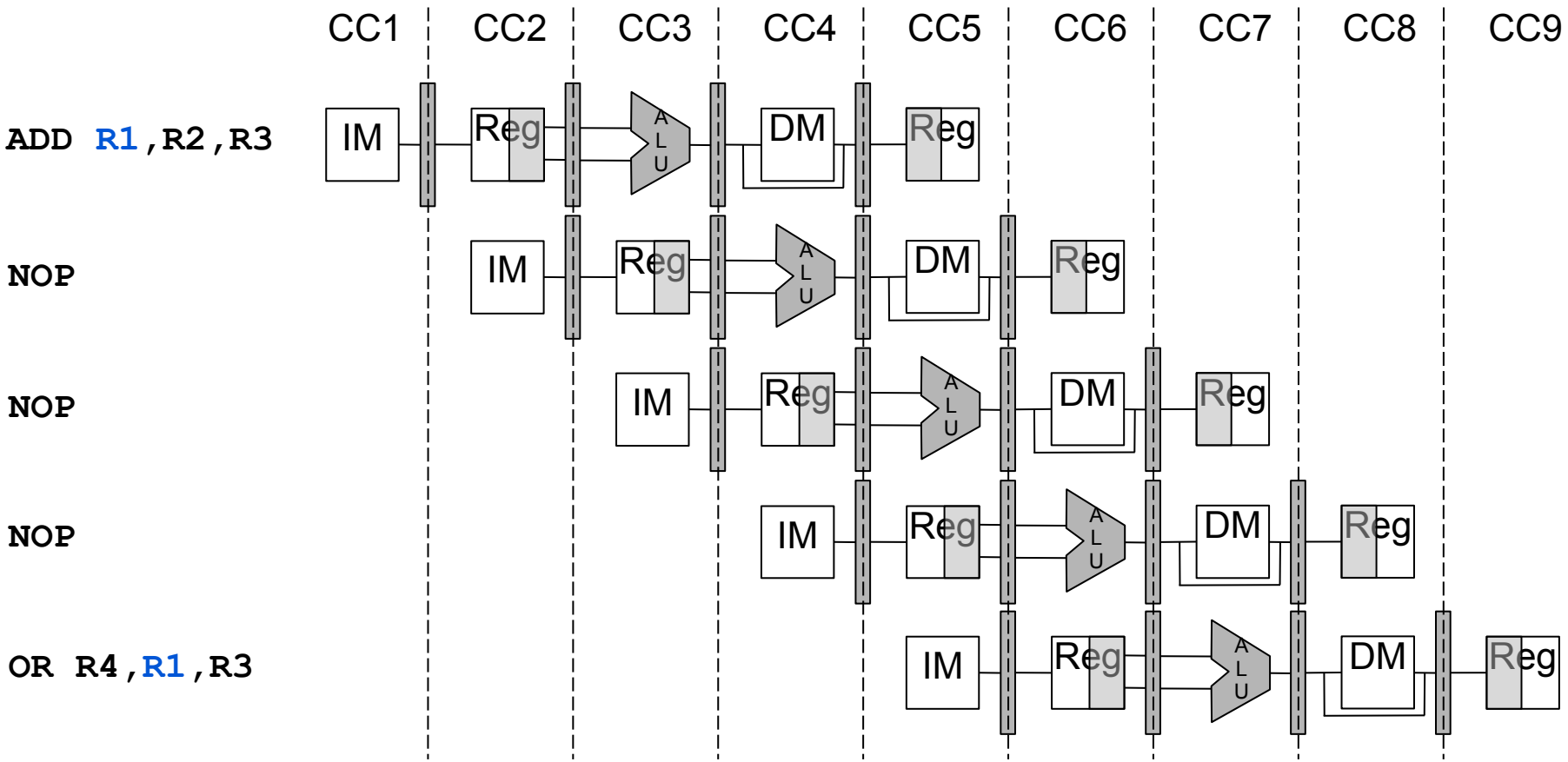
Data Hazard

- Occurs when a register is read before the write back of a value to that register

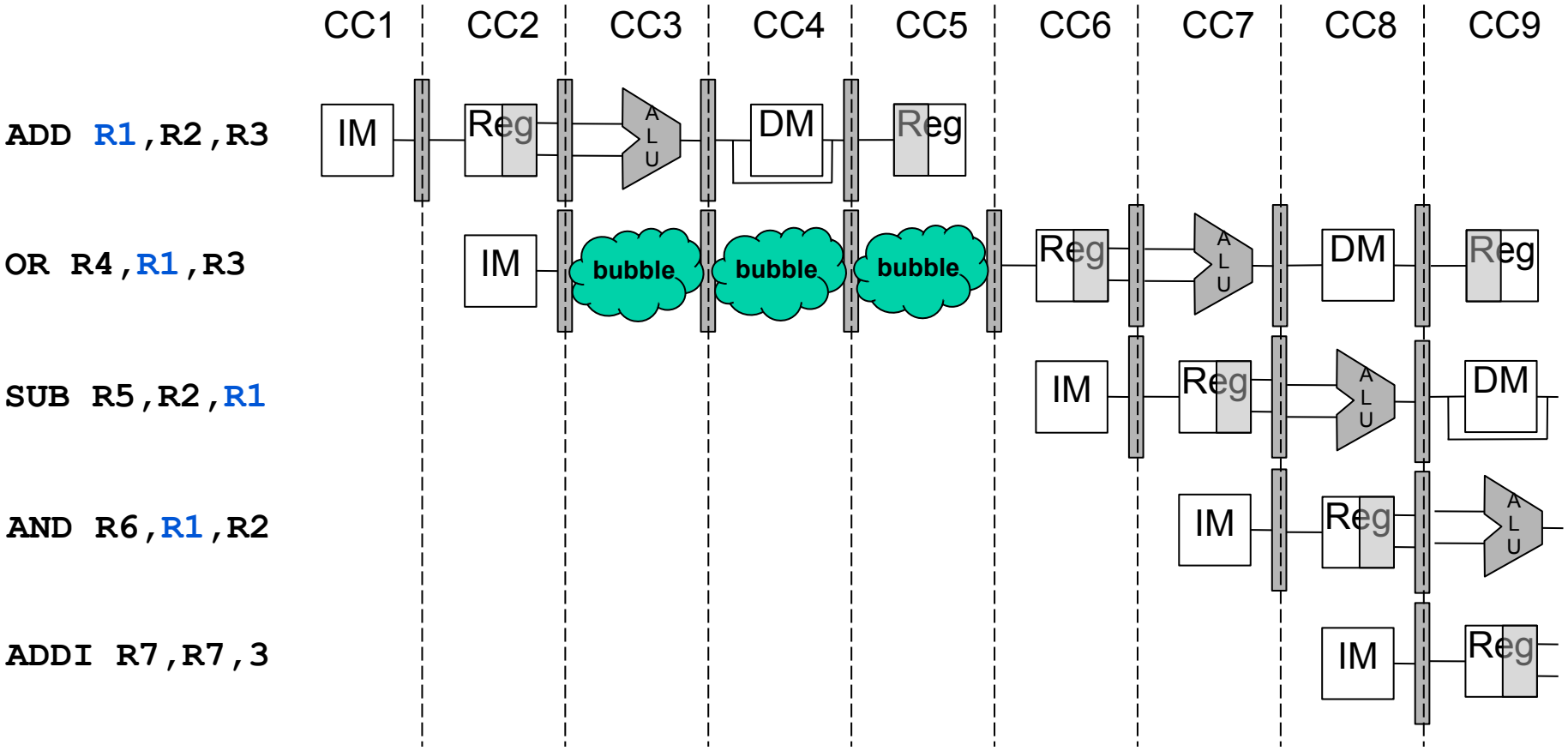


- What should happen
  - The 1st instruction calculates a new value for R1
  - The 2nd, 3rd, and 4th instructions use this new value
- What actually happens
  - The 2nd, 3rd, and 4th instructions read the old value of R1
  - The first instruction then writes the new value into R1

Solution 1: SW Inserts NOP Instructions

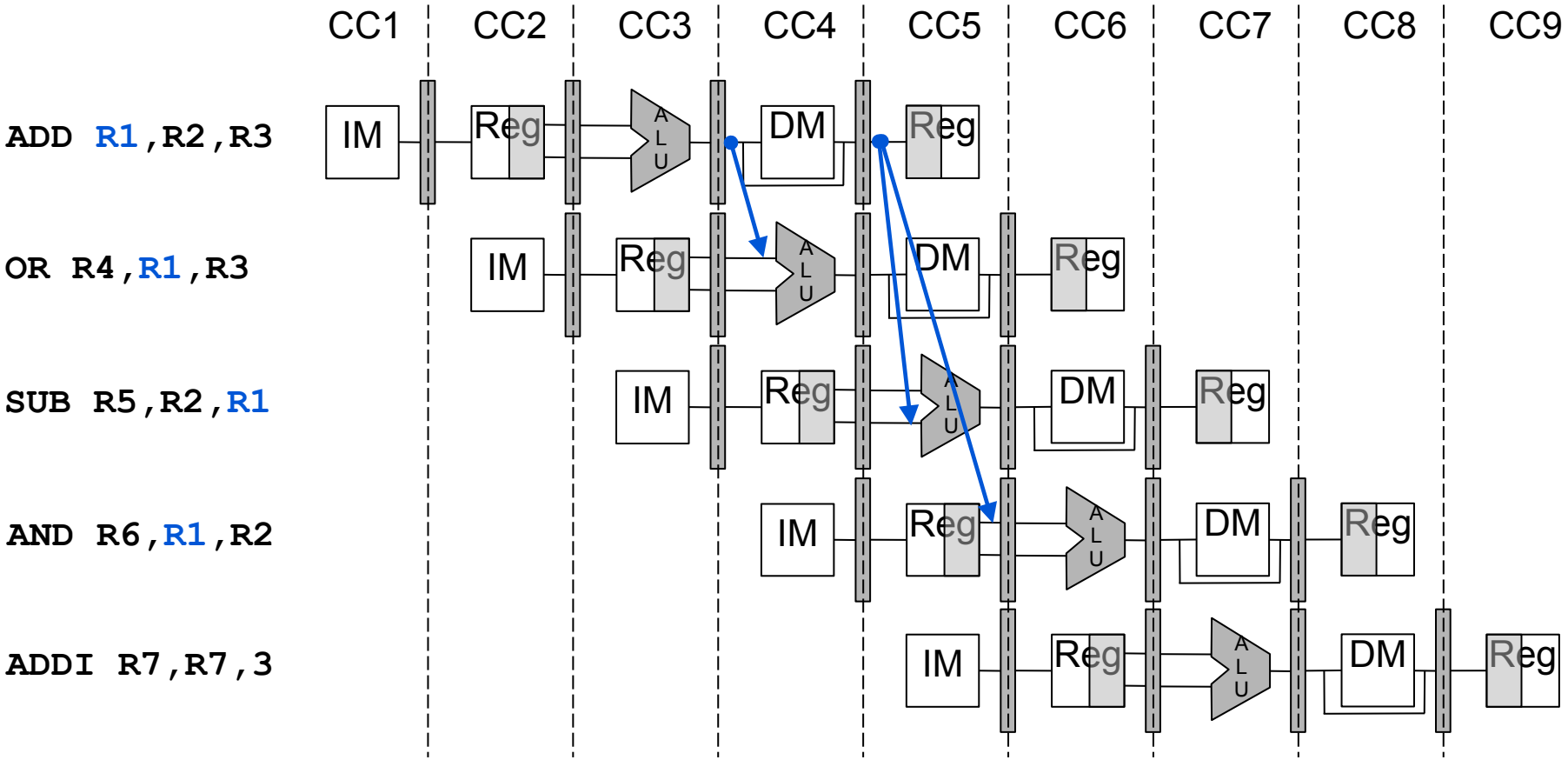


Solution 2: HW *Stalls* the Pipeline

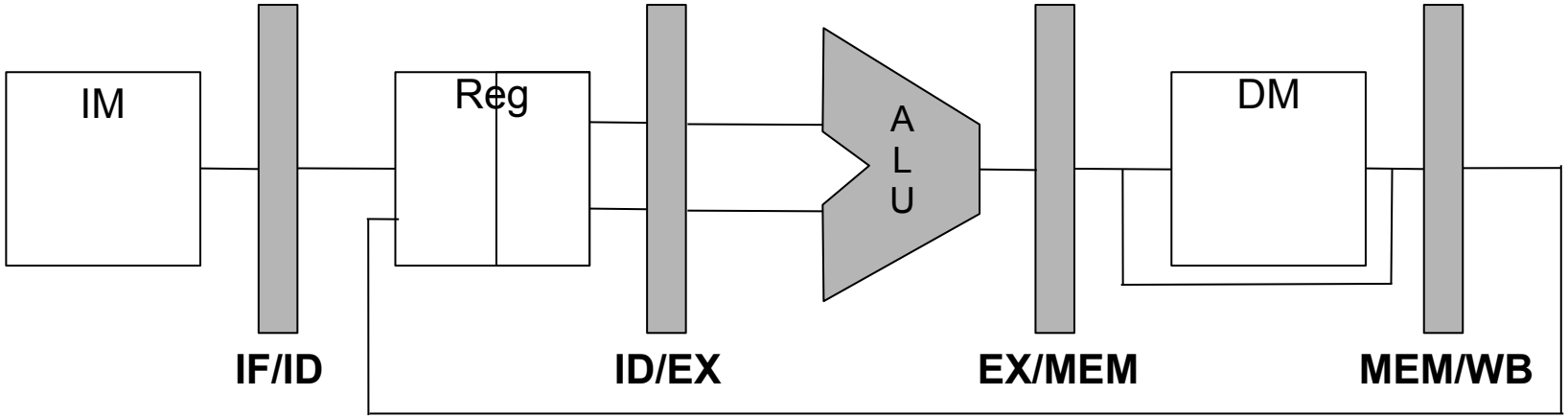


The OR instruction remains in the IF stage for three additional cycles

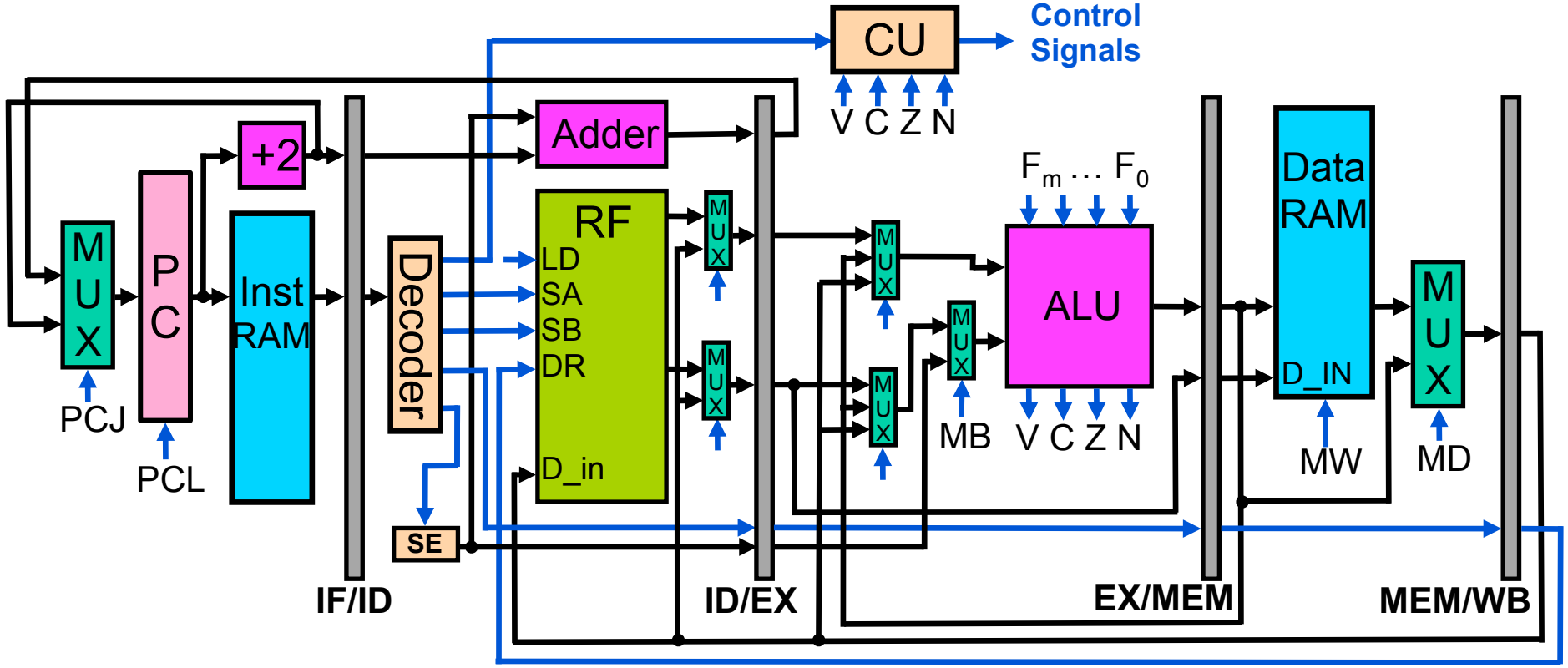
Solution 3: HW Forwarding (Bypassing)



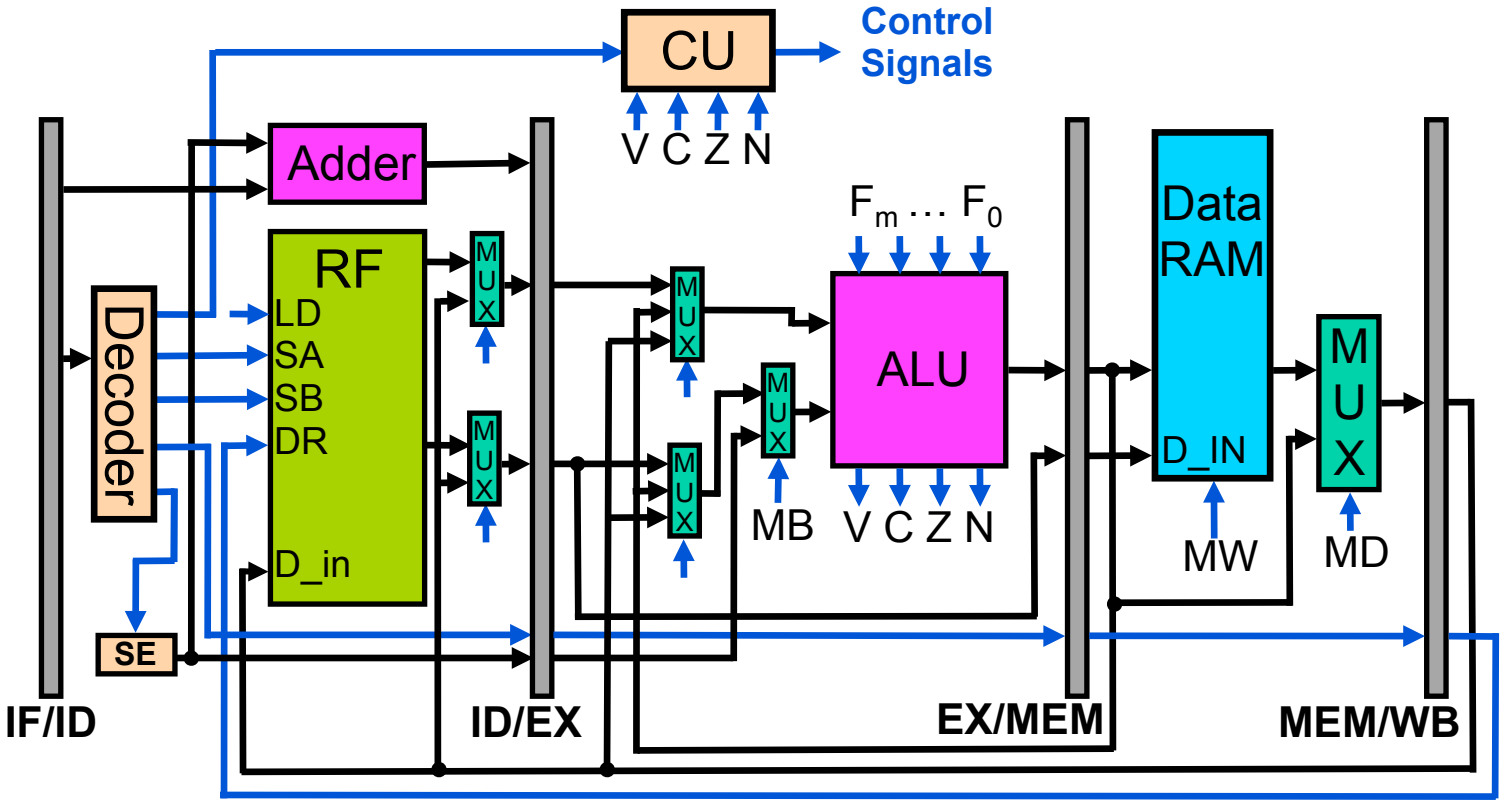
Pipeline Modifications for Forwarding?



Pipelined Processor with Forwarding

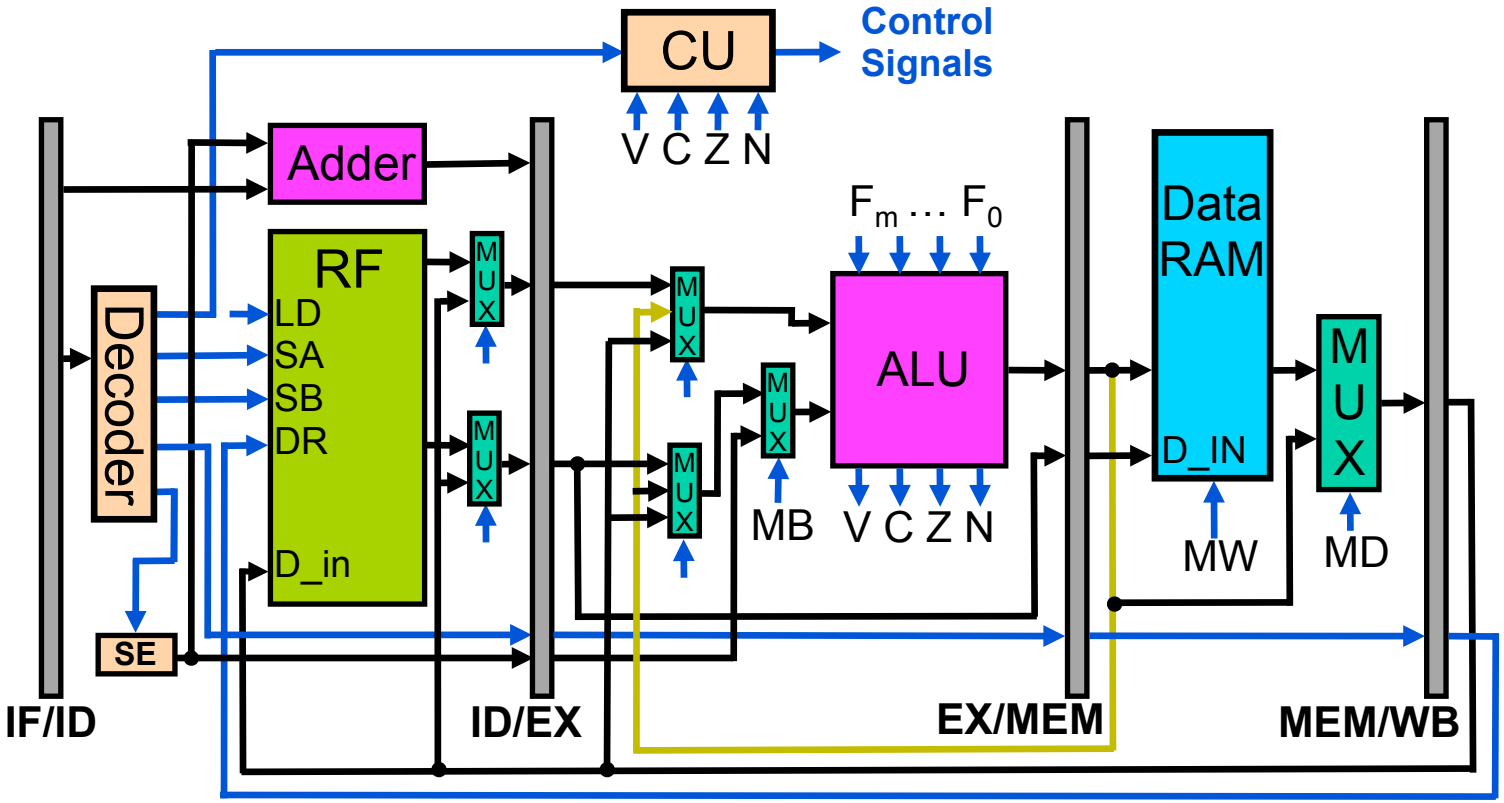


Forwarding in Action



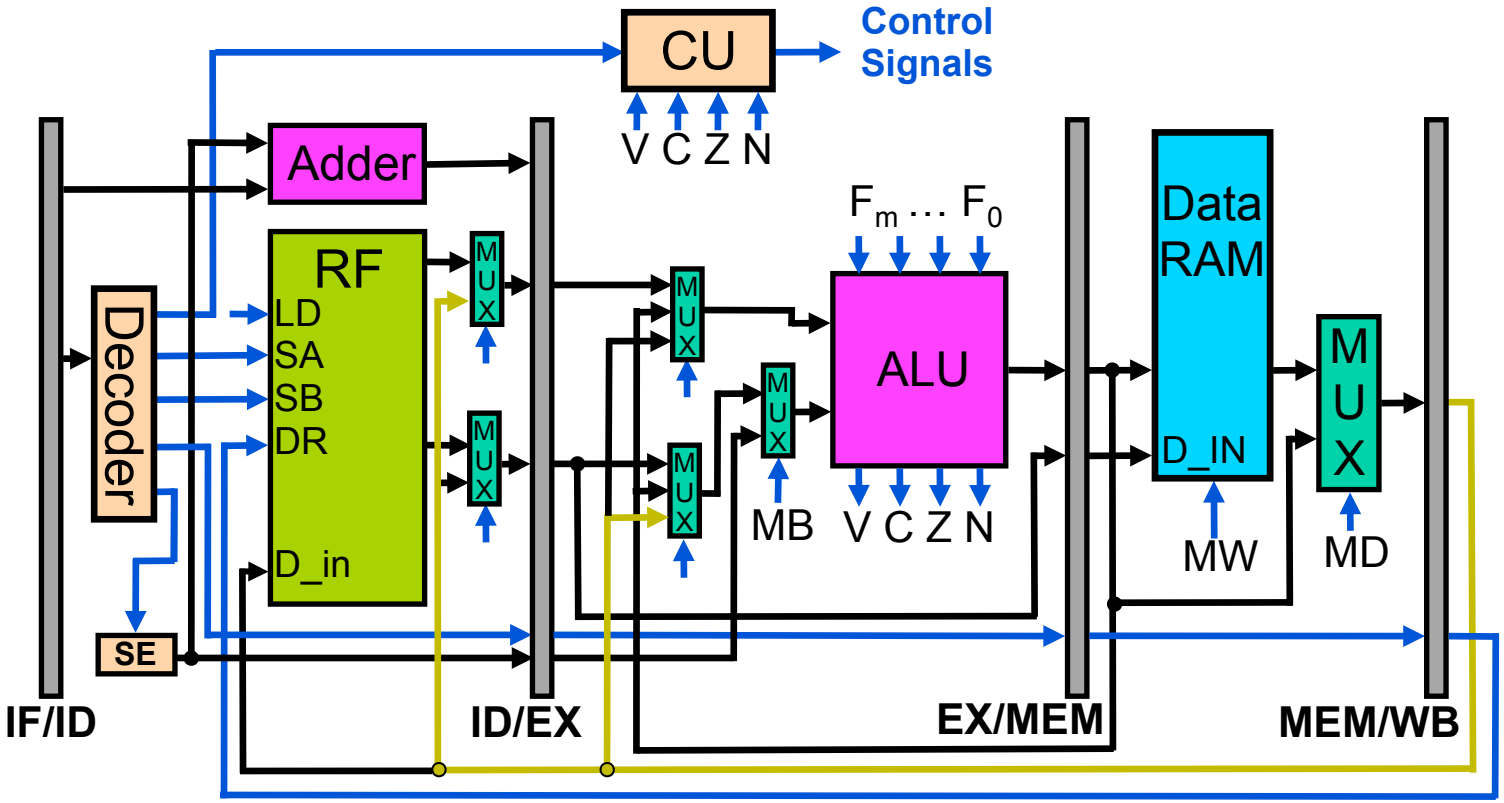

```
OR  R4,R1,R3        ADD  R1,R2,R3
```

Forwarding in Action




```
SUB R5,R2,R1        OR R4,R1,R3        ADD R1,R2,R3
```

Forwarding in Action




```
AND R6,R1,R2        SUB R5,R2,R1        OR R4,R1,R3        ADD R1,R2,R3
```

Next Time

More Pipelined Microprocessor