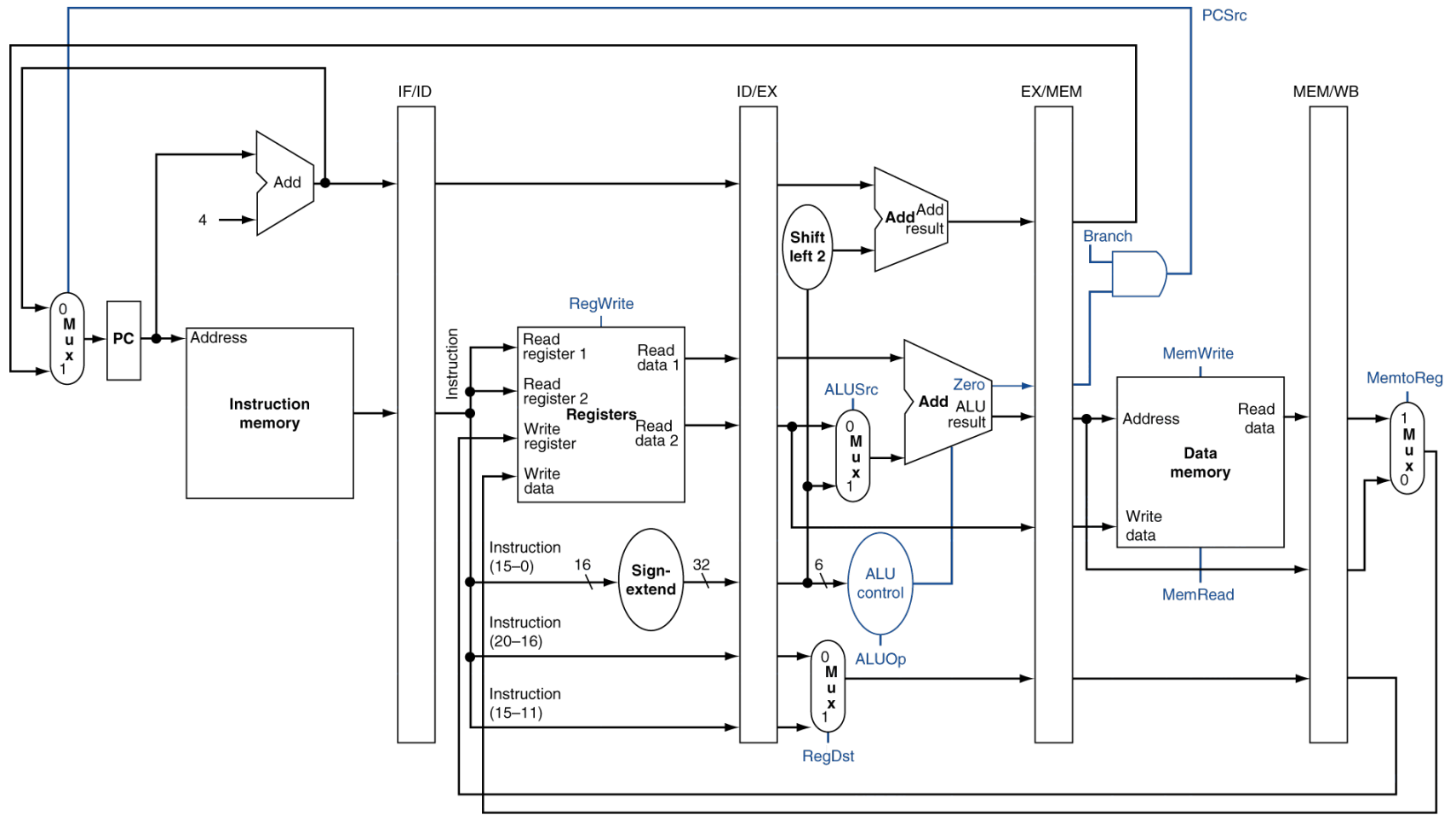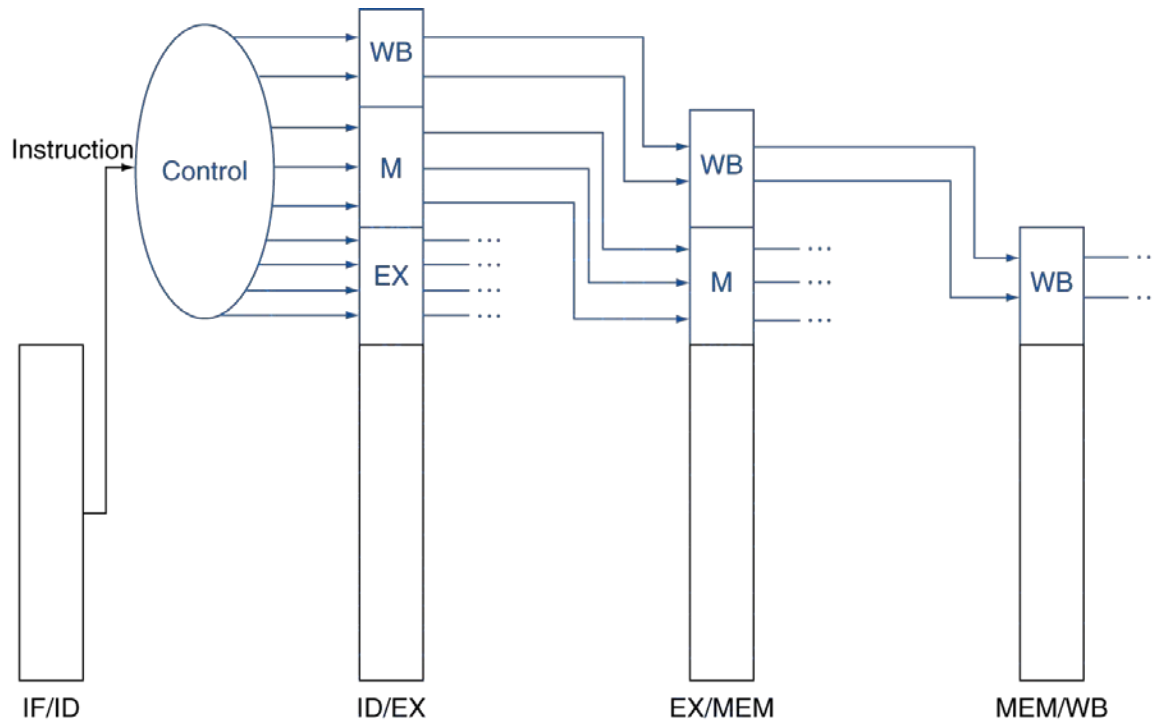# The Processor

Computer Architecture

2019 1학기

한양대학교 공과대학 컴퓨터소프트웨어학부
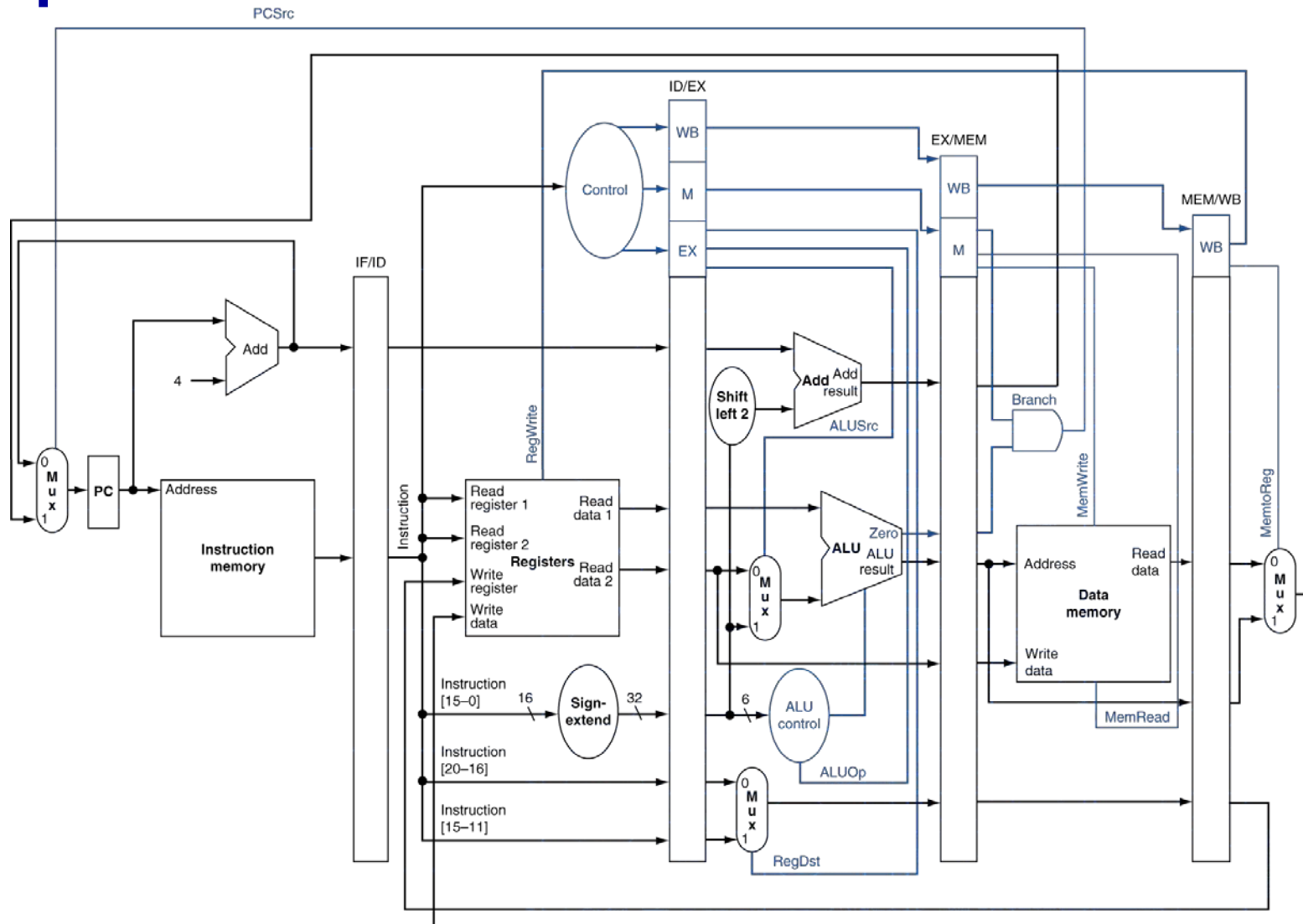홍석준

Pipelined Control (Simplified)

Pipelined Control

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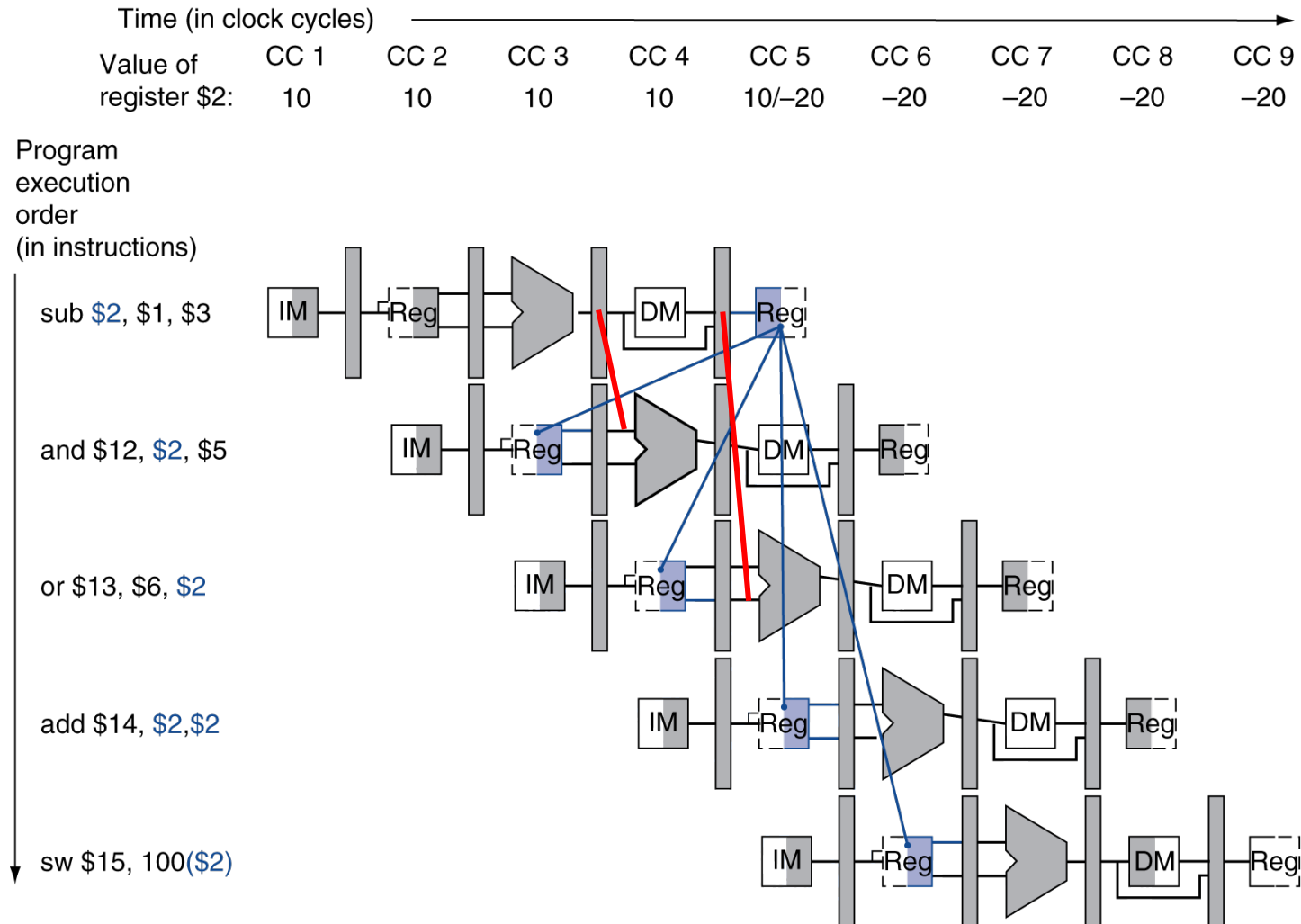 - As in single-cycle implementation

Pipelined Control

Data Hazards in ALU Instructions

- Consider this sequence:

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

- 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.RegisterRs = register number for Rs sitting 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
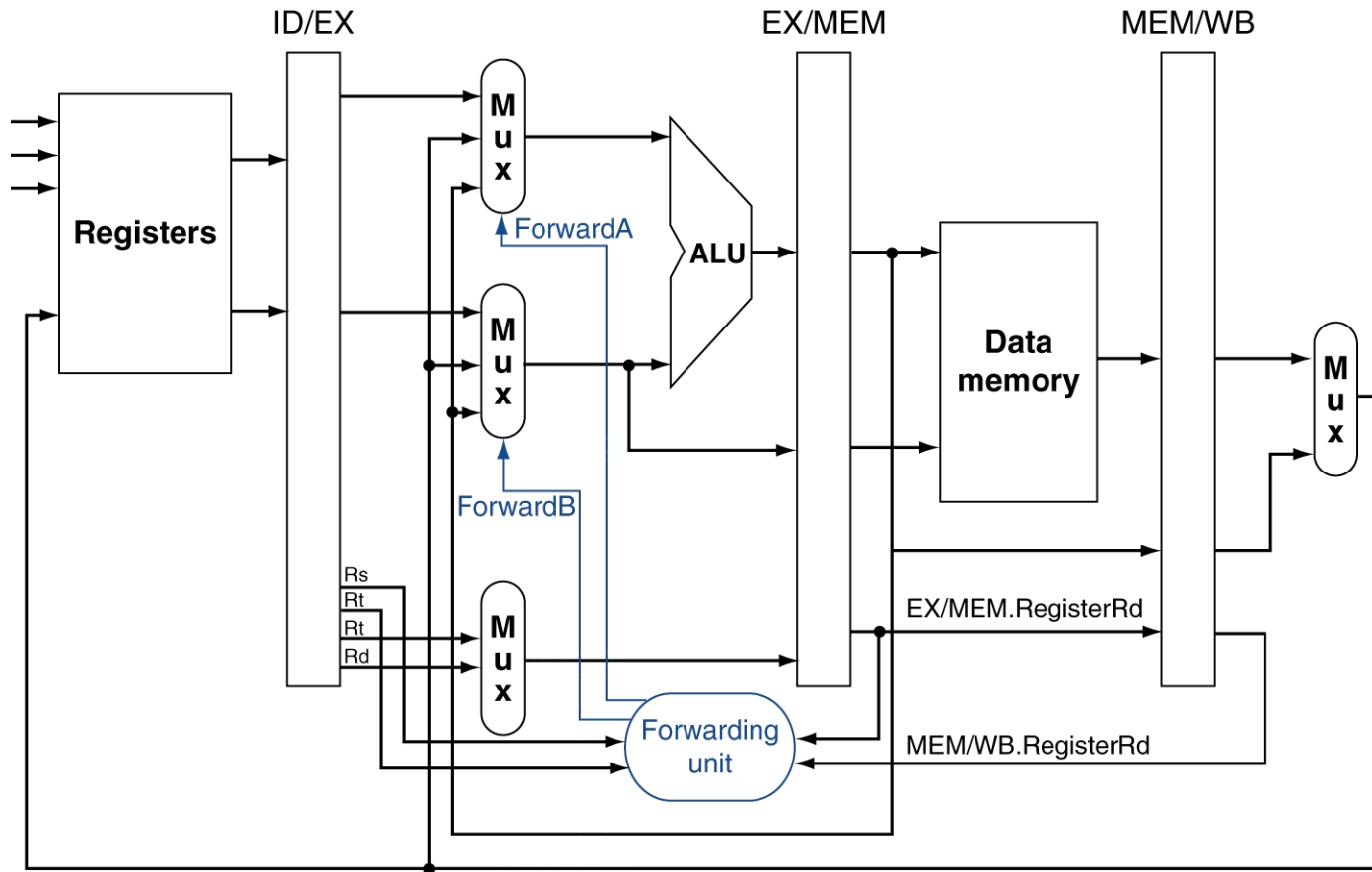 - 2b. MEM/WB.RegisterRd = ID/EX.RegisterRt

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 $zero
  - EX/MEM.RegisterRd ≠ 0,
    MEM/WB.RegisterRd ≠ 0

Forwarding Paths

b. With forwarding

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

Double Data Hazard

- Consider the sequence:

      add $1, $1, $2
      add $1, $1, $3
      add $1, $1, $4

- Both hazards occur
  - Want to use the most recent
- Revise MEM hazard condition
  - Only fwd if EX hazard condition isn't true

Revised Forwarding Condition

- **MEM hazard**

  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
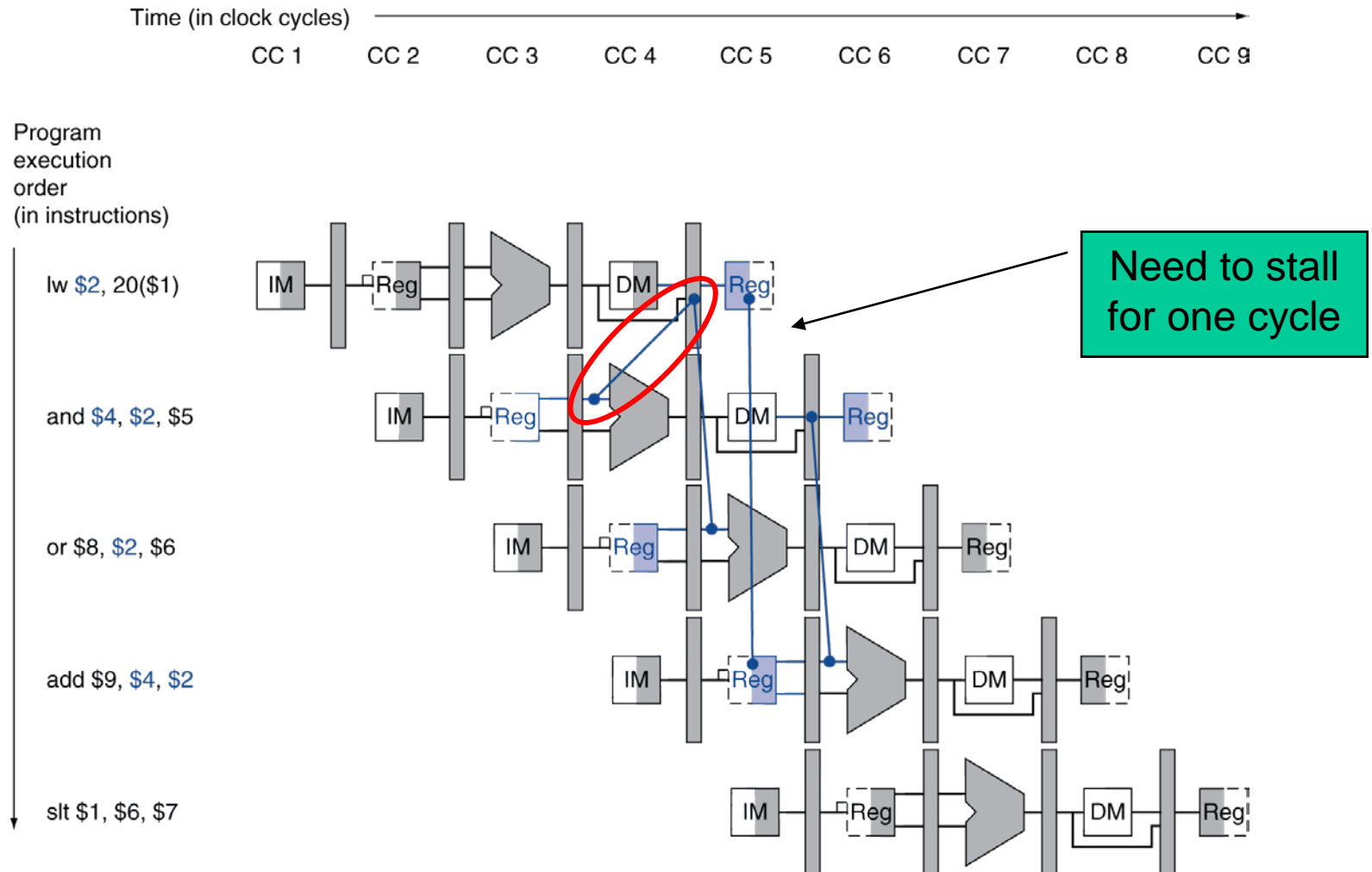   ForwardA = 01

  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Datapath with Forwarding

Load-Use Data Hazard

Time (in clock cycles)

CC 1   CC 2   CC 3   CC 4   CC 5   CC 6   CC 7   CC 8   CC 9

Program execution order (in instructions)

lw $2, 20($1)

and $4, $2, $5

or $8, $2, $6

add $9, $4, $2

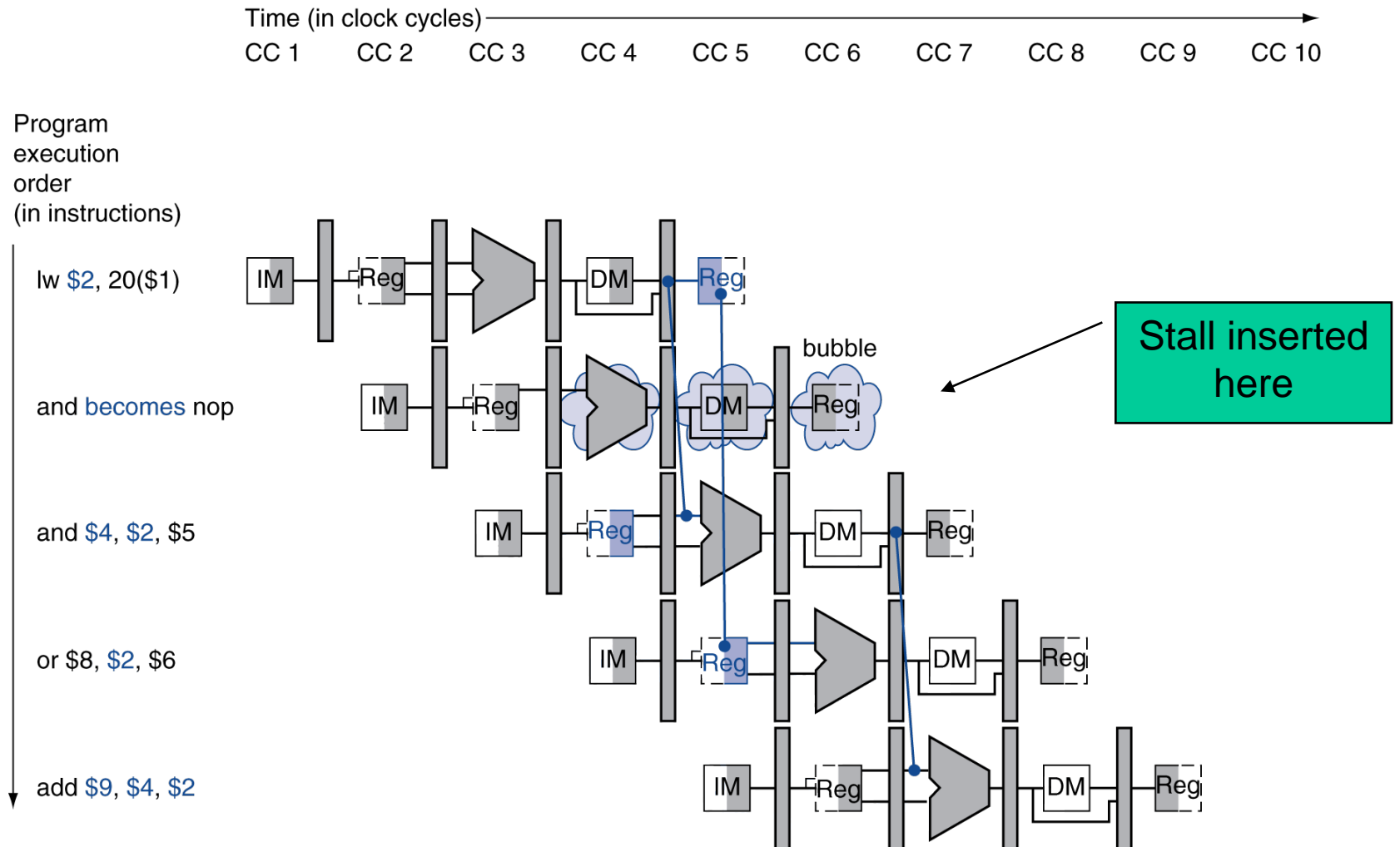slt $1, $6, $7

Need to stall for one cycle

Load-Use Hazard Detection

- Check when using 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))

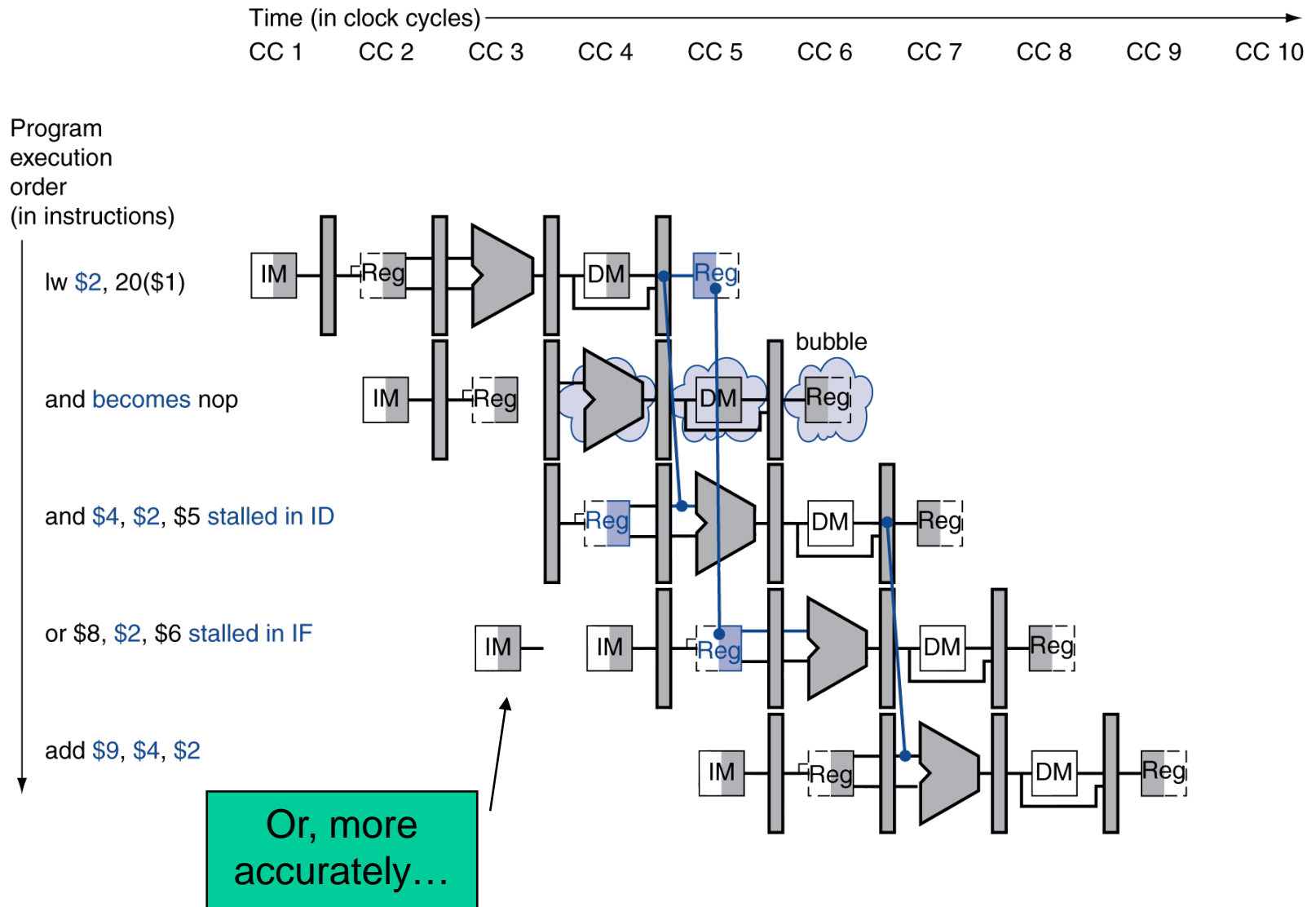- If detected, stall and insert bubble

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
  - Using 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
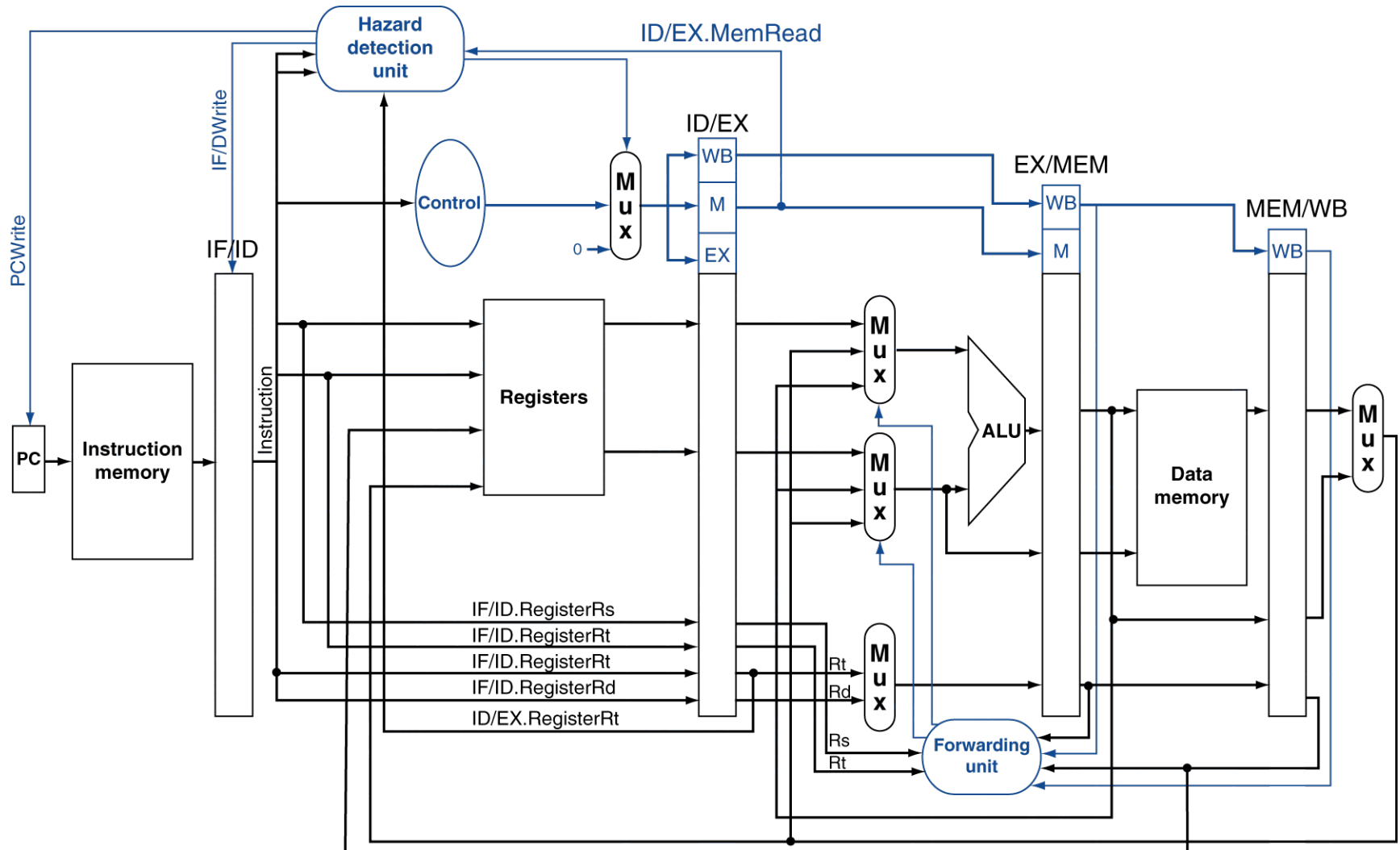

Time (in clock cycles)

CC 1   CC 2   CC 3   CC 4   CC 5   CC 6   CC 7   CC 8   CC 9   CC 10

Program execution order (in instructions)

lw $2, 20($1)

and becomes nop

and $4, $2, $5

or $8, $2, $6

add $9, $4, $2

bubble

Stall inserted here

Stall/Bubble in the Pipeline

Time (in clock cycles)

CC 1   CC 2   CC 3   CC 4   CC 5   CC 6   CC 7   CC 8   CC 9   CC 10

Program
execution
order
(in instructions)

lw $2, 20($1)

and becomes nop

and $4, $2, $5 stalled in ID

or $8, $2, $6 stalled in IF

add $9, $4, $2
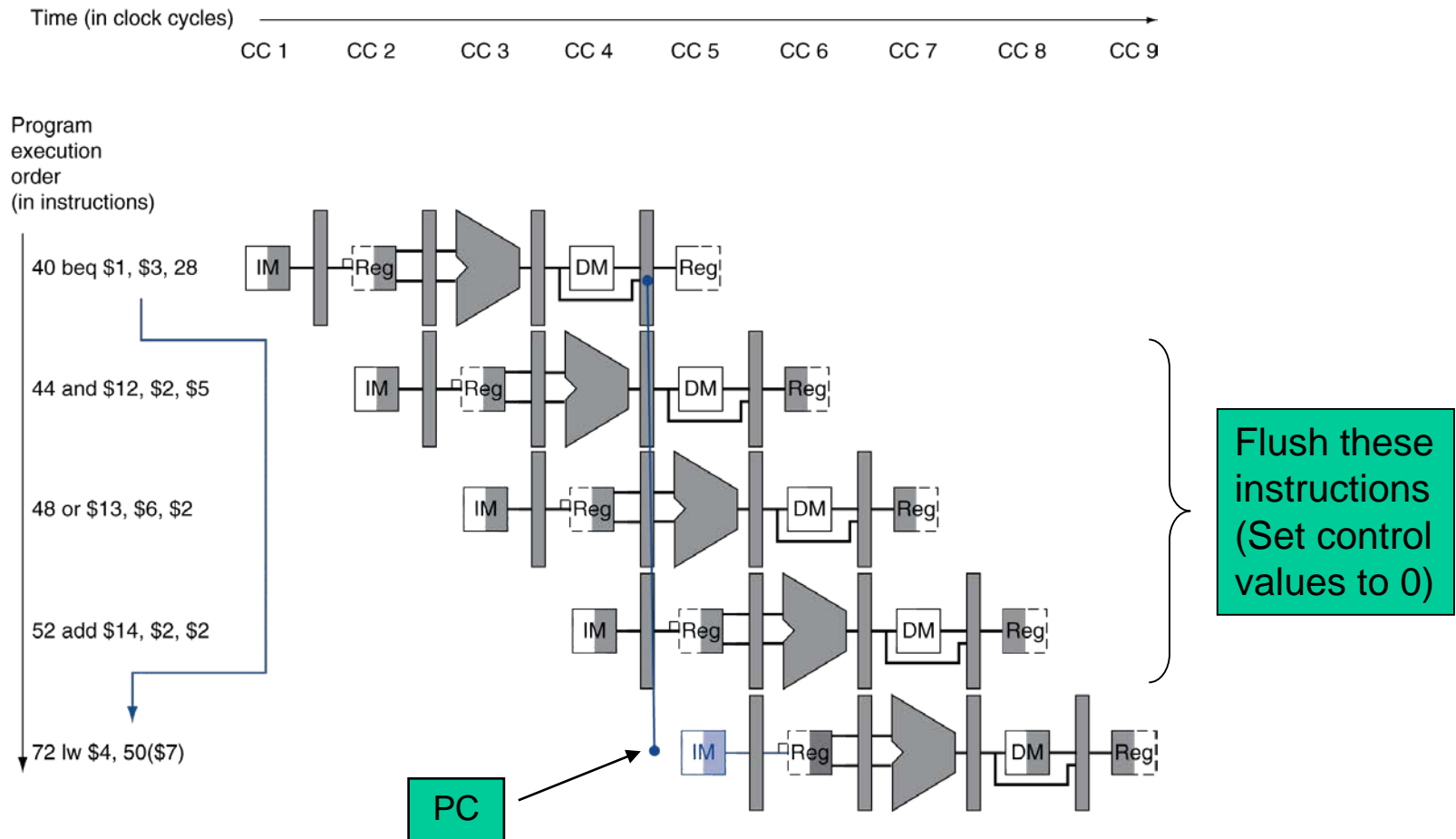
bubble

Or, more accurately…

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

Branch Hazards

- If branch outcome determined in MEM

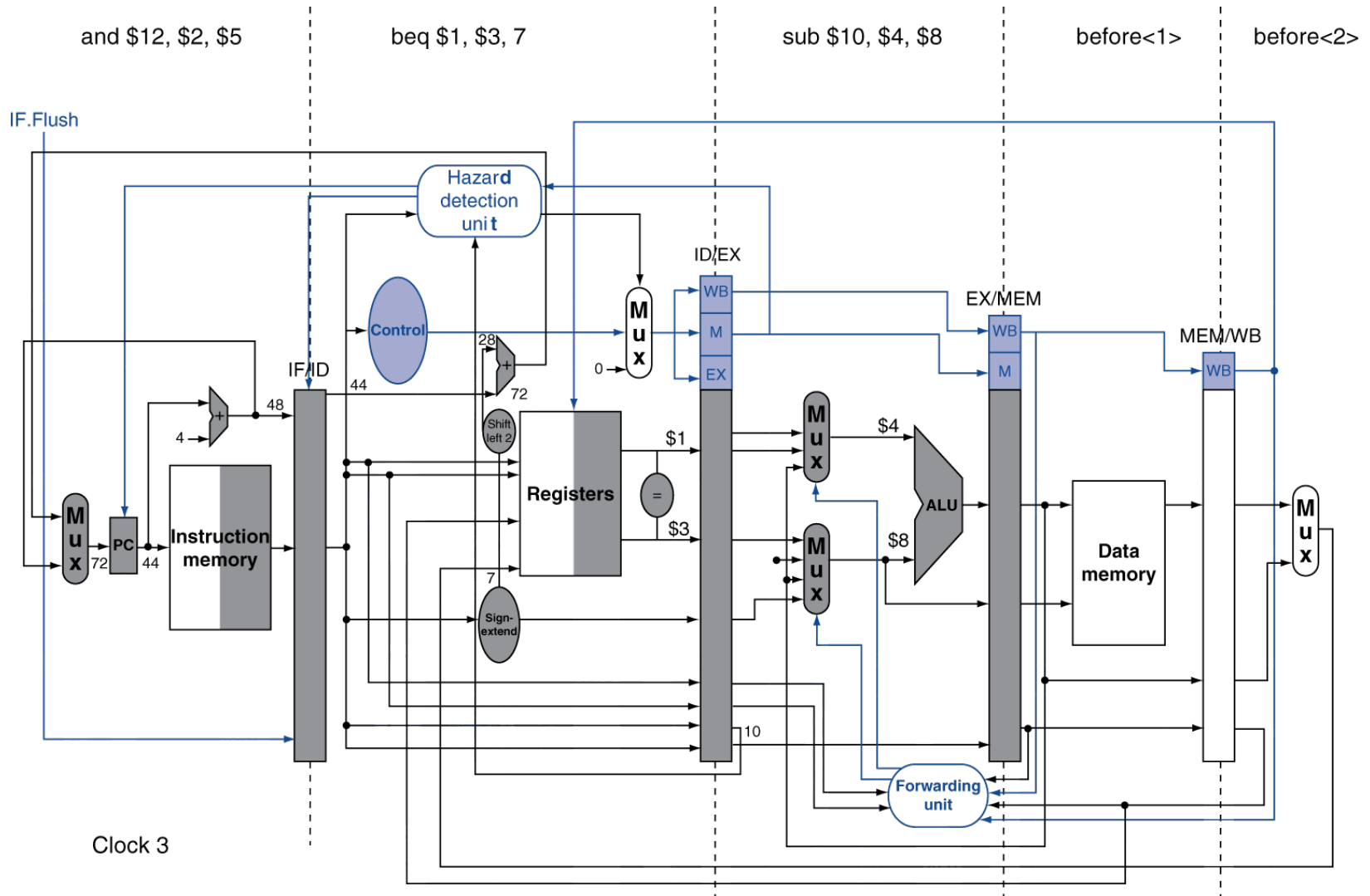Flush these instructions (Set control values to 0)

PC

Reducing Branch Delay

- Move hardware to determine outcome to ID stage
  - Target address adder
  - Register comparator
- Example: branch taken

```
36:    sub    $10,  $4,  $8
40:    beq    $1,   $3,  7
44:    and    $12,  $2,  $5
48:    or     $13,  $2,  $6
52:    add    $14,  $4,  $2
56:    slt    $15,  $6,  $7
       . . .
72:    lw     $4,   50($7)
```

Example: Branch Taken

Example: Branch Taken