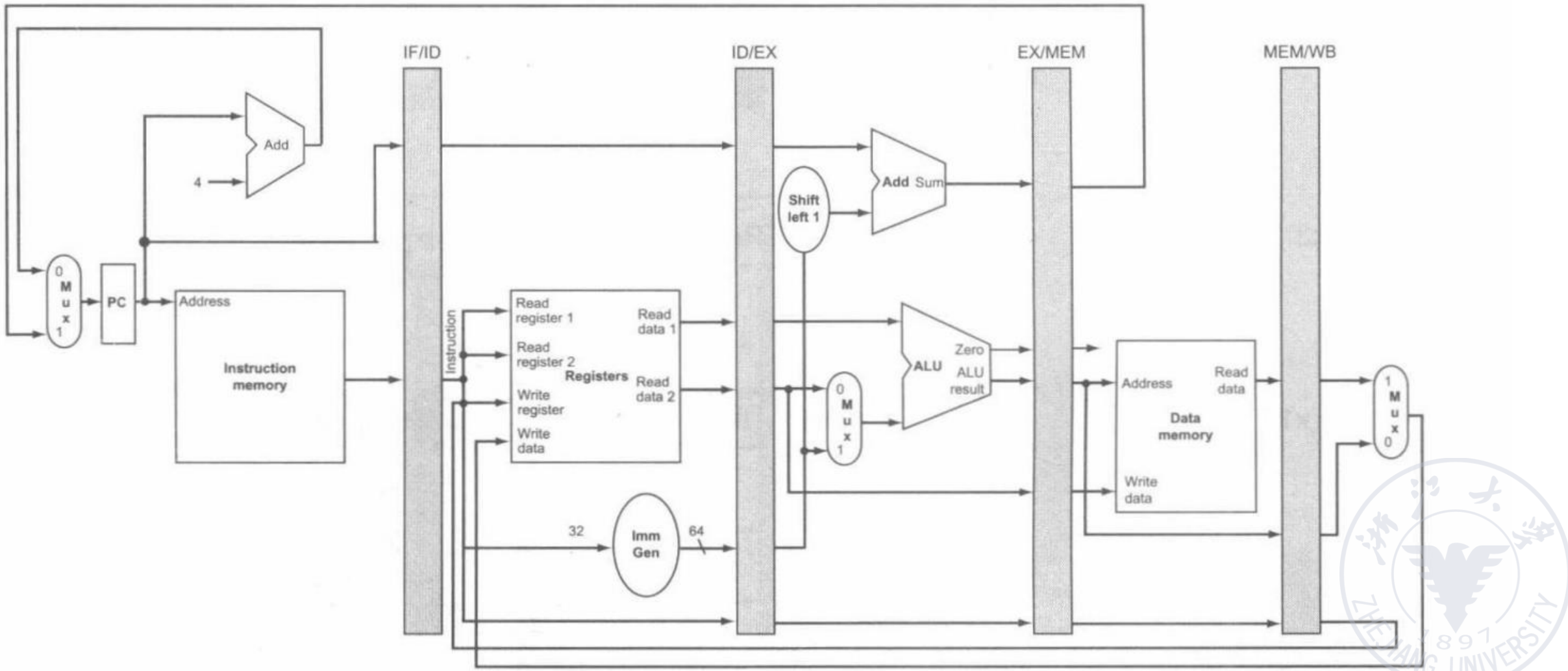
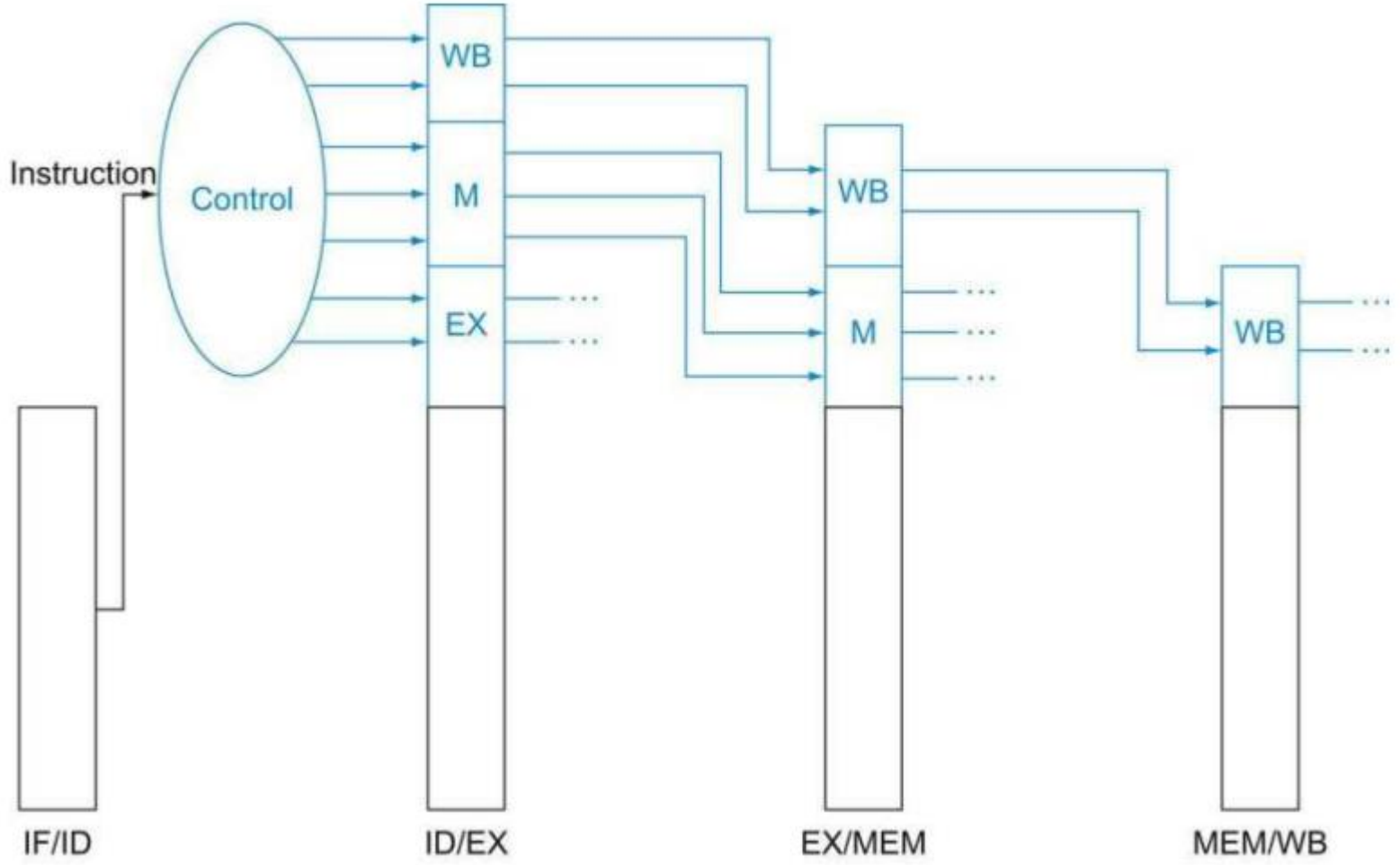
Computer Systems II

Li Lu
Room 605, CaoGuangbiao Building
li.lu@zju.edu.cn
https://person.zju.edu.cn/lynnluli

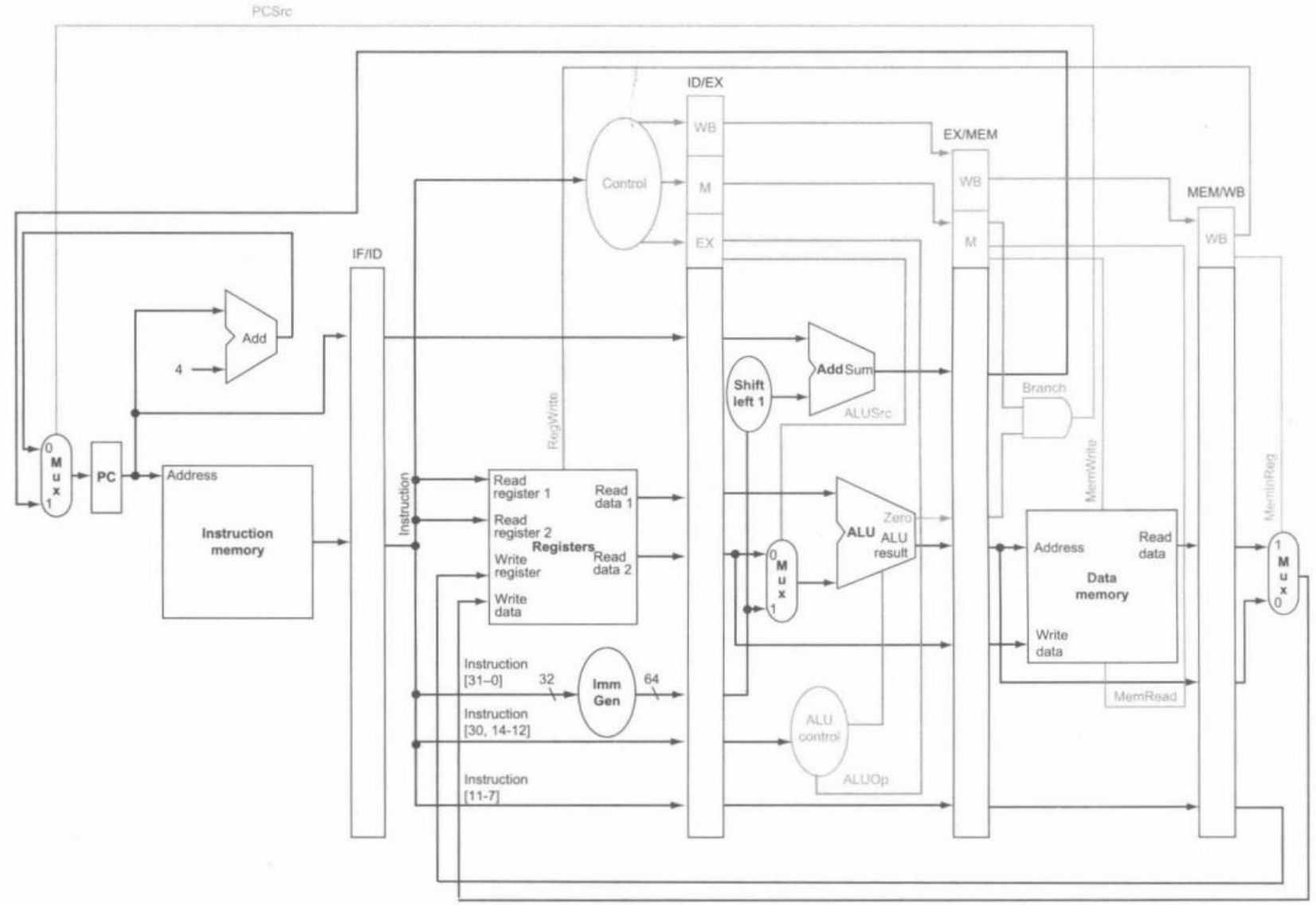
Pipelining Datapath



Extend Pipeline Registers for Controller



Pipelined Datapath with the Control Signals



Pipeline Hazards

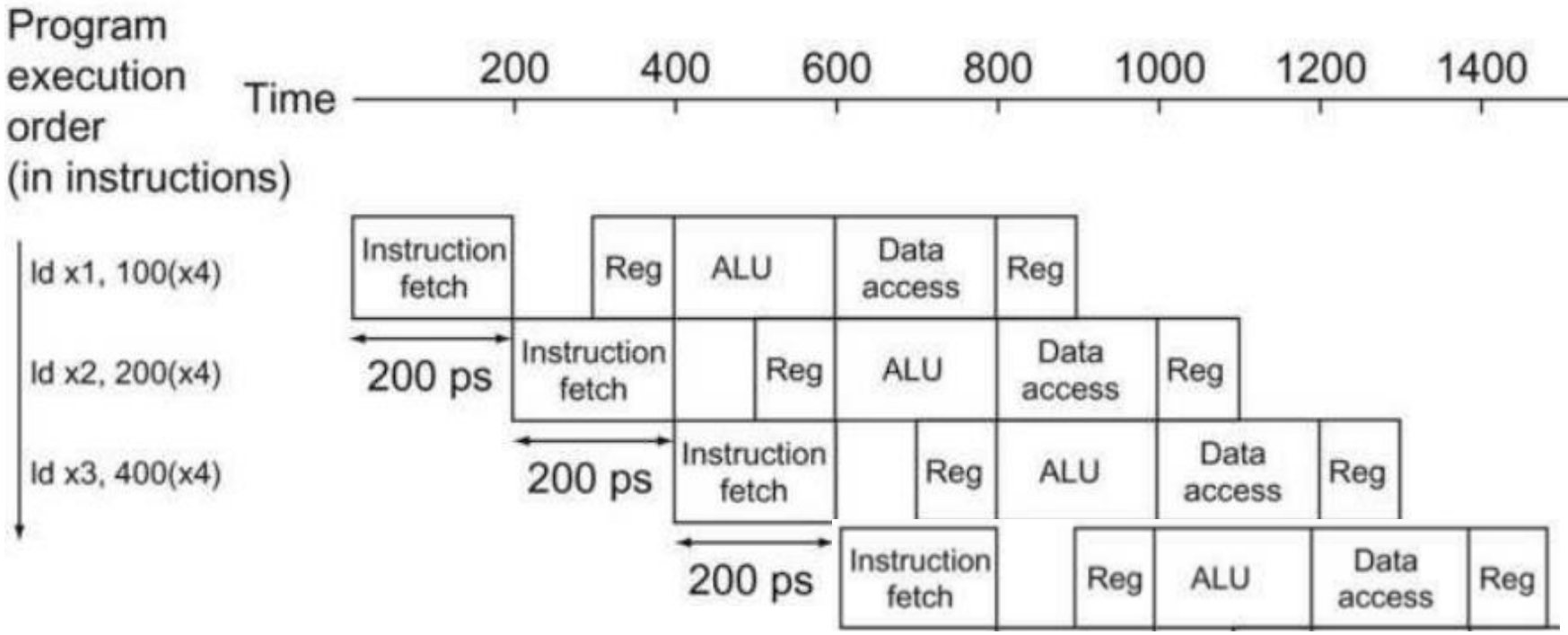
- Structural Hazard — A required resource is busy
- Data Hazards
  - Data dependency between instructions
  - Need to wait for previous instruction to complete its data read/write
- Control Hazards — Flow of execution depends on previous instruction

Structural Hazard

Structural Hazard

A required resource is busy

Example: Consider the situation while the pipeline only has a single memory



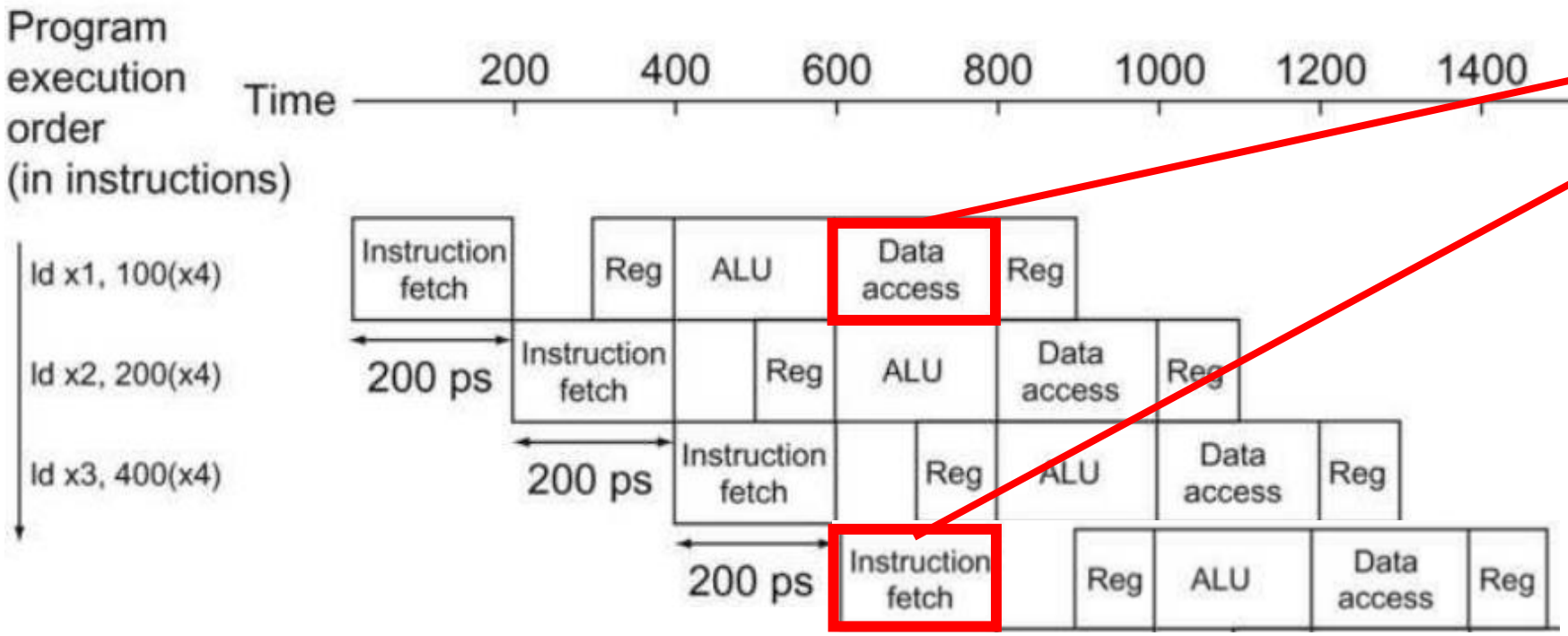
Question: Can the four instructions execute correctly?

Structural Hazard

A required resource is busy

Solution:
Use Instruction and data memory simultaneously.

Example: Consider the situation while the pipeline only has a single memory



How to Deal with Structural Hazard?

Problem: Two or more instructions in the pipeline compete for access to a single physical resource

- Solution 1: Instructions take it in turns to use resource, some instructions have to stall.

- Solution 2:Add more hardware to machine.

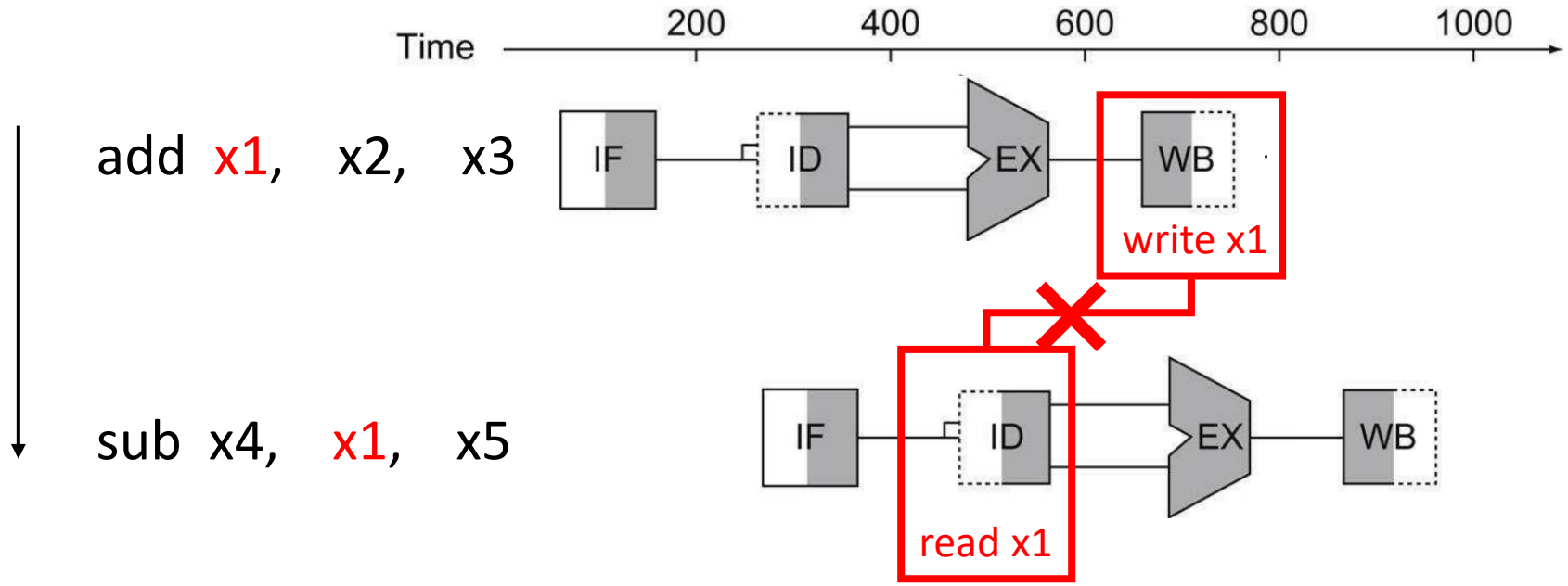
Can always solve a structural hazard by adding more hardware

Data Hazards

Data Hazard

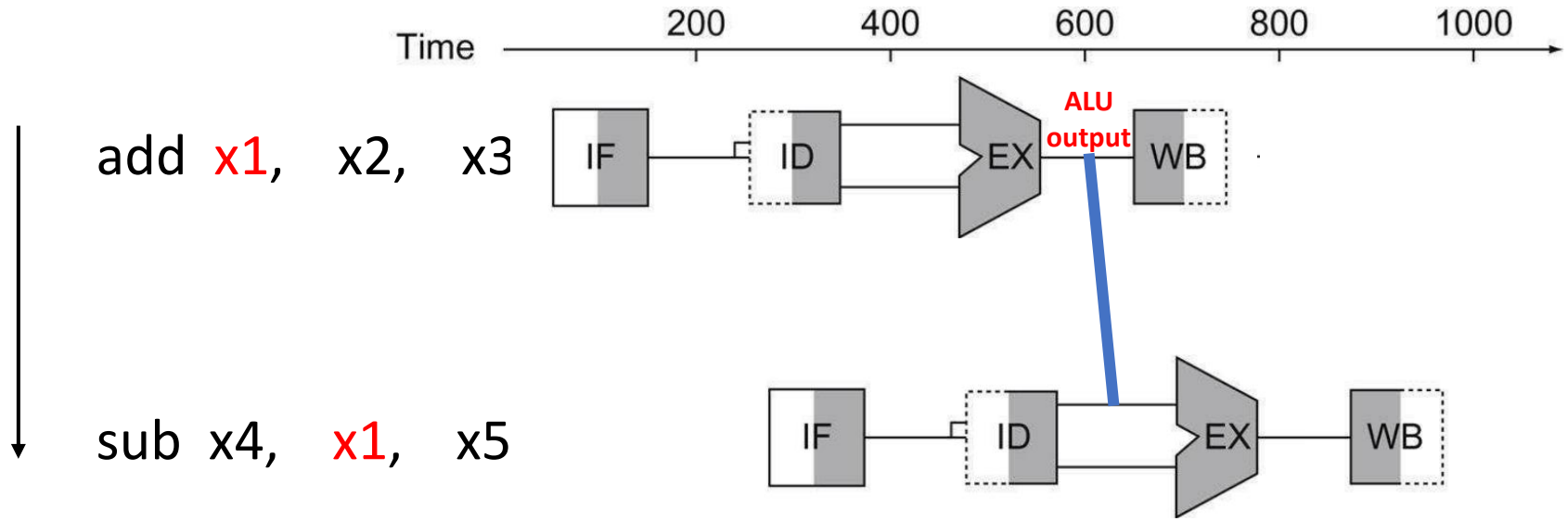
- Data dependency between instructions
- Need to wait for previous instruction to complete its data read/write

Problem: Instruction depends on result from previous



Data Hazard

Solution "forwarding": Adding extra hardware to retrieve the missing item early from the internal resources



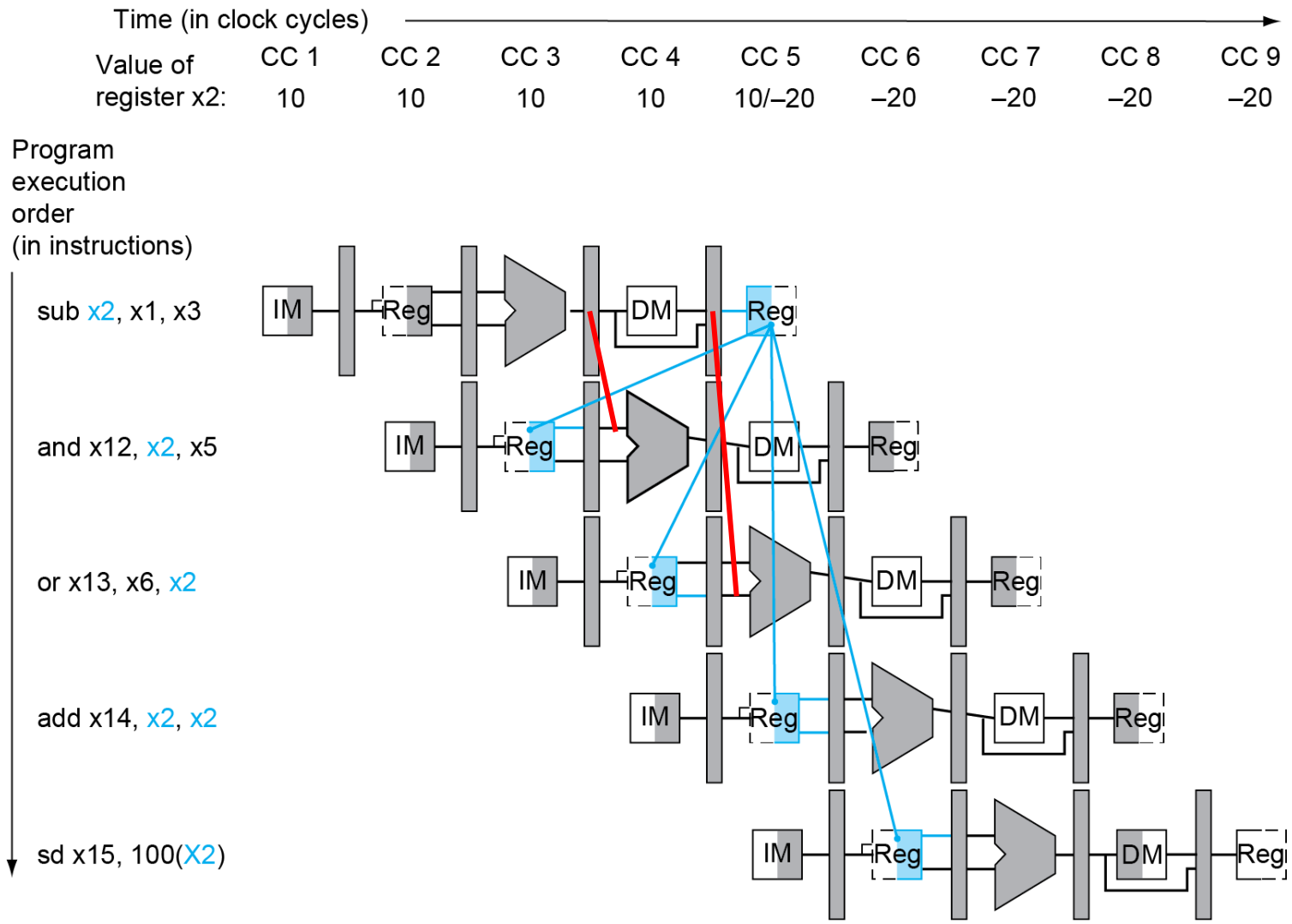
Data Hazard in ALU Instructions

- Consider this sequence:

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

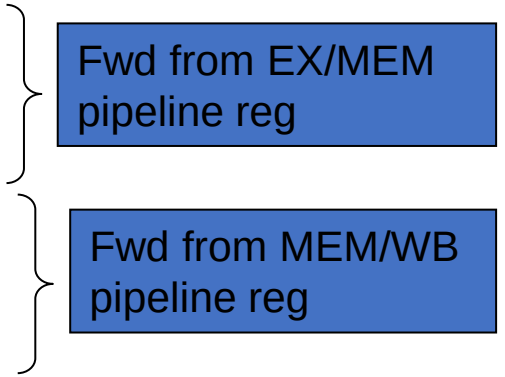
- We can resolve hazards with forwarding
  - How do we detect when to forward?

Dependencies & Forwarding



Detecting the Need to Forward

- Pass register numbers along pipeline
  - e.g., ID/EX.Rs1 = register number for Rs1 sitting in ID/EX pipeline register
- ALU operand register numbers in EX stage are given by
  - ID/EX.Rs1, ID/EX.Rs2
- Data hazards when
  - 1a. EX/MEM. Rd = ID/EX. Rs1
  - 1b. EX/MEM. Rd = ID/EX. Rs2
  - 2a. MEM/WB. Rd = ID/EX. Rs1
  - 2b. MEM/WB. Rd = ID/EX. Rs2

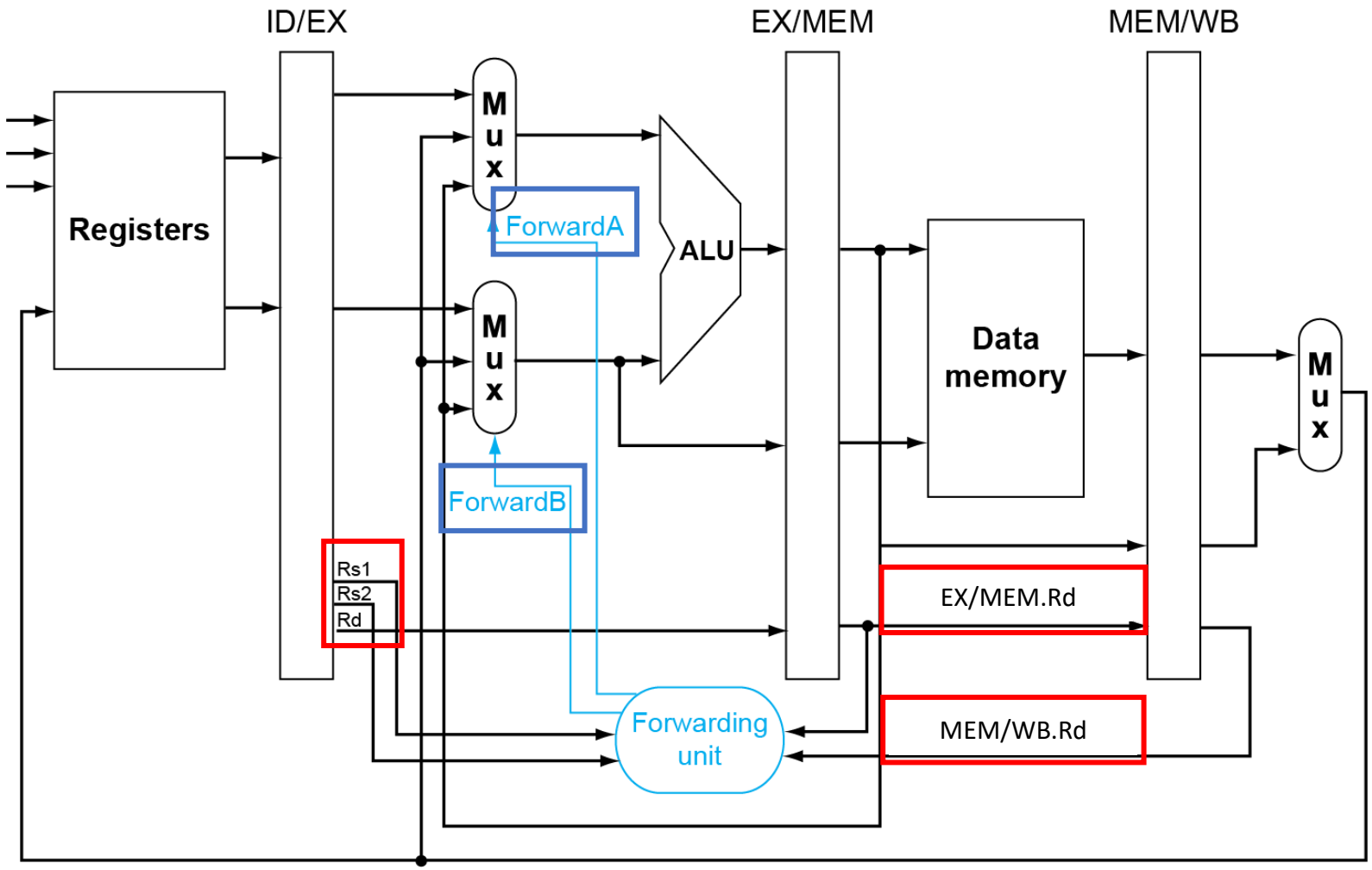


Detecting the Need to Forward

- 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. Rd ≠ 0,
    MEM/WB. Rd ≠ 0

Forwarding Paths

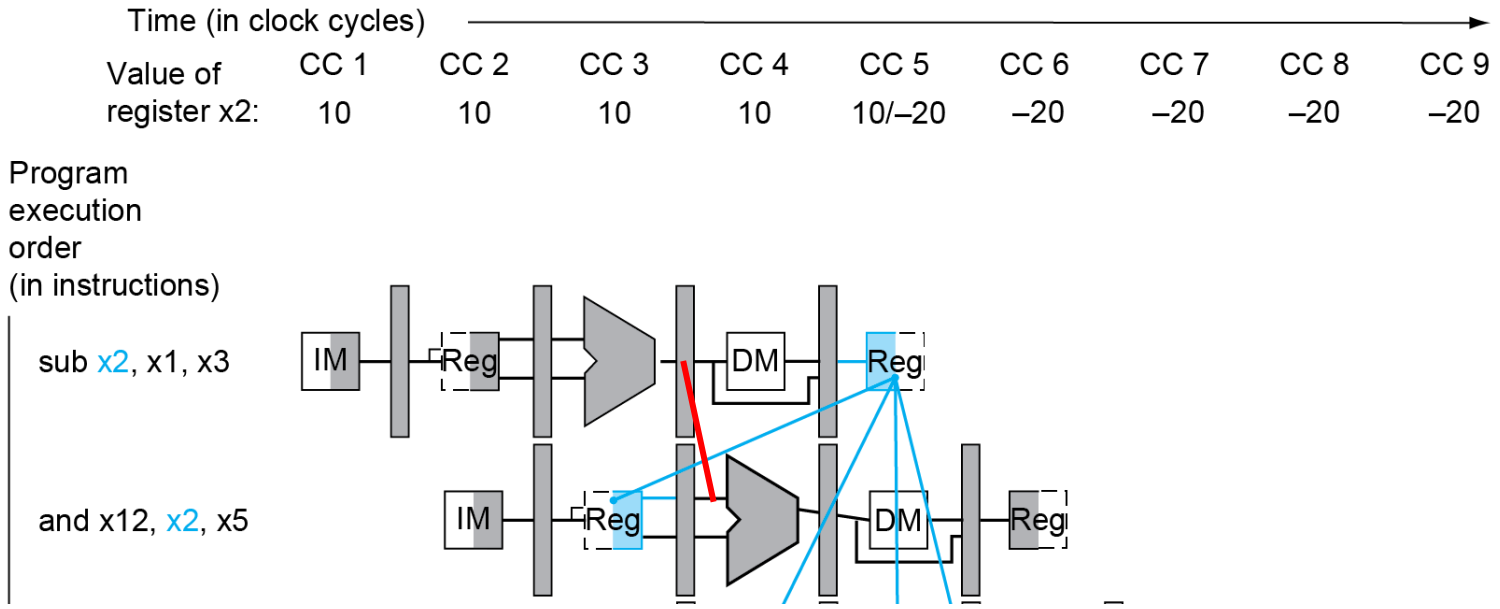


Forwarding Conditions

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

Example 1

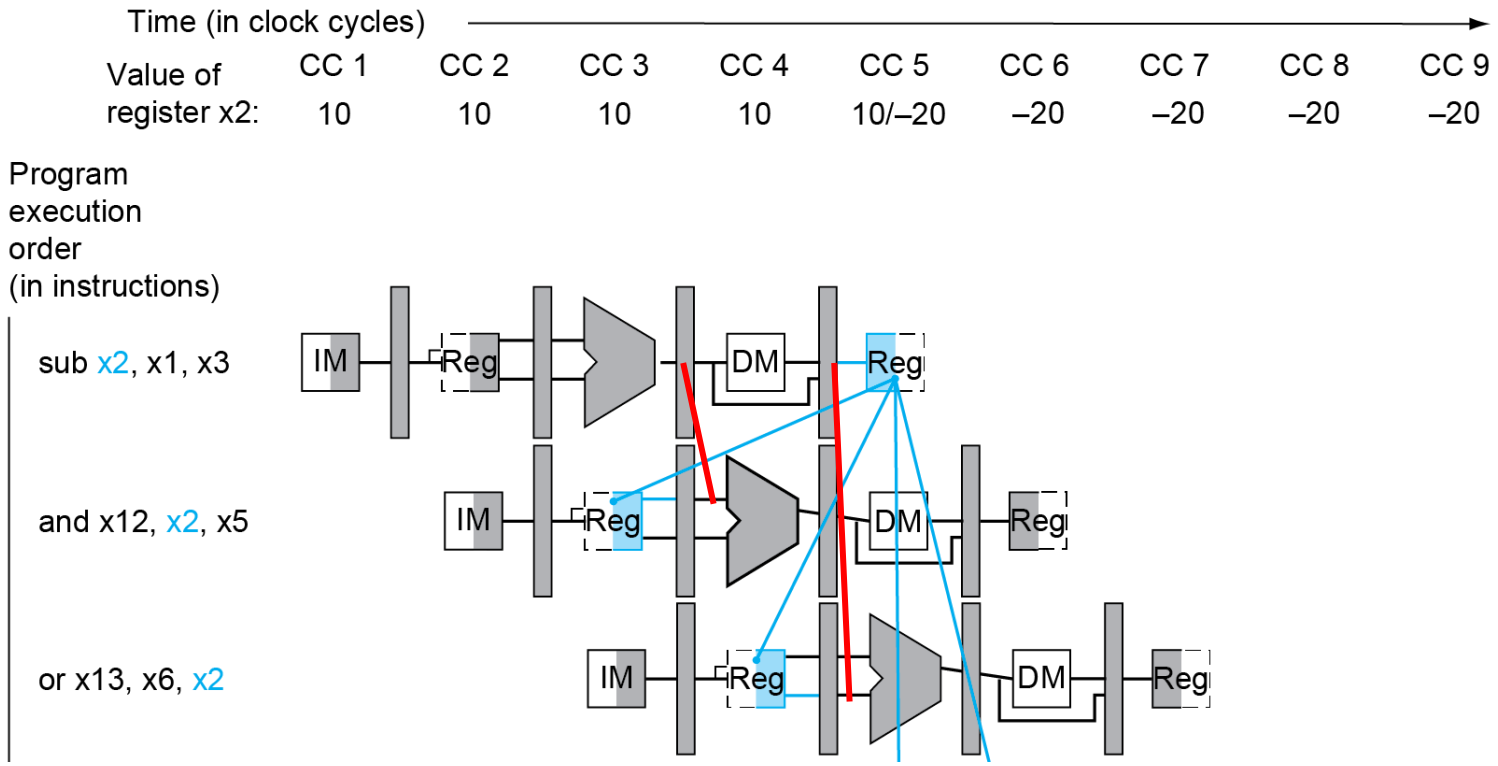
What is the Data hazards condition in the following case?



EX/MEM. Rd = ID/EX. Rs1

Example 2

What is the Data hazards condition in the following case?



MEM/WB. Rd = ID/EX. Rs2

Forwarding Conditions

Mux control	Source	Explanation
ForwardA/B = 00	ID/EX	No forwarding
ForwardA/B = 10	EX/MEM	Forwarding with data hazard in EX/MEM
ForwardA/B = 01	MEM/WB	Forwarding with data hazard in MEM/WB

Forwarding Conditions

- EX hazard
  - if (EX/MEM.RegWrite and (EX/MEM. Rd ≠ 0)
    and (EX/MEM. Rd = ID/EX. Rs1))
    ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM. Rd ≠ 0)
    and (EX/MEM. Rd = ID/EX. Rs2))
    ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)
    and (MEM/WB. Rd = ID/EX. Rs1))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)
    and (MEM/WB. Rd = ID/EX. Rs2))
    ForwardB = 01

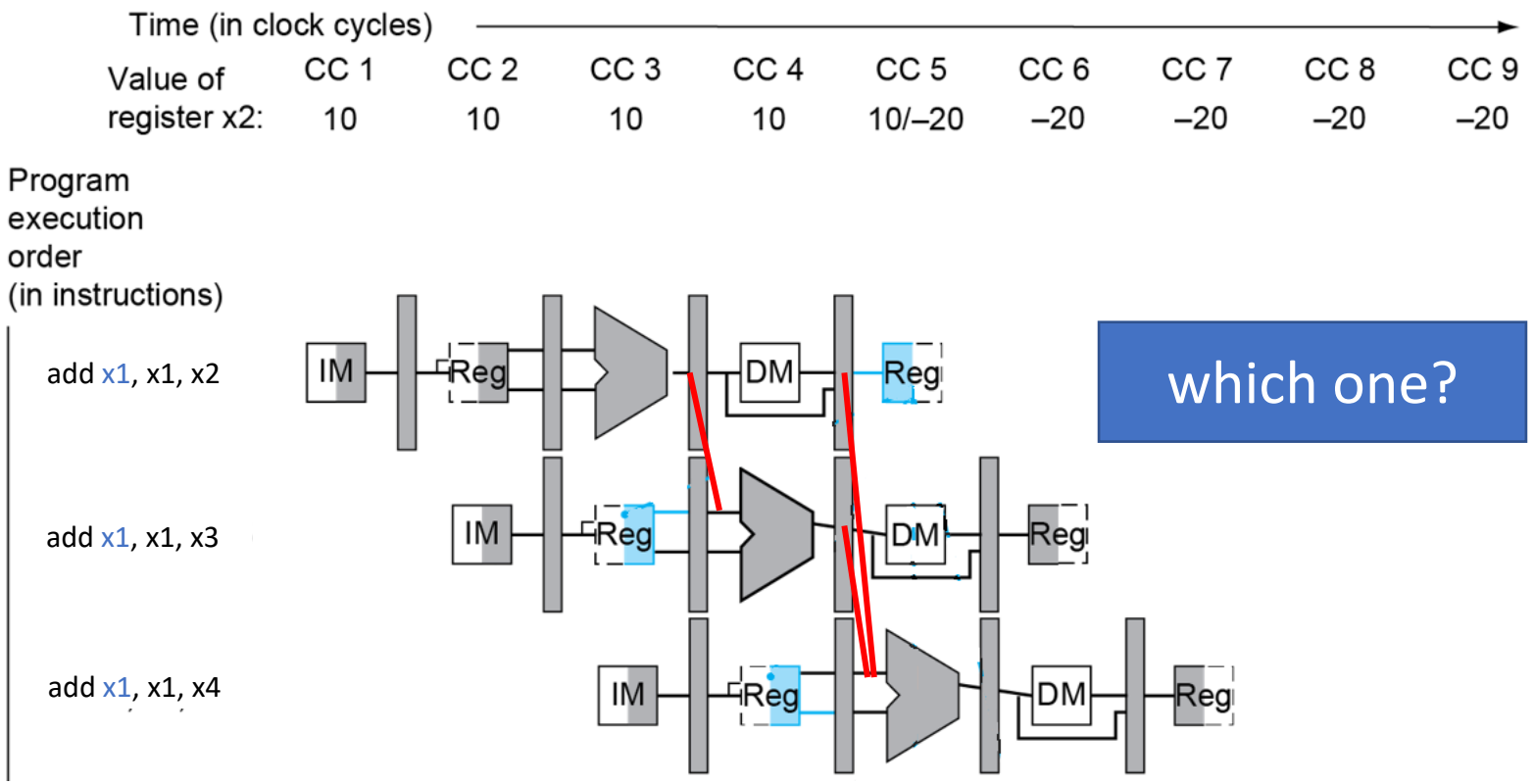
Double Data Hazard

- Consider the sequence:

```
add x1,x1,x2
add x1,x1,x3
add x1,x1,x4
```

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

Double Data Hazard



Such an exception should be added into MEM hazards!

Revised Forwarding Condition

- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)
    and not(EX/MEM.RegWrite and (EX/MEM. Rd ≠ 0)
    and (EX/MEM. Rd ≠ ID/EX. Rs1))
    and (MEM/WB. Rd = ID/EX. Rs1))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)
    and not(EX/MEM.RegWrite and (EX/MEM. Rd ≠ 0)
    and (EX/MEM. Rd ≠ ID/EX. Rs2))
    and (MEM/WB. Rd = ID/EX. Rs2))
    ForwardB = 01

Revised Forwarding Condition

- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)

    and not(EX hazard)

    and (MEM/WB. Rd = ID/EX. Rs1))

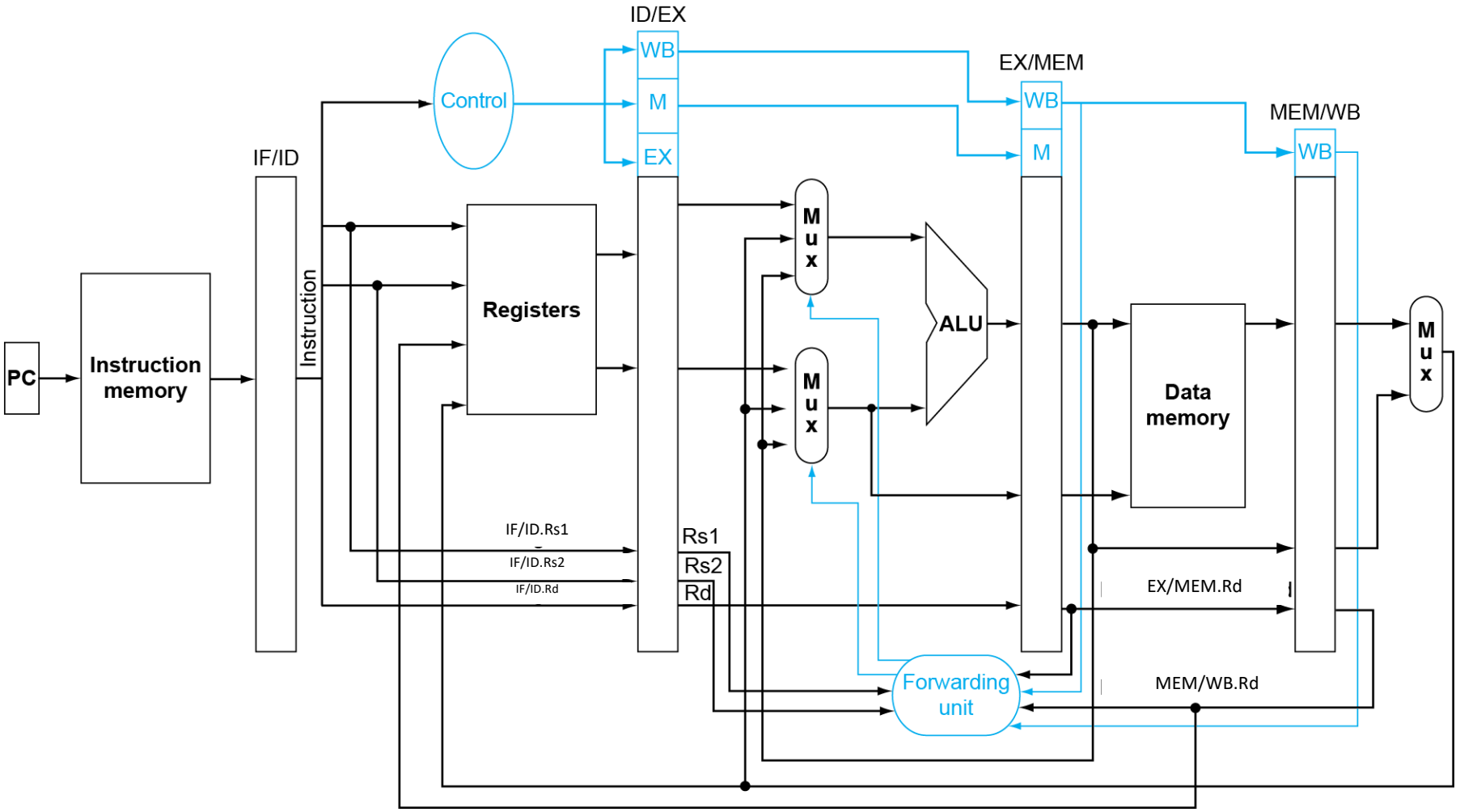
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB. Rd ≠ 0)

    and not(EX hazard)

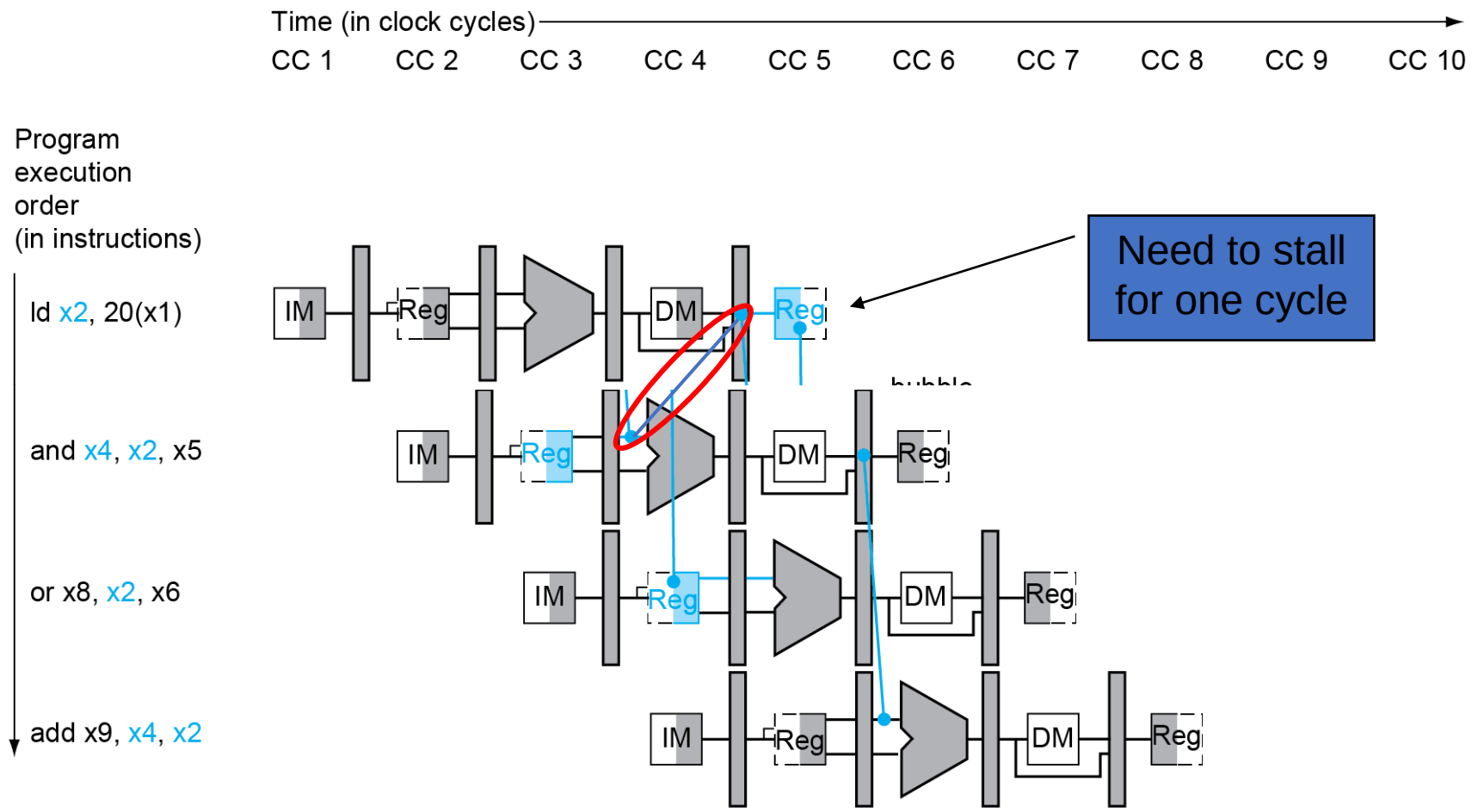
    and (MEM/WB. Rd = ID/EX. Rs2))

    ForwardB = 01

Datapath with Forwarding



Load-Use Data Hazard



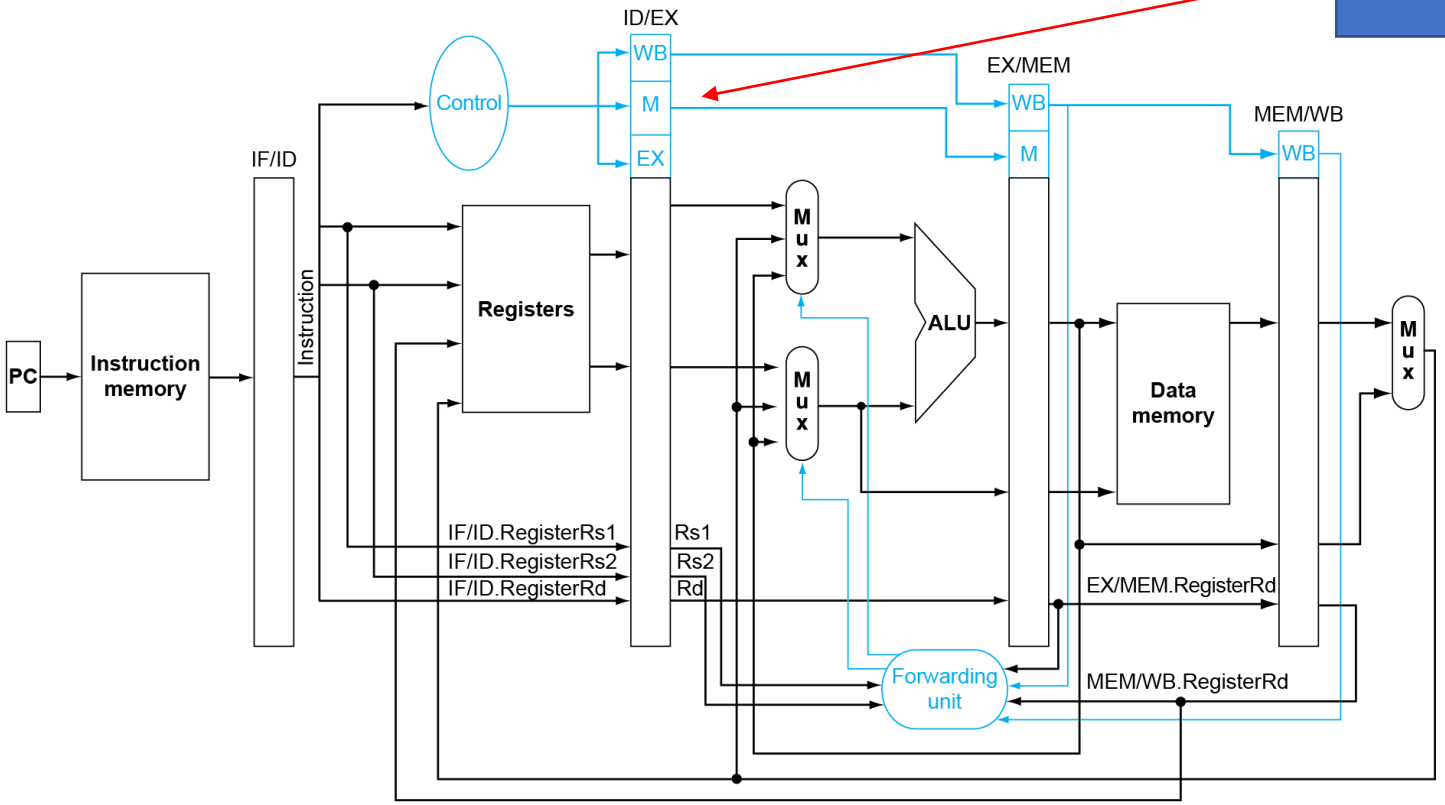
Load-Use Hazard Detection

- Check when using instruction is decoded in ID stage
- ALU operand register numbers in ID stage are given by
  - IF/ID. Rs1, IF/ID. Rs2
- Load-use hazard when
  - ID/EX.MemRead and
    ((ID/EX. Rd = IF/ID. Rs1) or
    (ID/EX. Rd = IF/ID. Rs2))
- If detected, stall and insert bubble

How to Stall the Pipeline

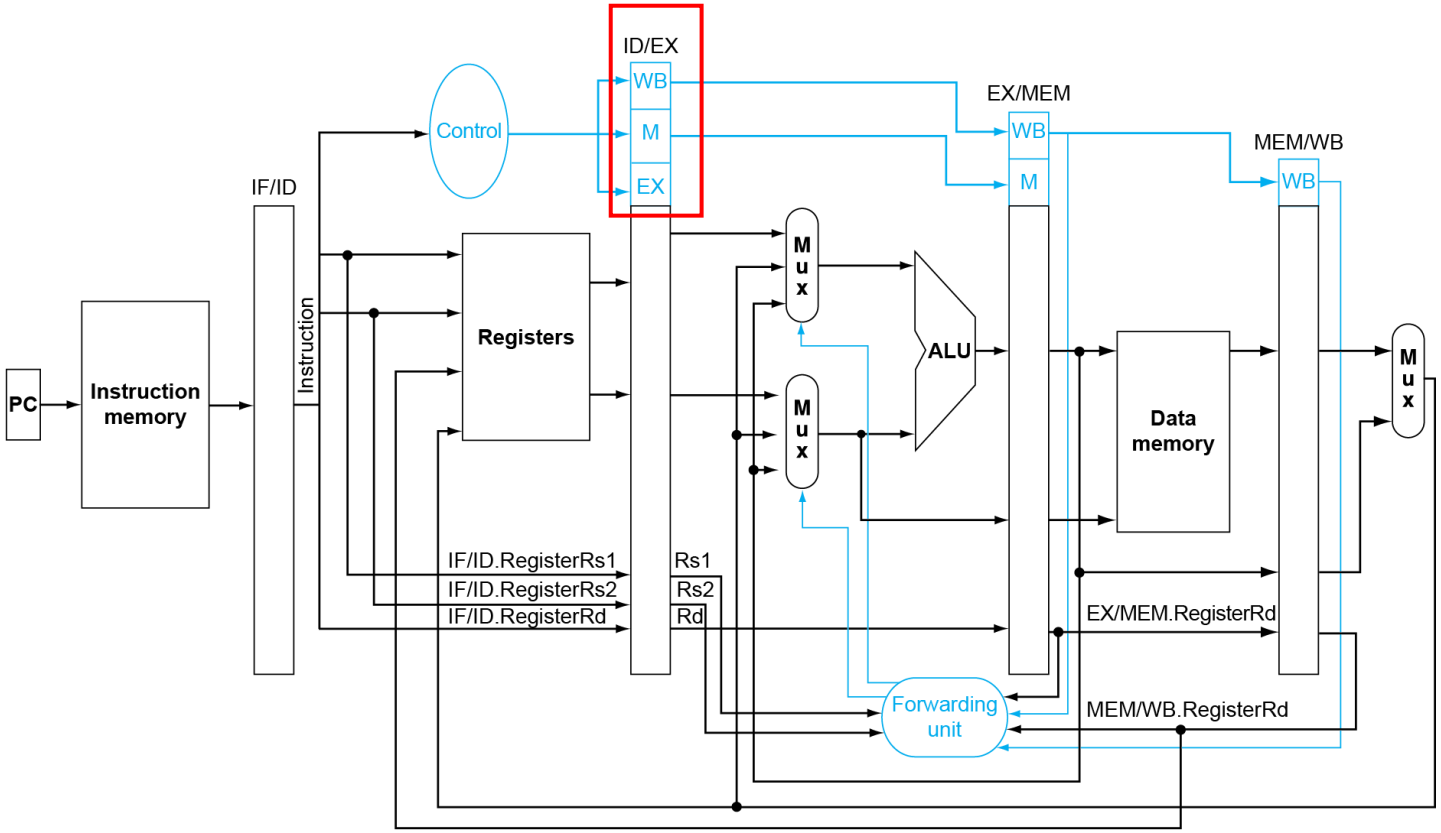
- Force control values in ID/EX register to 0
  - EX, MEM and WB do nop (no-operation)

Which one?



How to Stall the Pipeline

- Force control values in ID/EX register to 0
  - EX, MEM and WB do nop (no-operation)

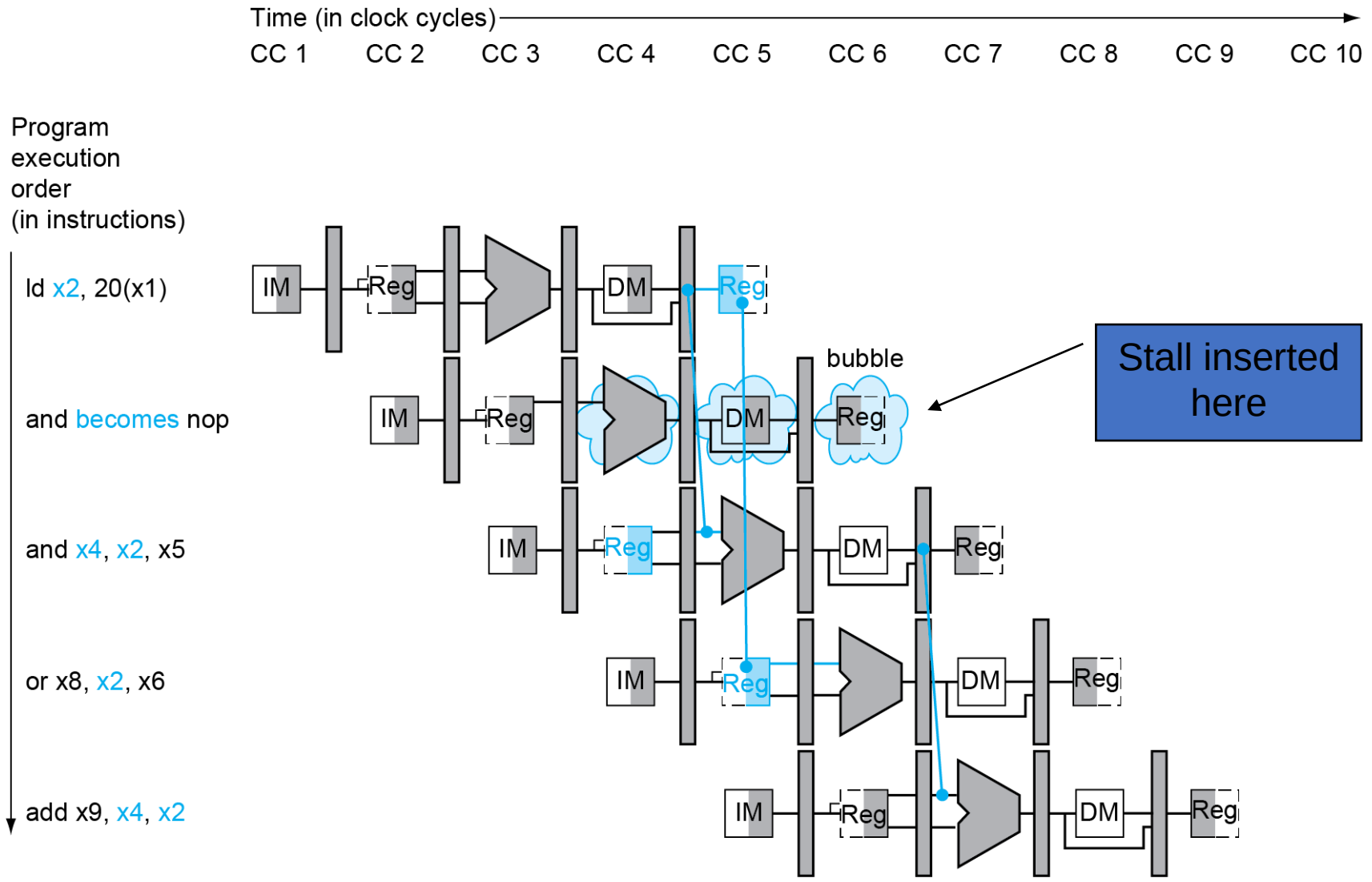


What's more?

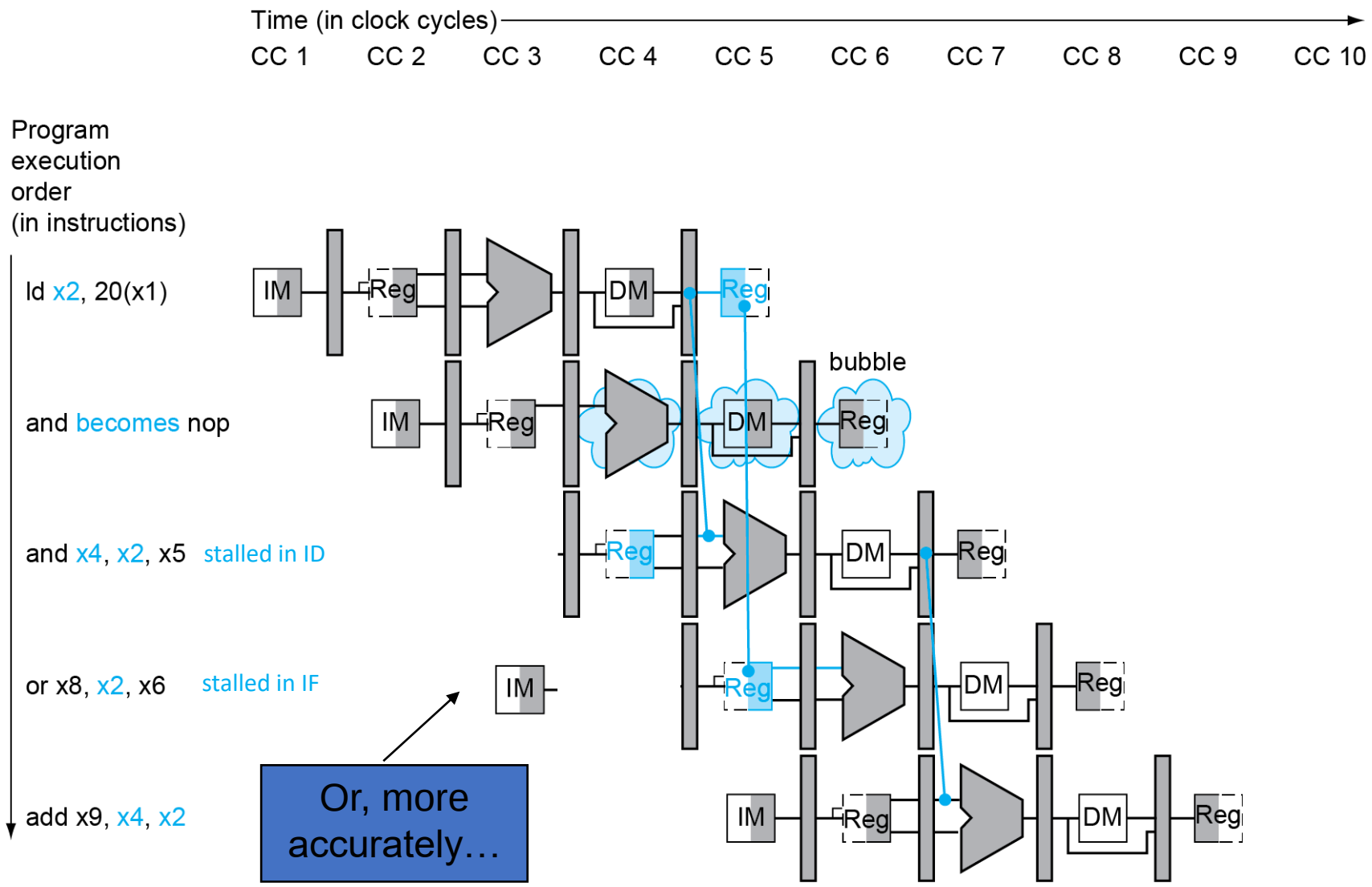
How to Stall the Pipeline

- 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 ld
    - Can subsequently forward to EX stage

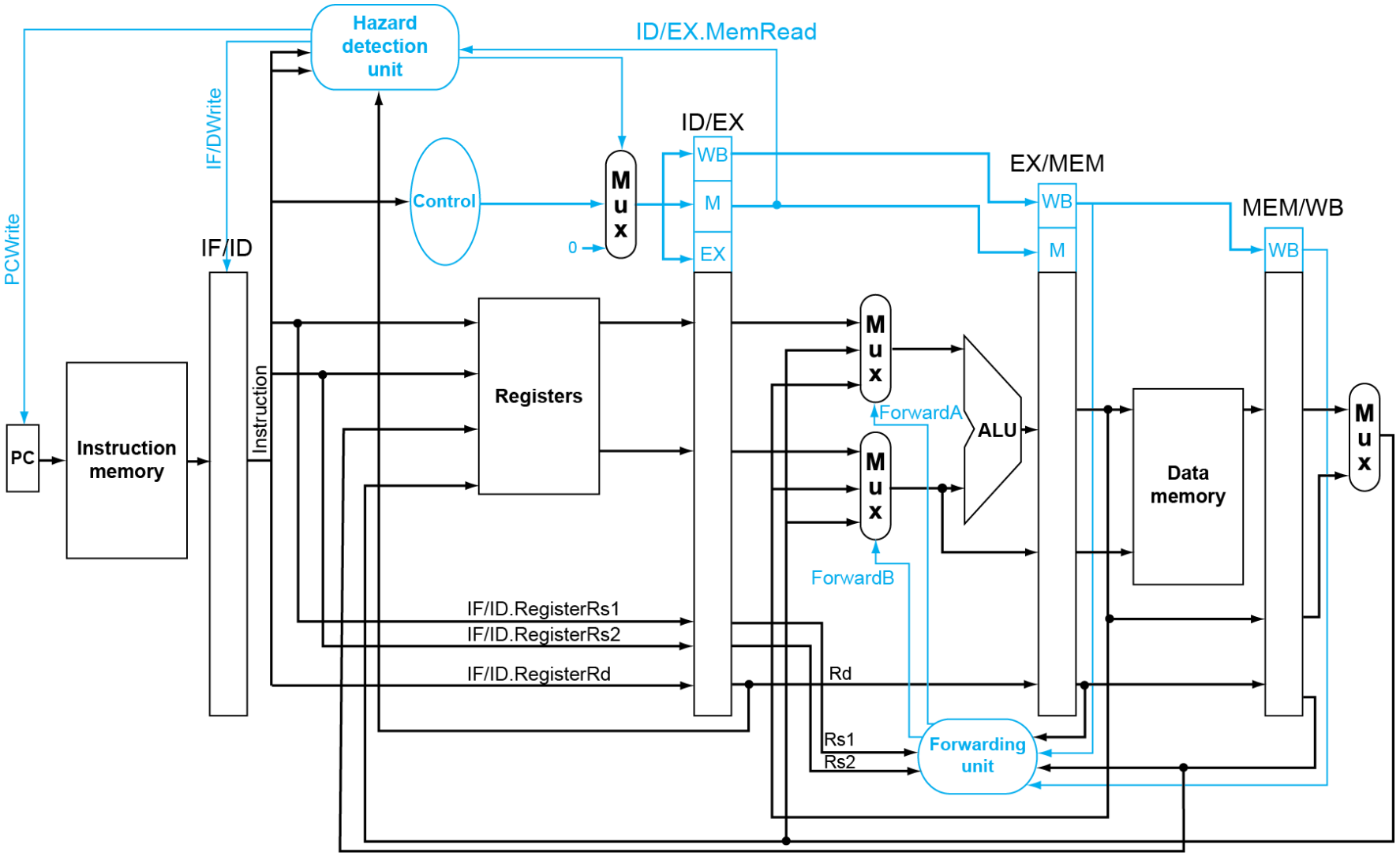
Stall/Bubble in the Pipeline



Stall/Bubble in the Pipeline



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