

Introduction to Computer Architecture

Chapter 4 - 5

Pipeline Hazards

Hyungmin Cho

Department of Computer Science and Engineering
Sungkyunkwan University

Data Hazards in ALU Instructions

- Data hazard example:

add **x2**, x10, x11

sub **x2**, x1, x2

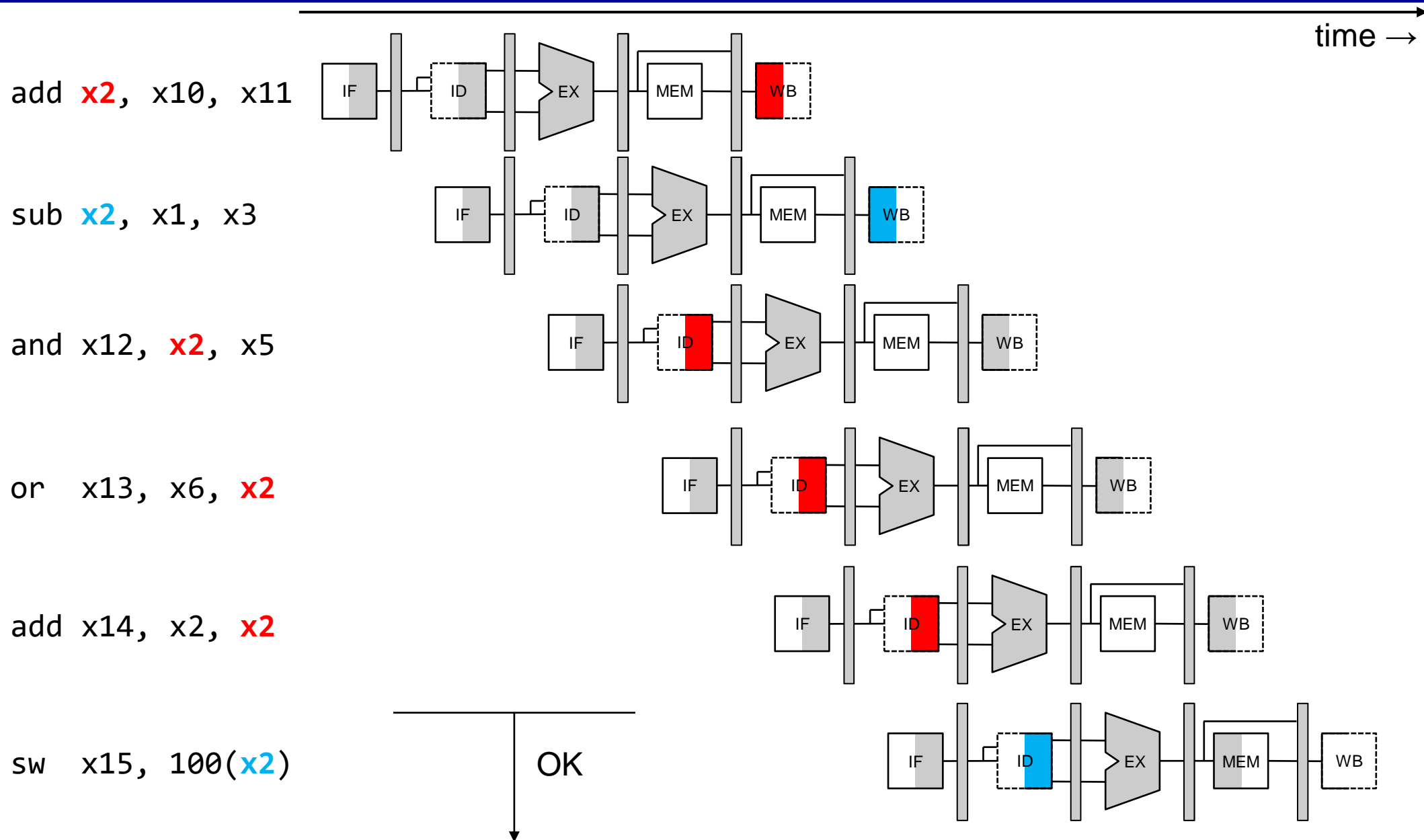
and x12, **x2**, x5

or x13, x6, **x2**

add x14, x2, **x2**

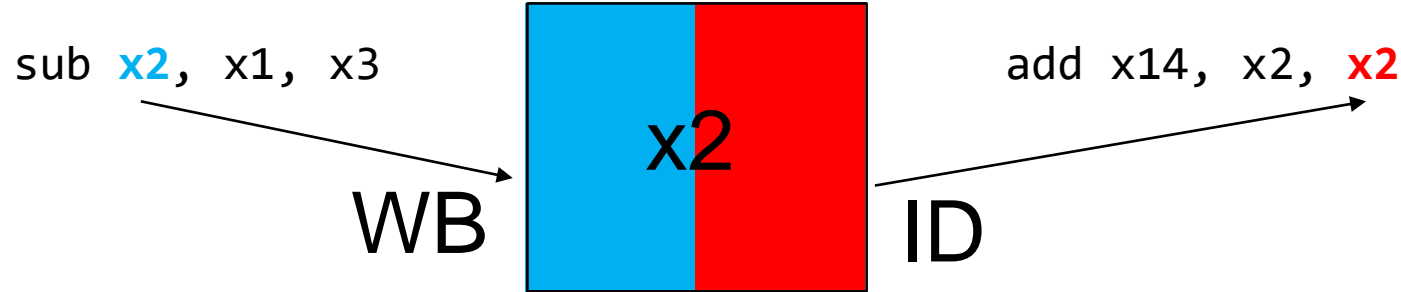
sw x15, 100(**x2**)

Dependencies & Forwarding



Same Register Read & Write

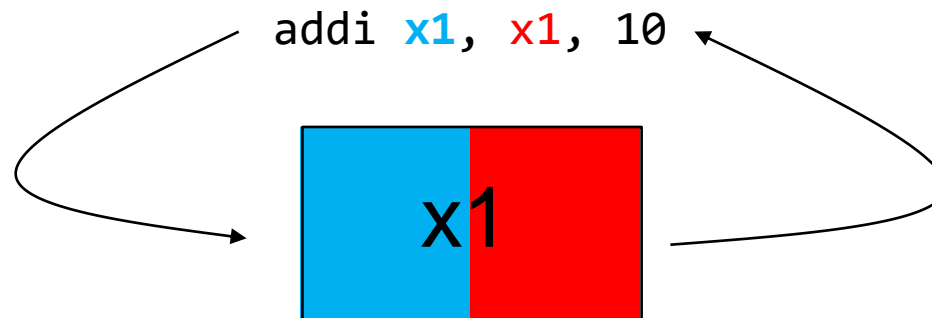
- What happens if you read & write the same register simultaneously?



- Single-cycle CPU: The CPU reads the OLD value before the write

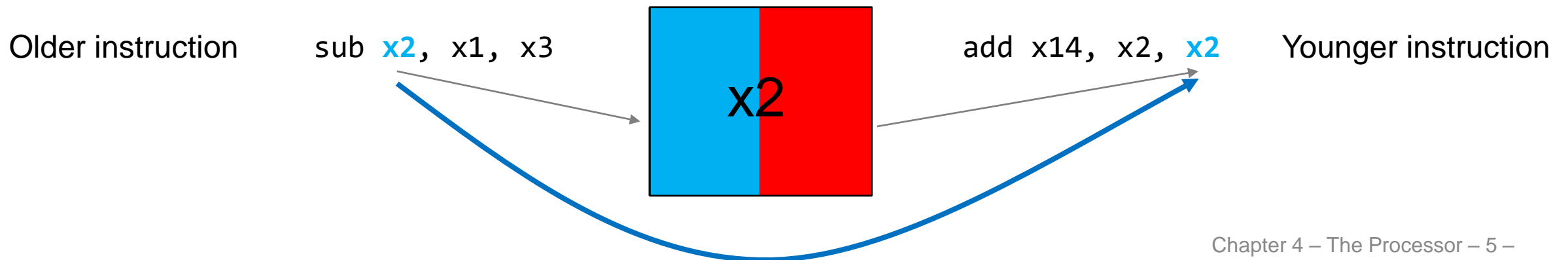
- ❖ Example) `addi x1, x1, 10`

- What you read from x1 is the old value
- As we proceed to the next cycle, x1 is updated with $x1 + 10$

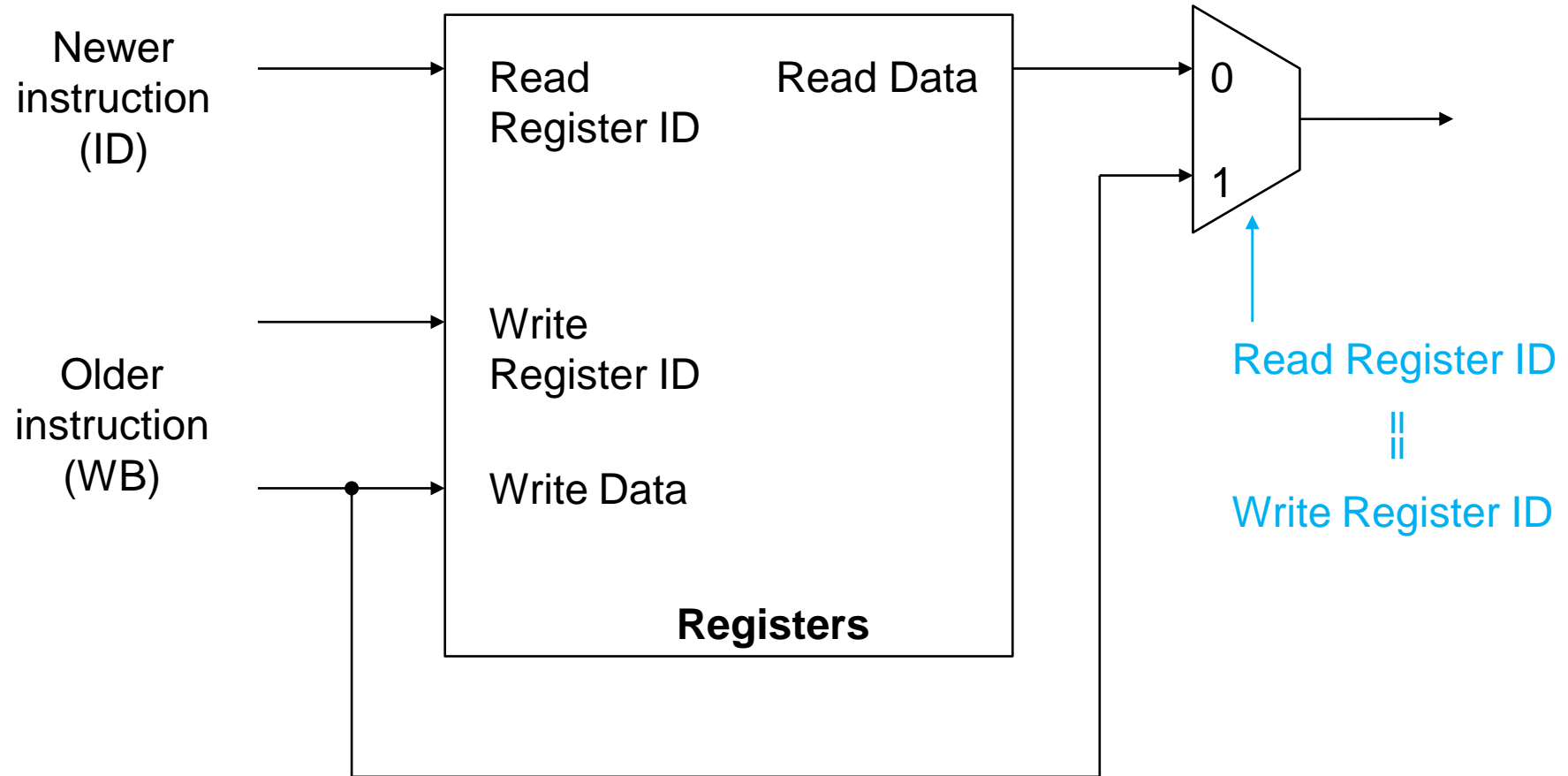


Same Register Read & Write

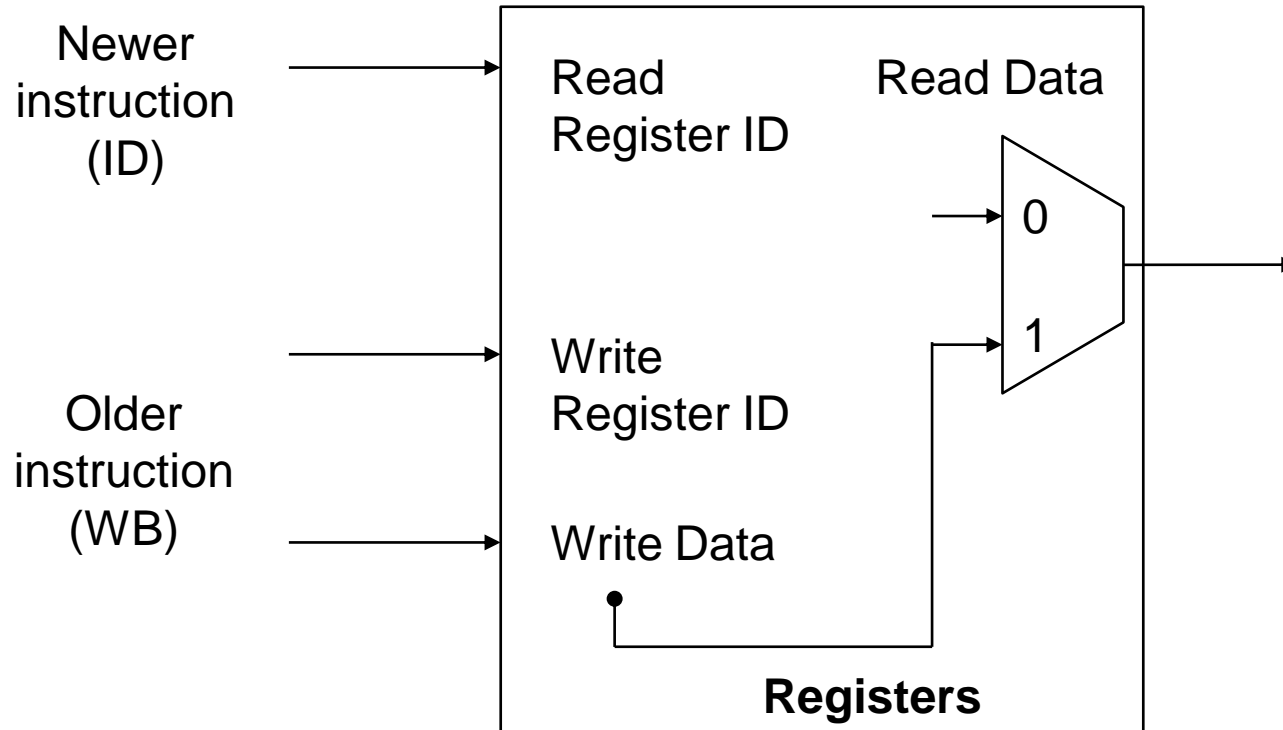
- Pipelined CPU: Let the CPU to read the NEW value written in the current cycle
 - ❖ An instruction does not read and write its source/destination registers at the same time
 - Read: ID stage, Write: WB stage
 - ❖ From the register file's perspective,
 - Read: Younger instruction
 - Write: Older instruction
 - ❖ → **Bypass** the older instruction's value to the younger instruction



Register Bypass

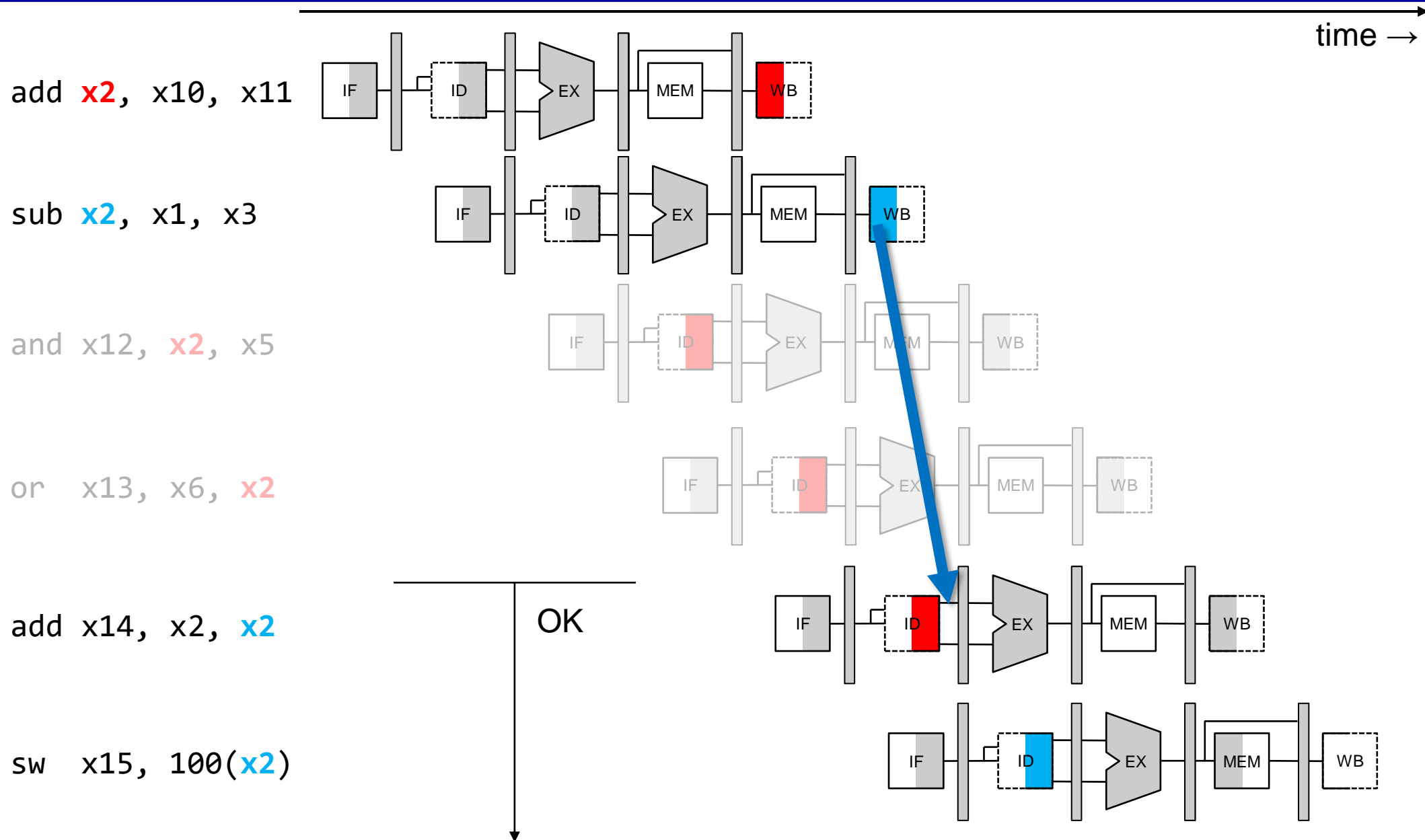


Register Bypass

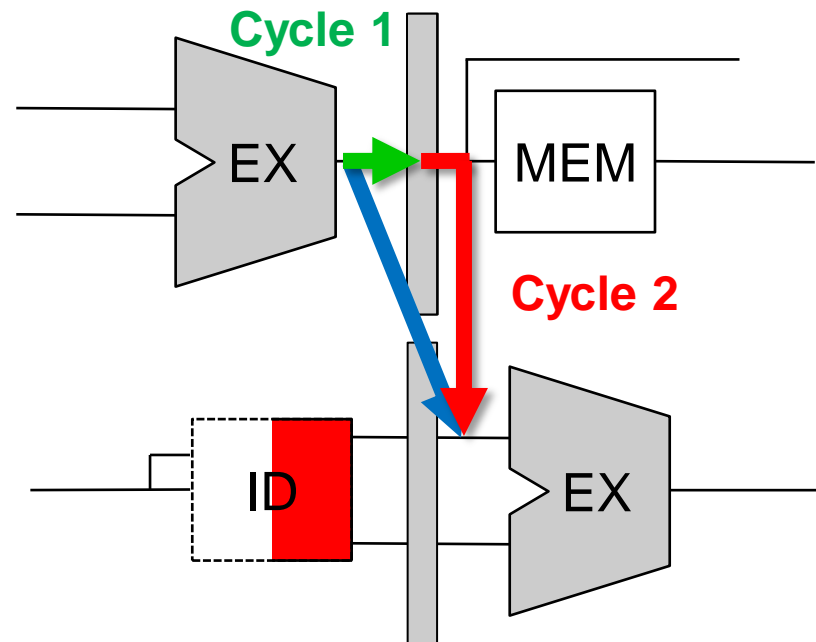
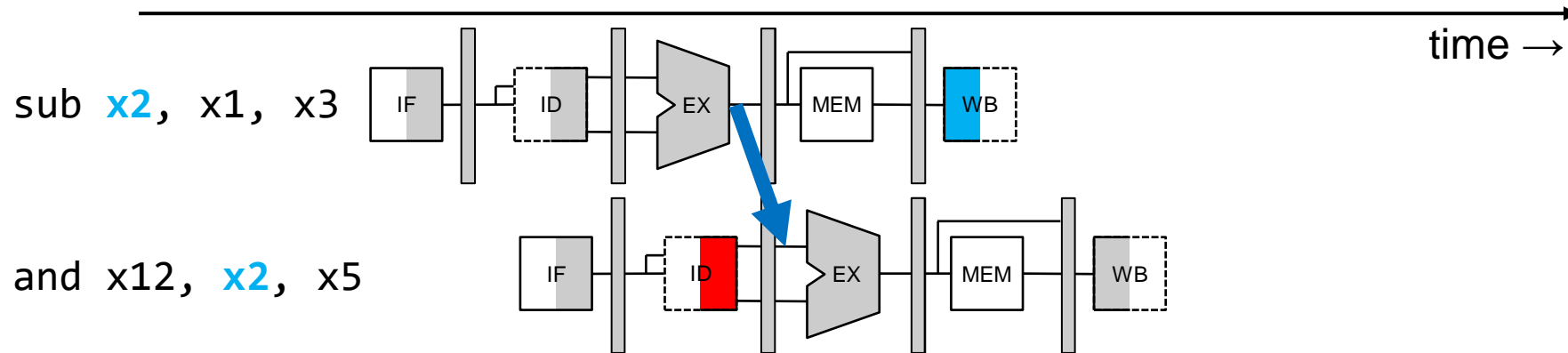


- From now on, we assume the bypass circuit is embedded in the register file

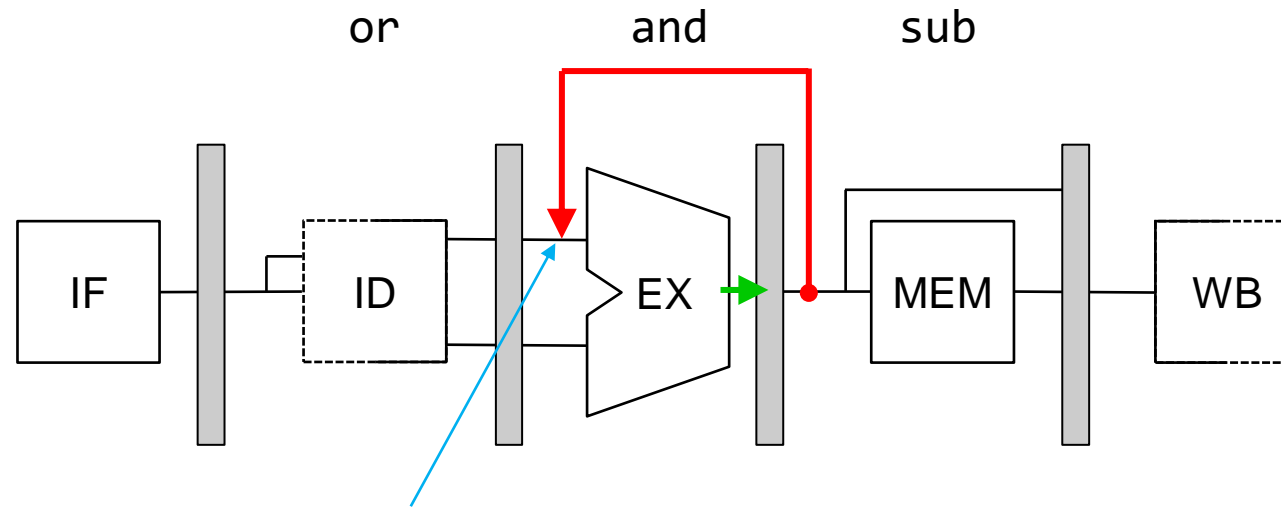
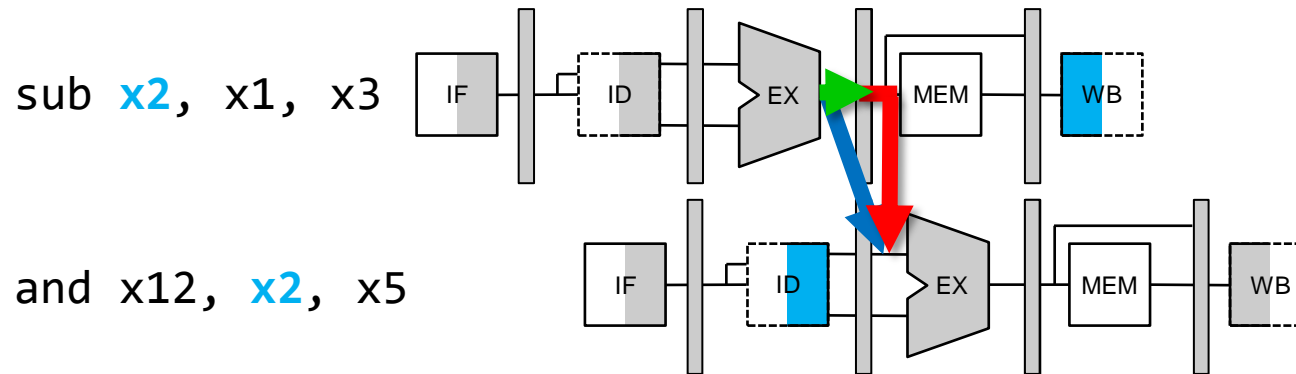
Dependency Resolving using Register Bypass



Register Forwarding

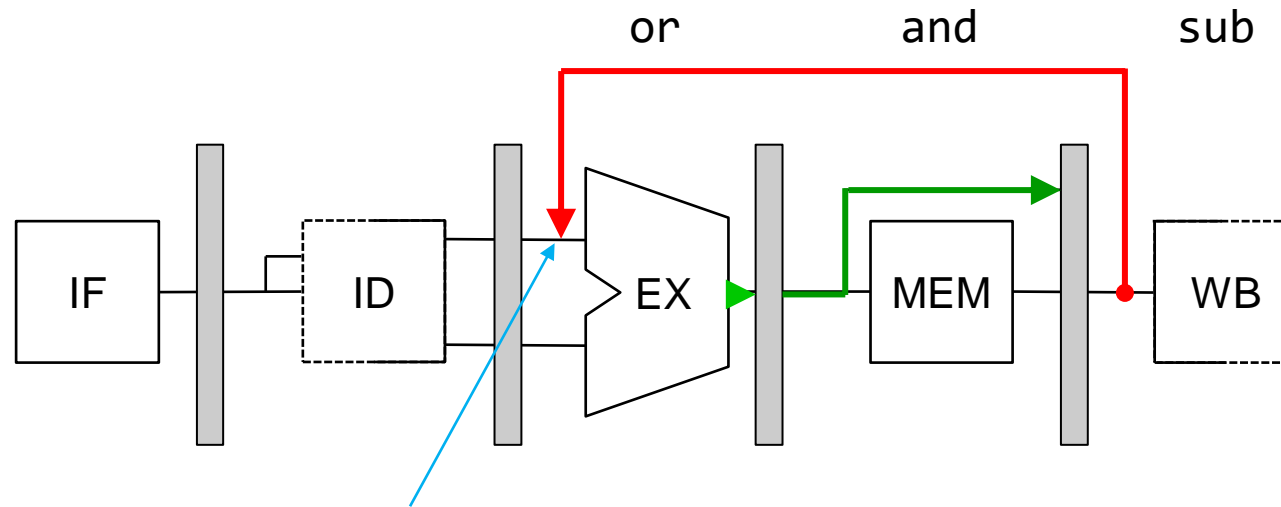
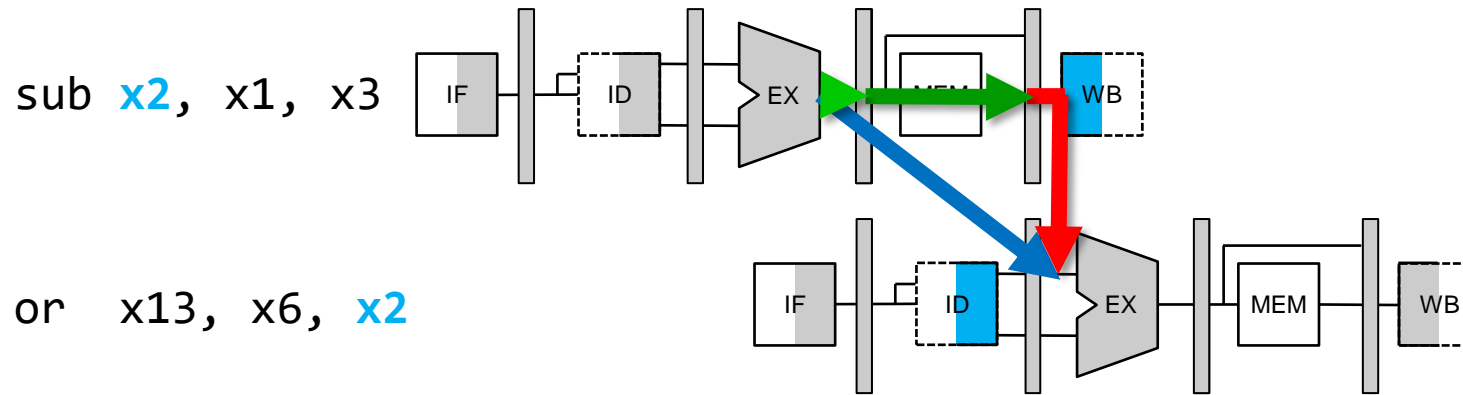


Register Forwarding



Need to decide when to use forwarding

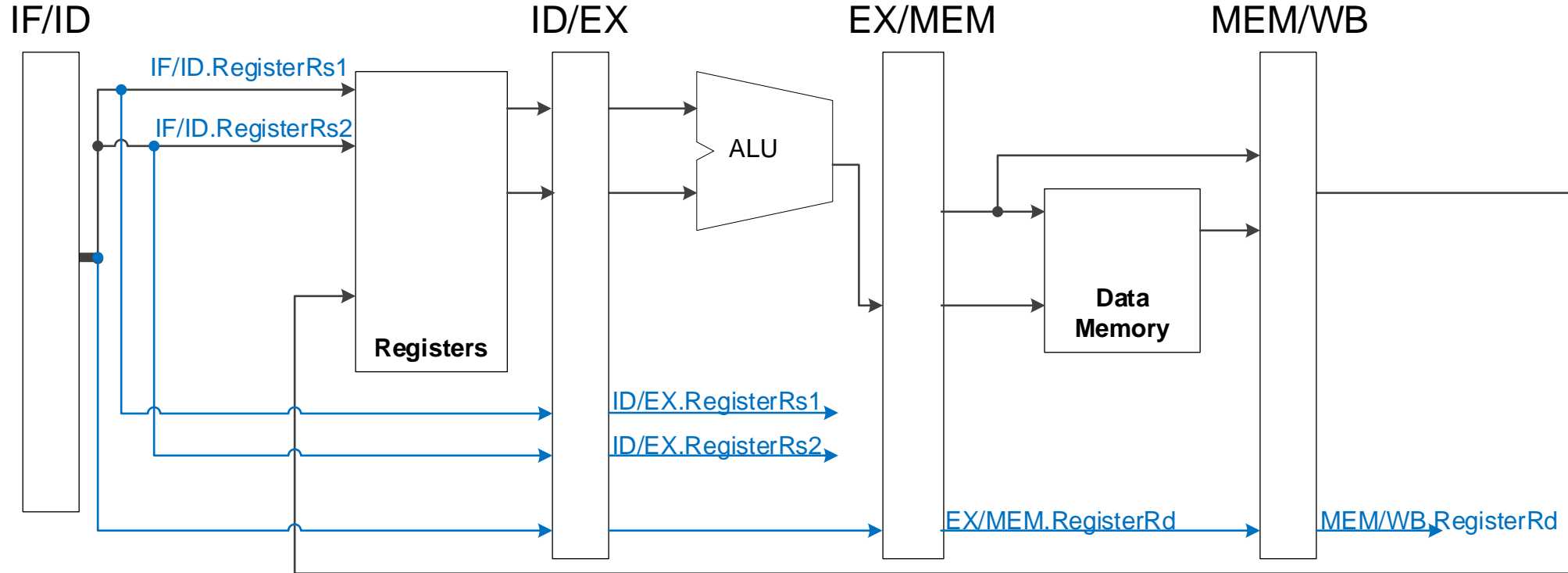
Register Forwarding



Need to decide when to use forwarding

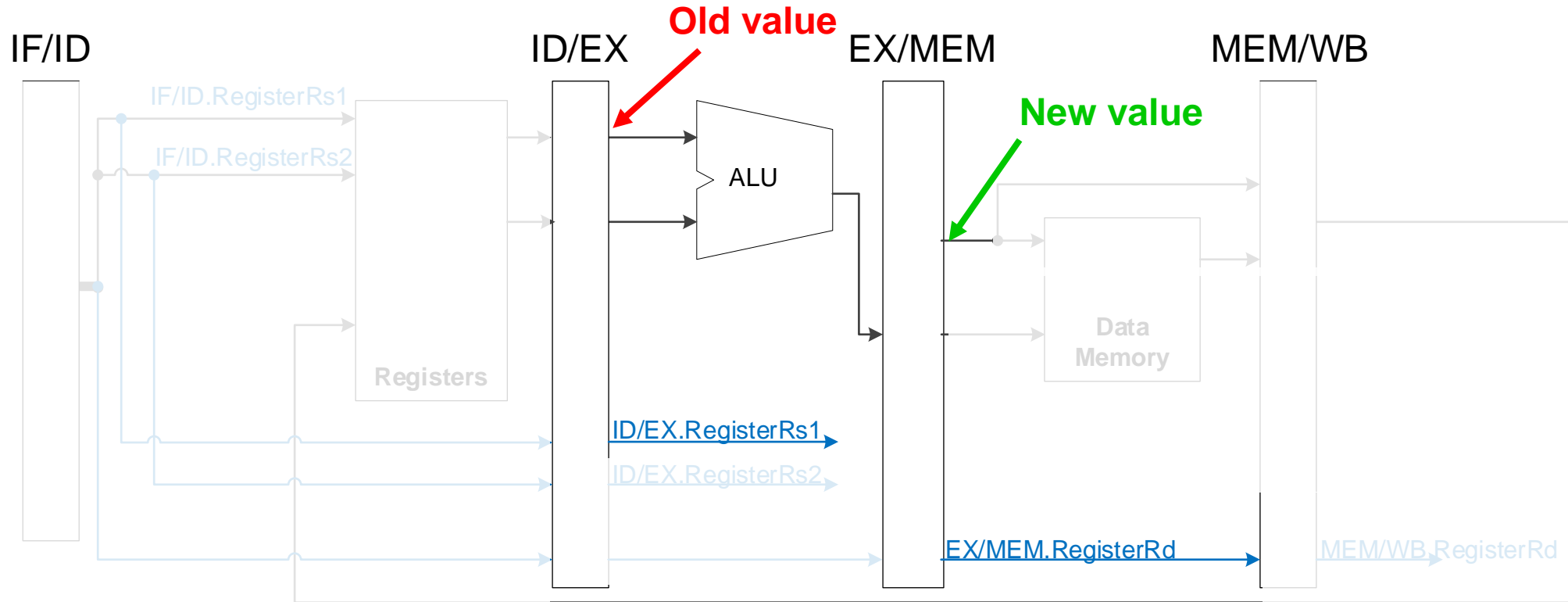
Detecting the Need for Forwarding

- Pass register numbers along the pipeline



Detecting the Need for Forwarding

- e.g., ID/EX.RegisterRs1 = register number for rs1 in ID/EX



Register Forwarding Condition

- Data hazards when

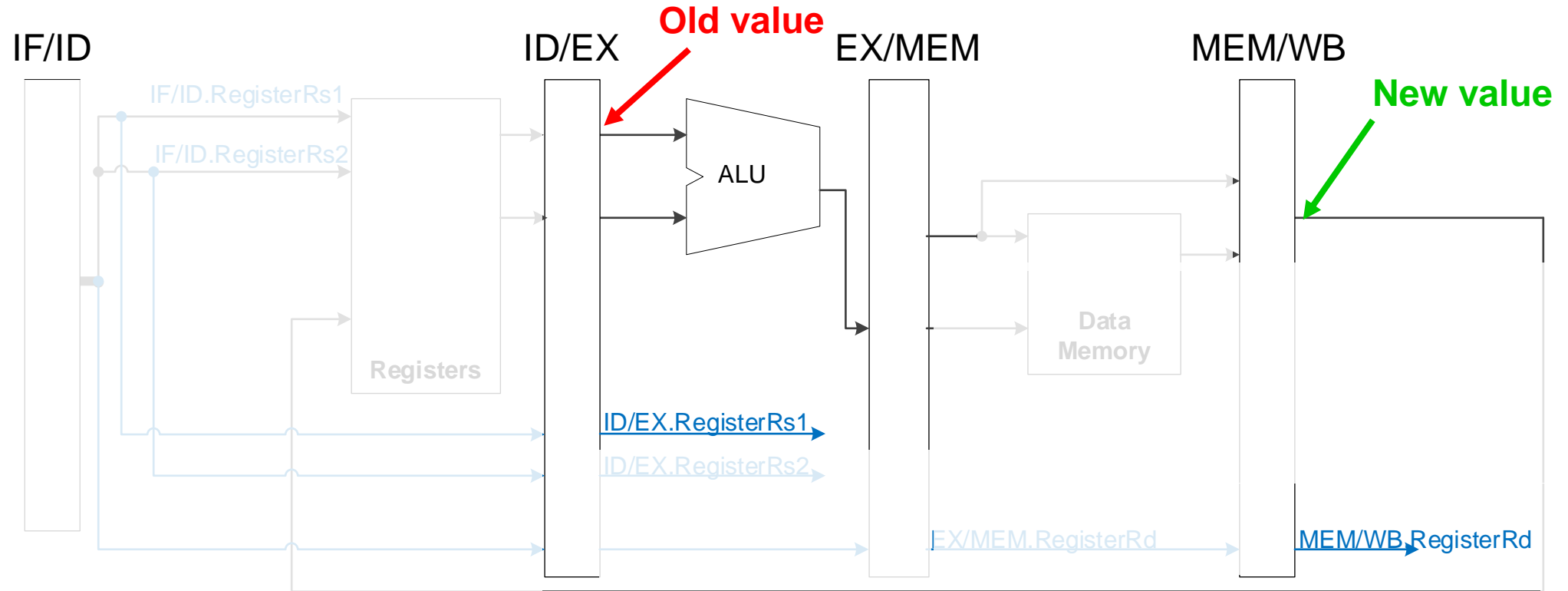
- ❖ 1a. **EX/MEM.RegisterRd = ID/EX.RegisterRs1**

- ❖ 1b. **EX/MEM.RegisterRd = ID/EX.RegisterRs2**

→ *“EX Hazard”*

Forward from
EX/MEM
pipeline reg

Detecting the Need for Forwarding



Register Forwarding Condition

■ Data hazards when

❖ 1a. **EX/MEM.RegisterRd = ID/EX.RegisterRs1**

❖ 1b. **EX/MEM.RegisterRd = ID/EX.RegisterRs2**

→ *“EX Hazard”*

❖ 2a. **MEM/WB.RegisterRd = ID/EX.RegisterRs1**

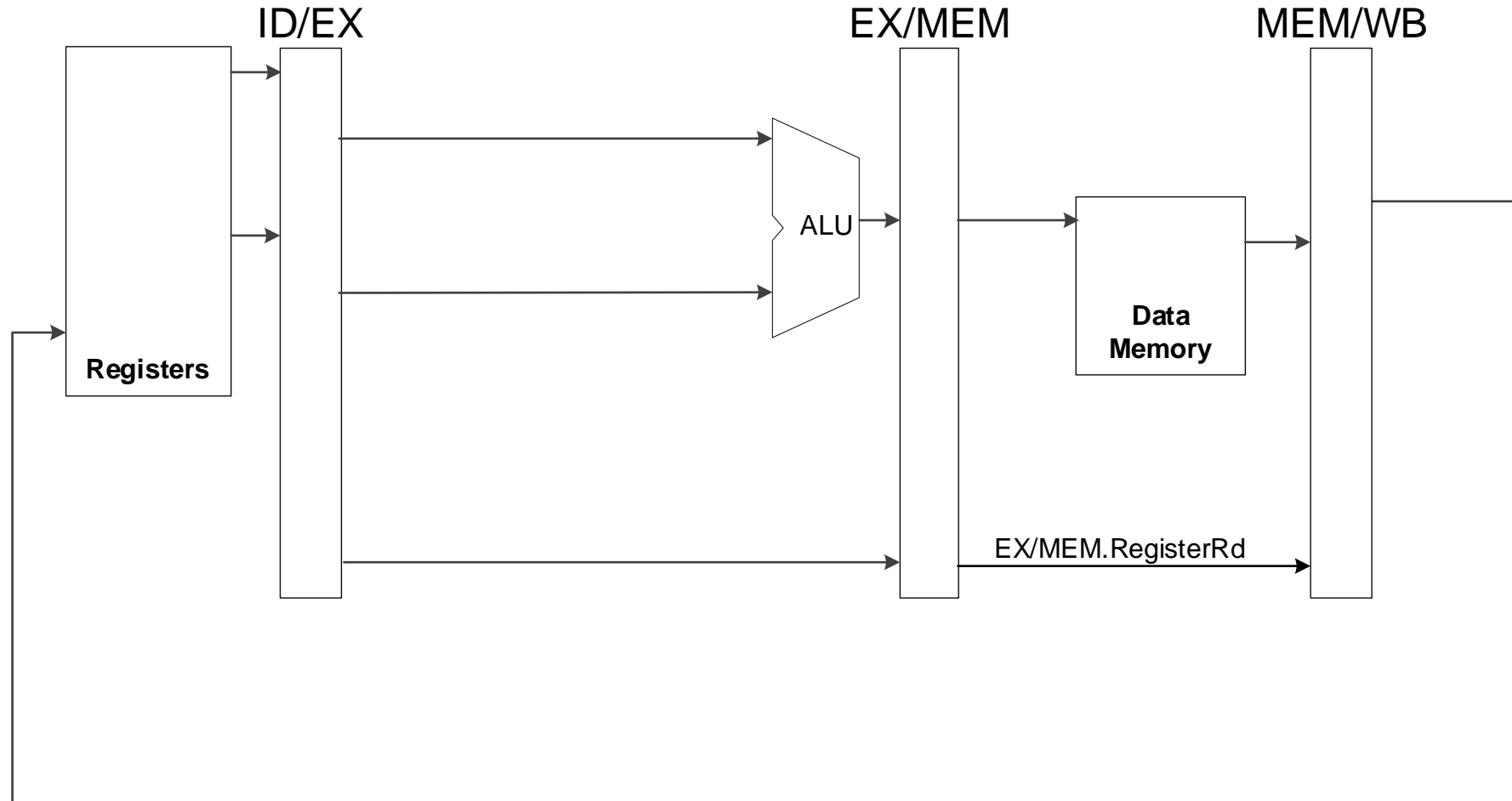
❖ 2b. **MEM/WB.RegisterRd = ID/EX.RegisterRs2**

→ *“MEM Hazard”*

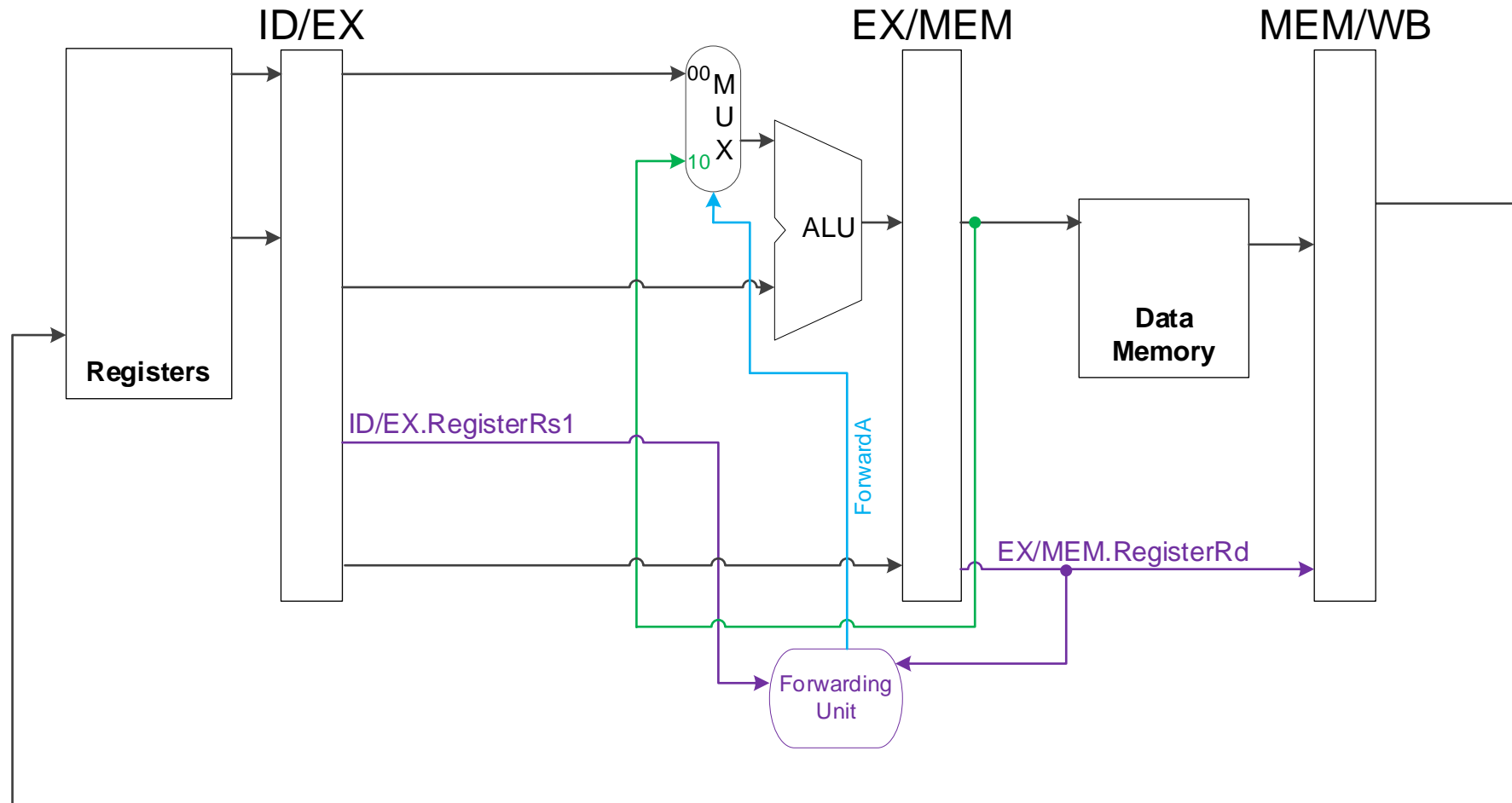
Forward from
EX/MEM
pipeline reg

Forward from
MEM/WB
pipeline reg

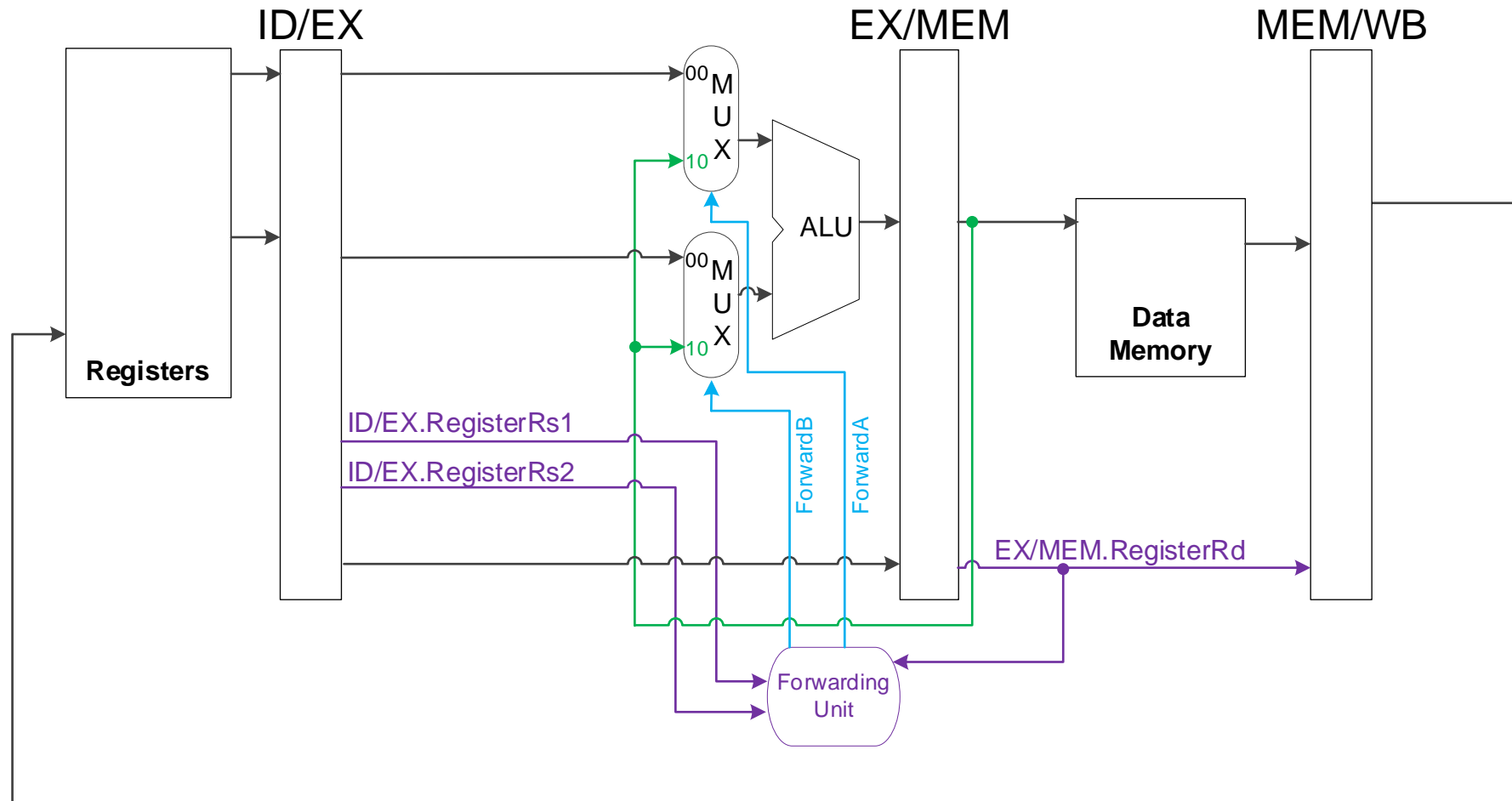
Forwarding Paths – Original Datapath



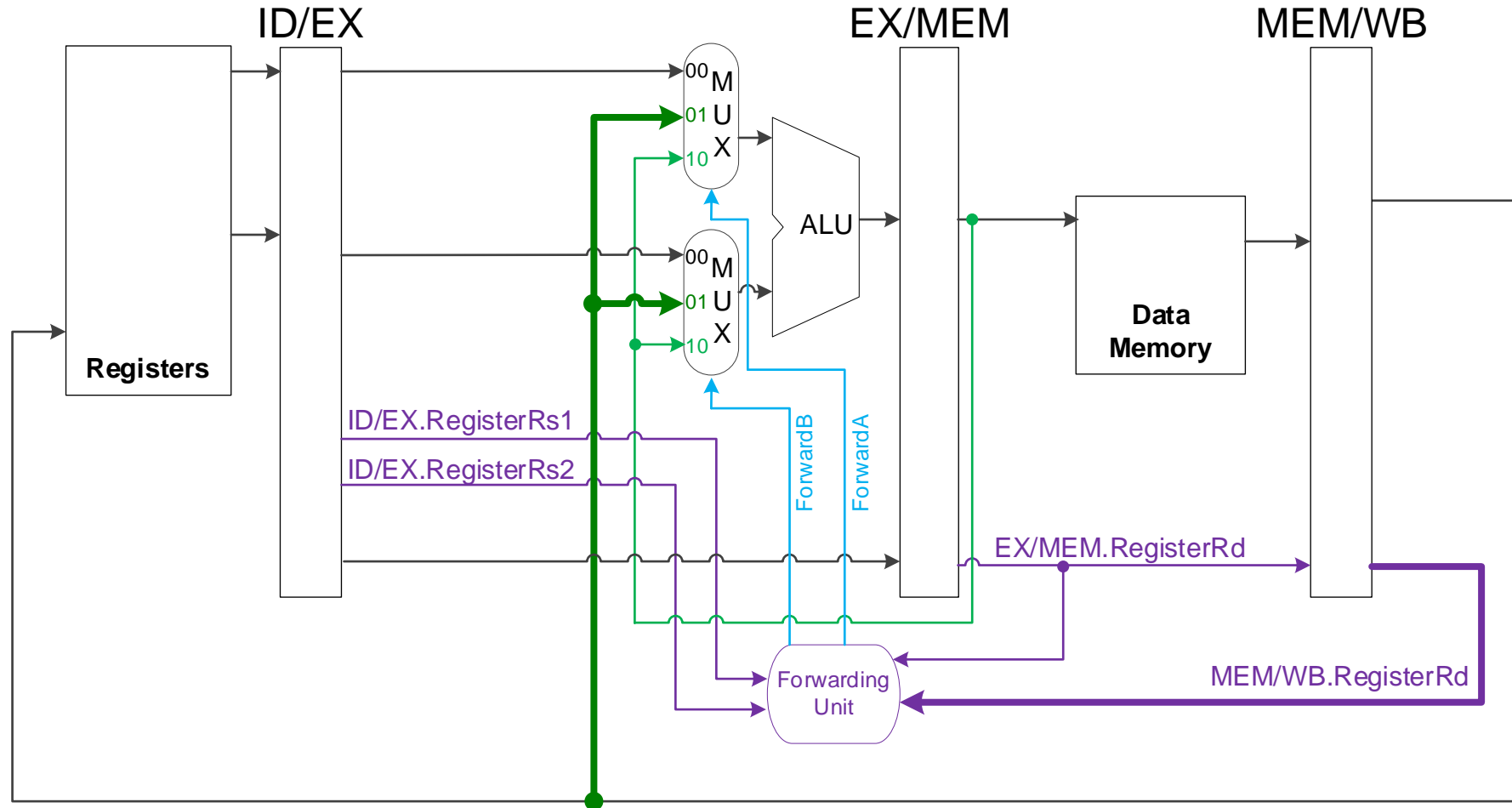
Forwarding Paths – Original Datapath



Forwarding Paths – EX Hazard Only



Forwarding Paths – EX & MEM Hazards



Forwarding Conditions – Rs1

- Default (No forwarding)
 - ❖ `ForwardA = 00`
- EX hazard
 - ❖ if (EX/MEM.RegisterRd = ID/EX.RegisterRs1)
`ForwardA = 10`
- MEM hazard
 - ❖ if (MEM/WB.RegisterRd = ID/EX.RegisterRs1)
`ForwardA = 01`

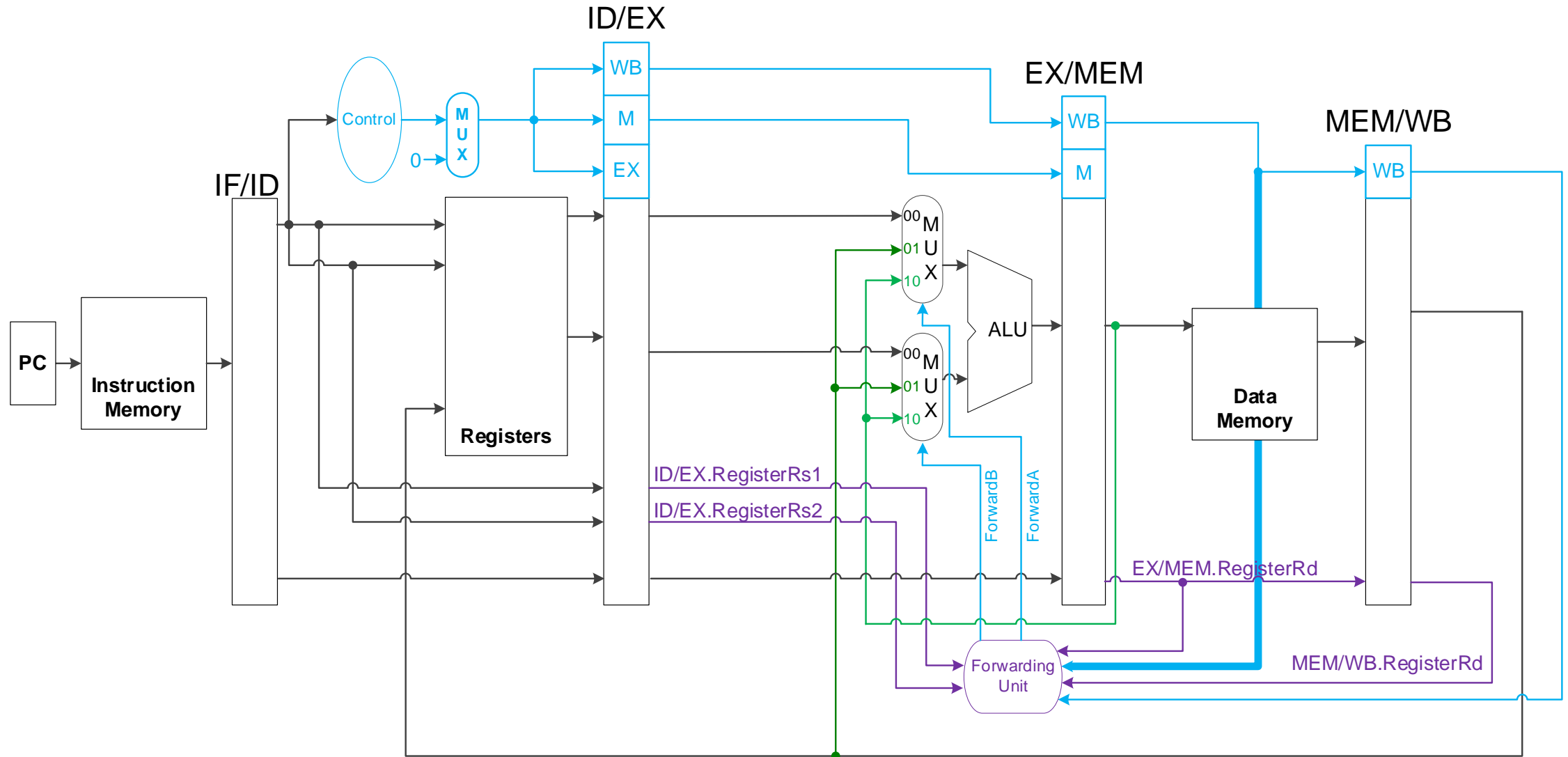
Detecting the Need to Forward

- But only if there is an **actual register write** !
 - ❖ EX/MEM.RegWrite, MEM/WB.RegWrite
- ... and only if **Rd is not zero** !
 - ❖ EX/MEM.RegisterRd $\neq 0$,
MEM/WB.RegisterRd $\neq 0$

Forwarding Conditions – Rs1

- Default (No forwarding)
 - ❖ `ForwardA = 00`
- EX hazard
 - ❖ `if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs1))`
`ForwardA = 10`
- MEM hazard
 - ❖ `if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs1))`
`ForwardA = 01`


Datapath with Forwarding




Double Data Hazard

- Consider the following sequences:

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



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



- Both hazards occur
 - ❖ Want to use the most recent one
- Revise MEM hazard condition
 - ❖ Forward MEM hazard only when there is no EX hazard.

Revised Forwarding Condition

■ EX hazard

- ❖ if (EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs1))

ForwardA = 10

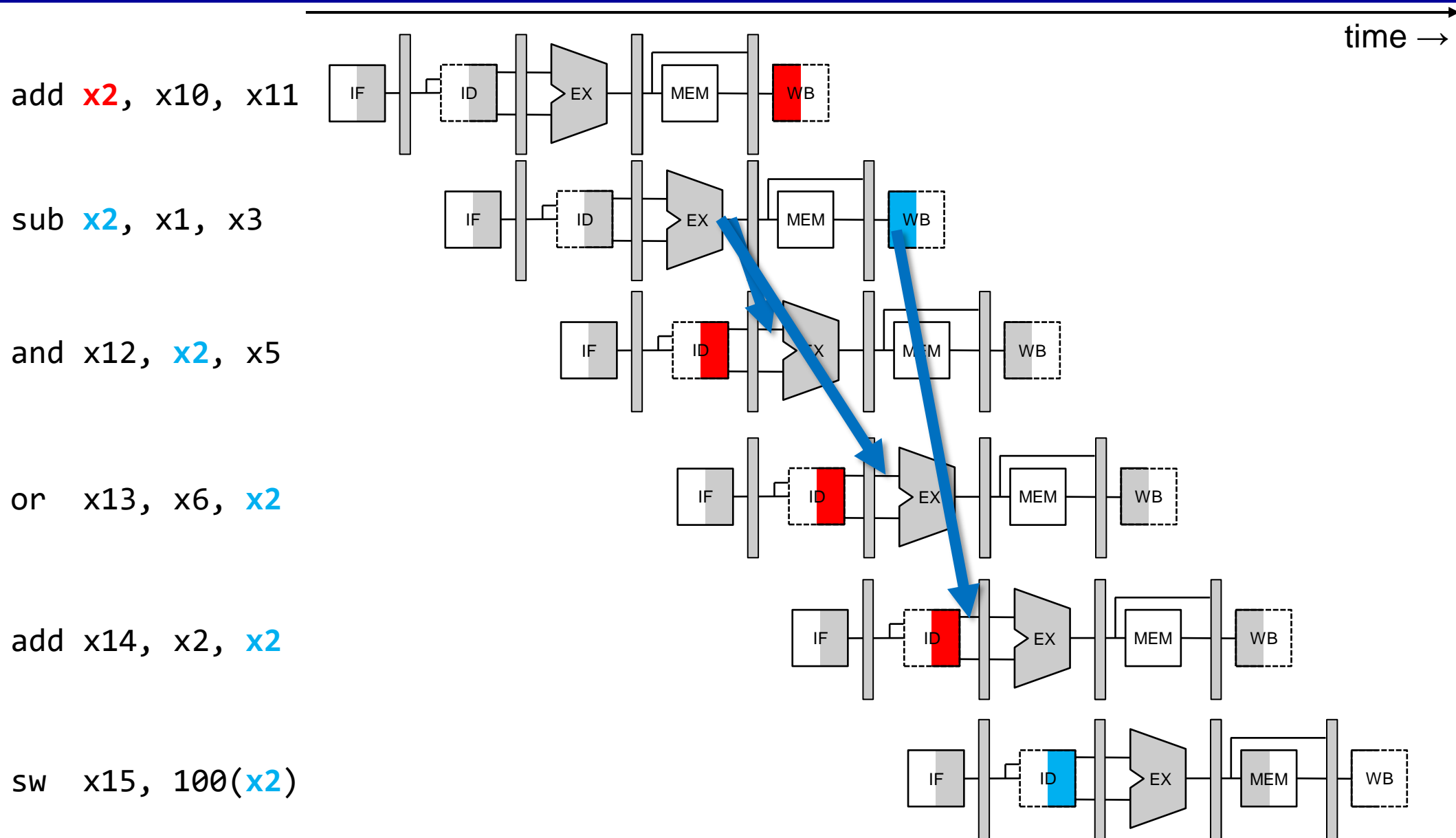
■ MEM hazard

- ❖ if (MEM/WB.RegWrite and (MEM/WB.RegisterRd \neq 0)
and no EX hazard
and (MEM/WB.RegisterRd = ID/EX.RegisterRs1))

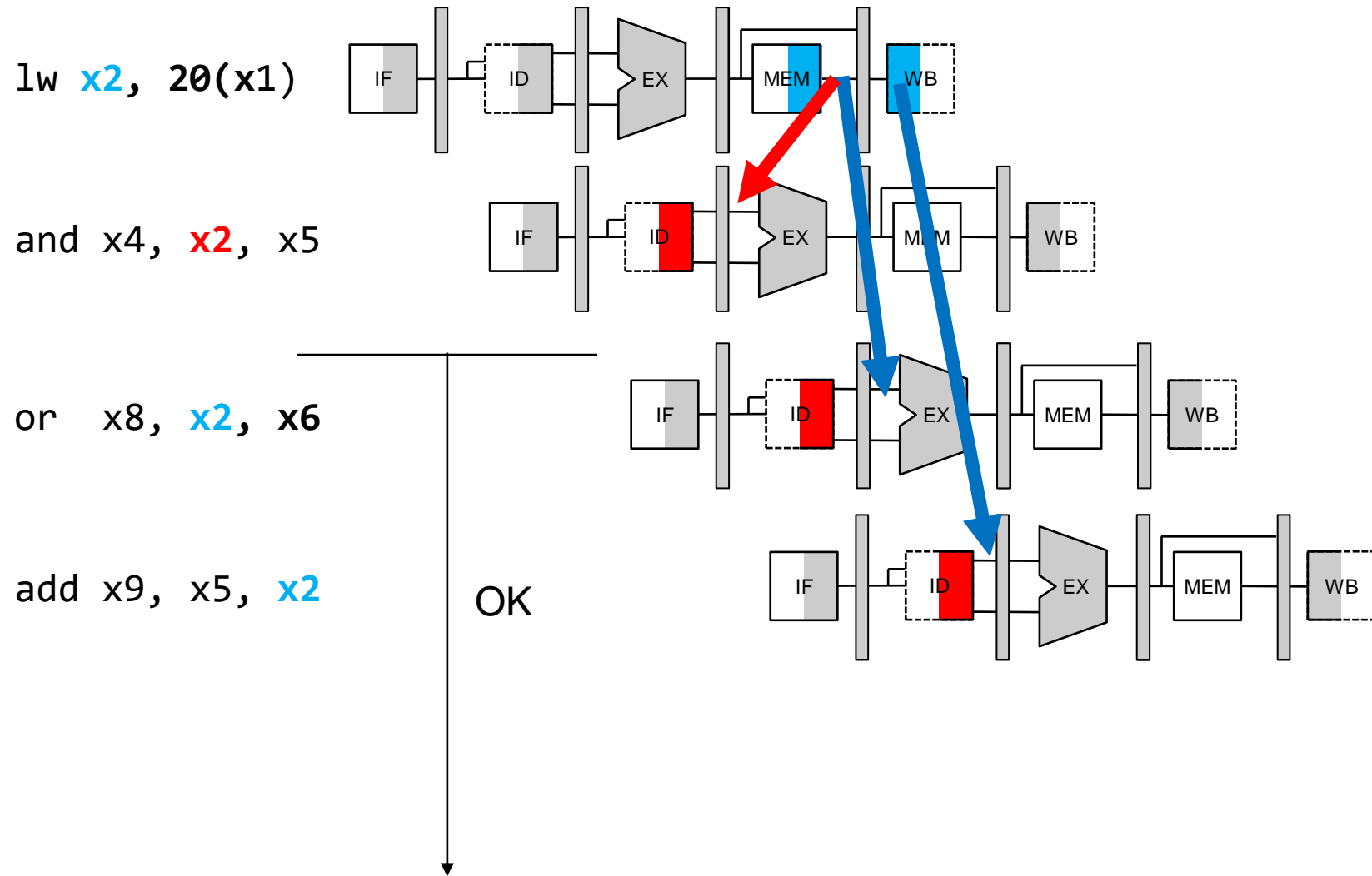
ForwardA = 01

Same for Rs2...

Data Hazard Between R-Type Arithmetic Instructions

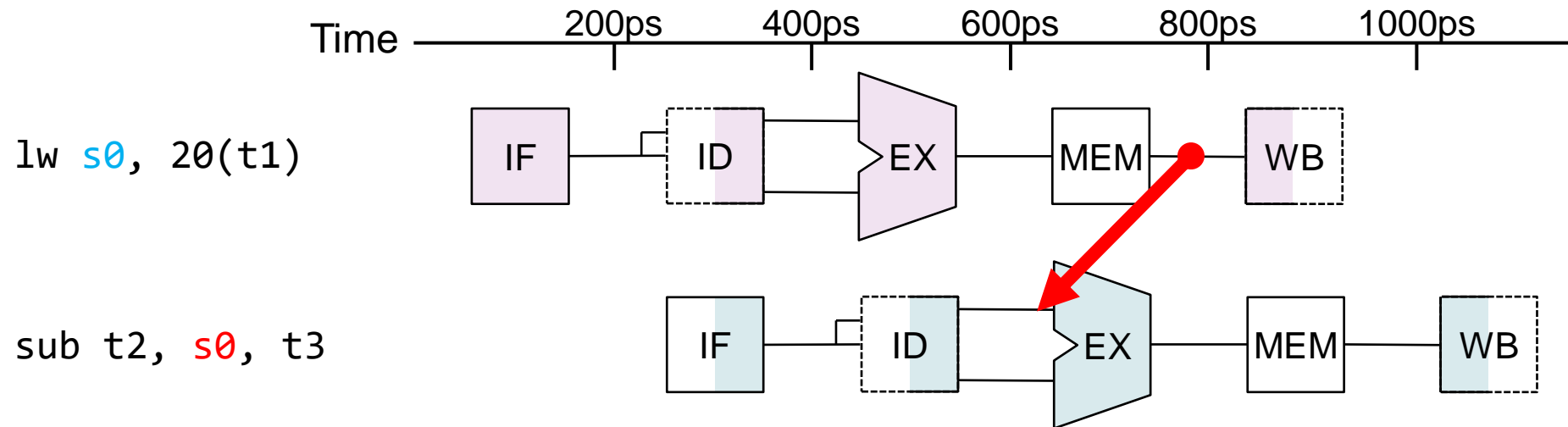


Load-Use Data Hazard

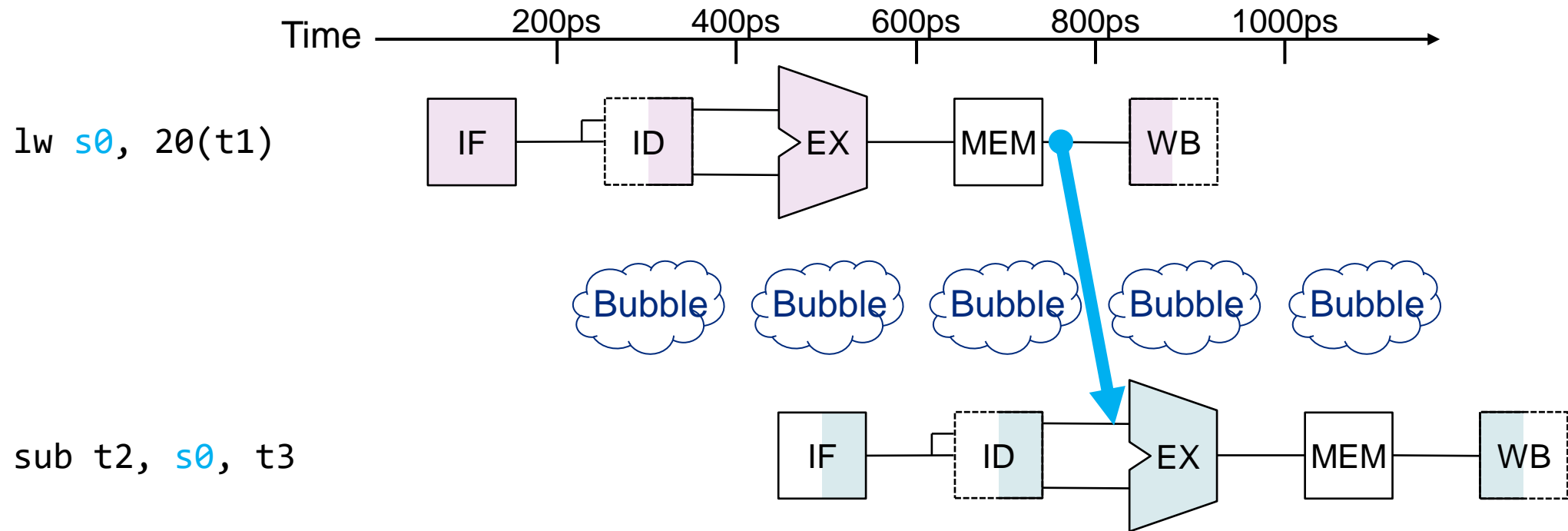


Load-Use Data Hazard

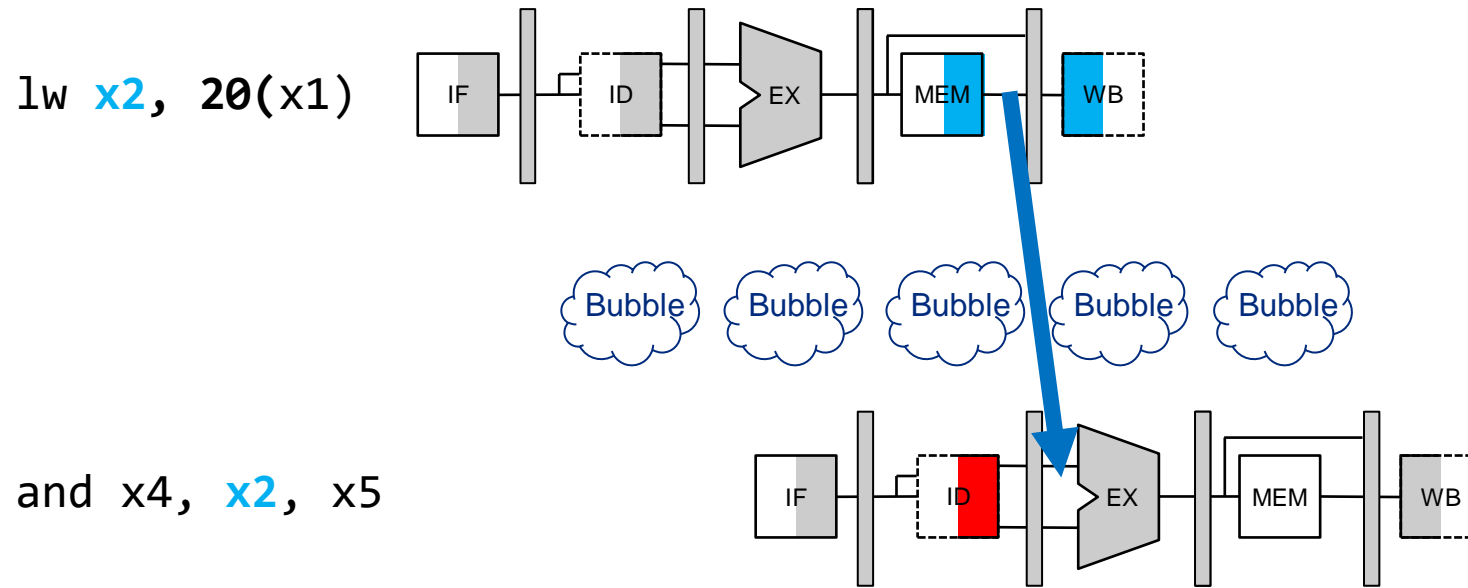
- Can't always avoid stalls by forwarding
 - ❖ If value not computed when needed
 - ❖ Can't forward backward in time!



Load-Use Data Hazard



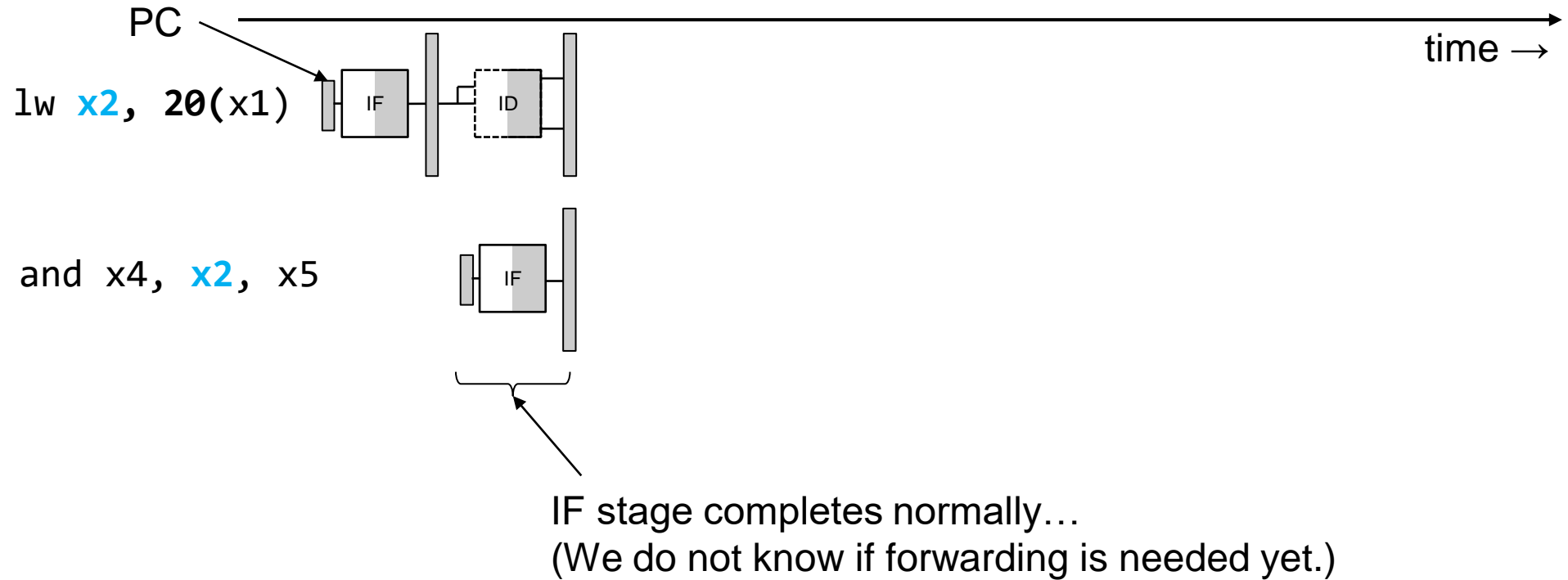
Load-Use Data Hazard



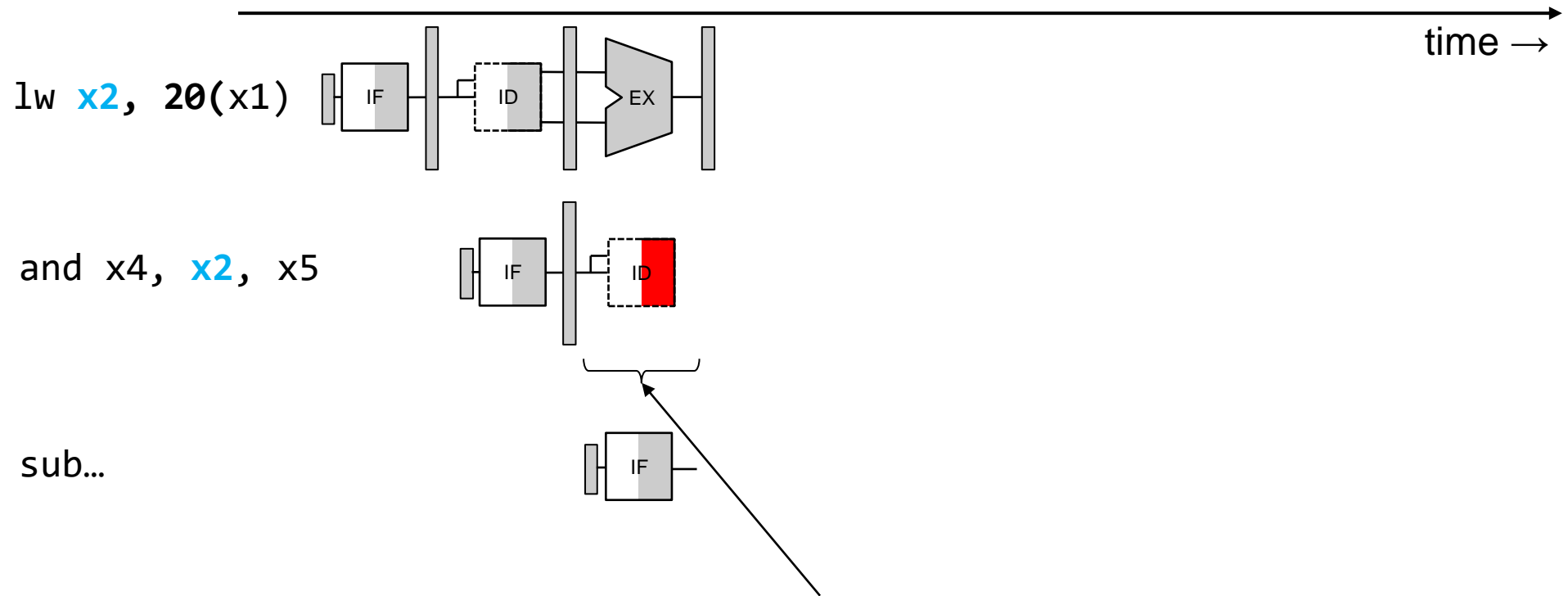
Need to stall 1 cycle

Let's see how it exactly works...

Stall for Load-Use Data Hazard (1)



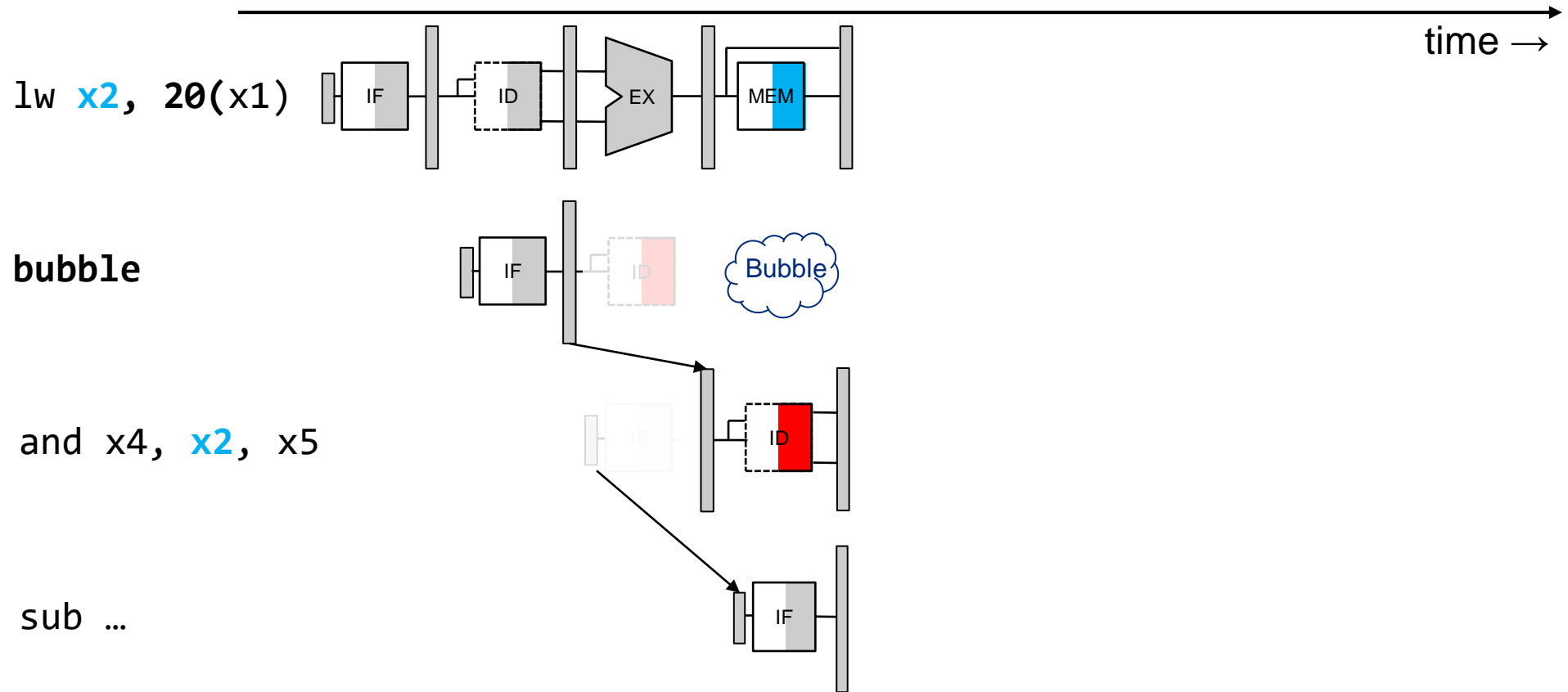
Stall for Load-Use Data Hazard (2)



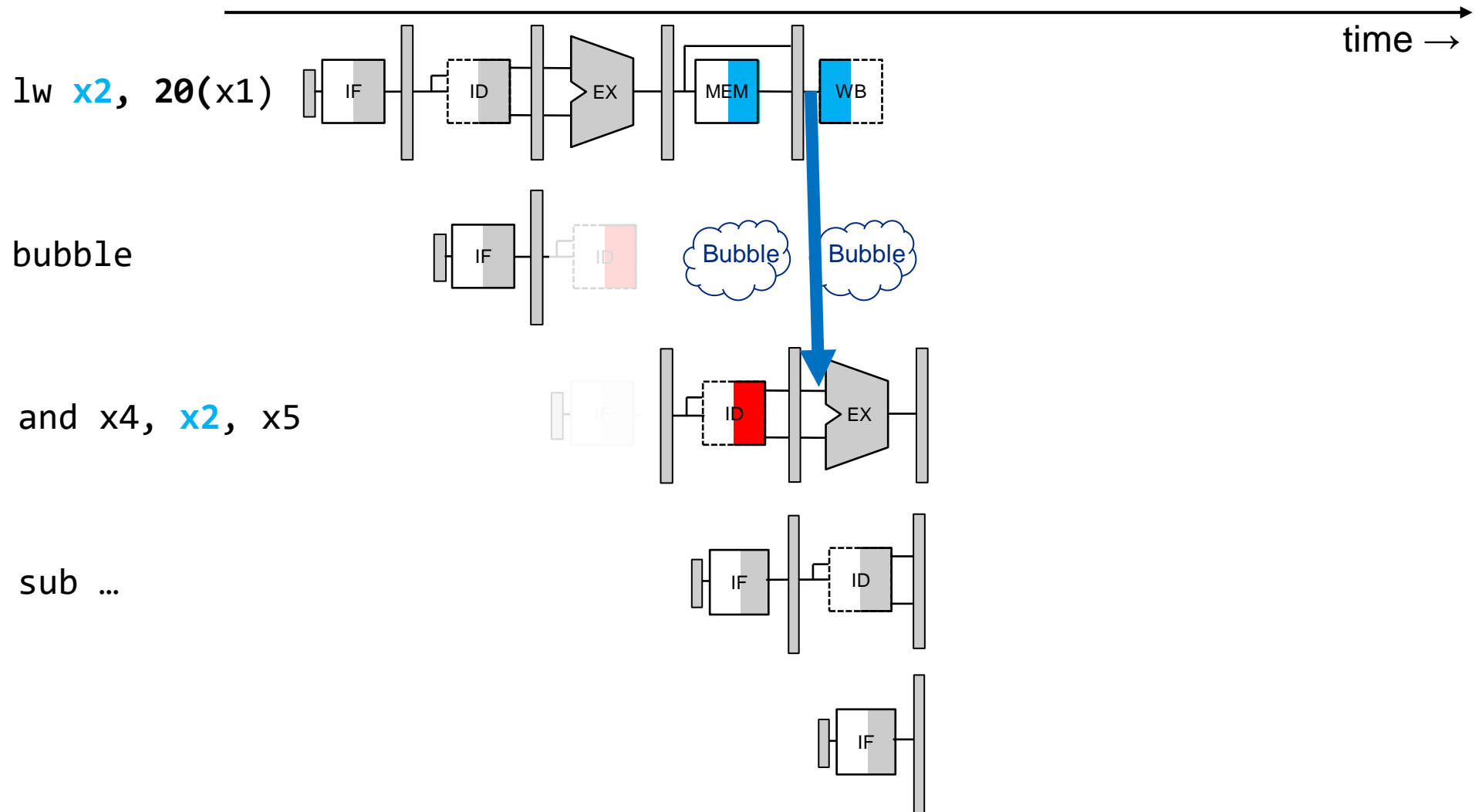
At the ID stage, we can identify if forwarding is required.

This instruction need to stay at the ID stage for 1 more cycle

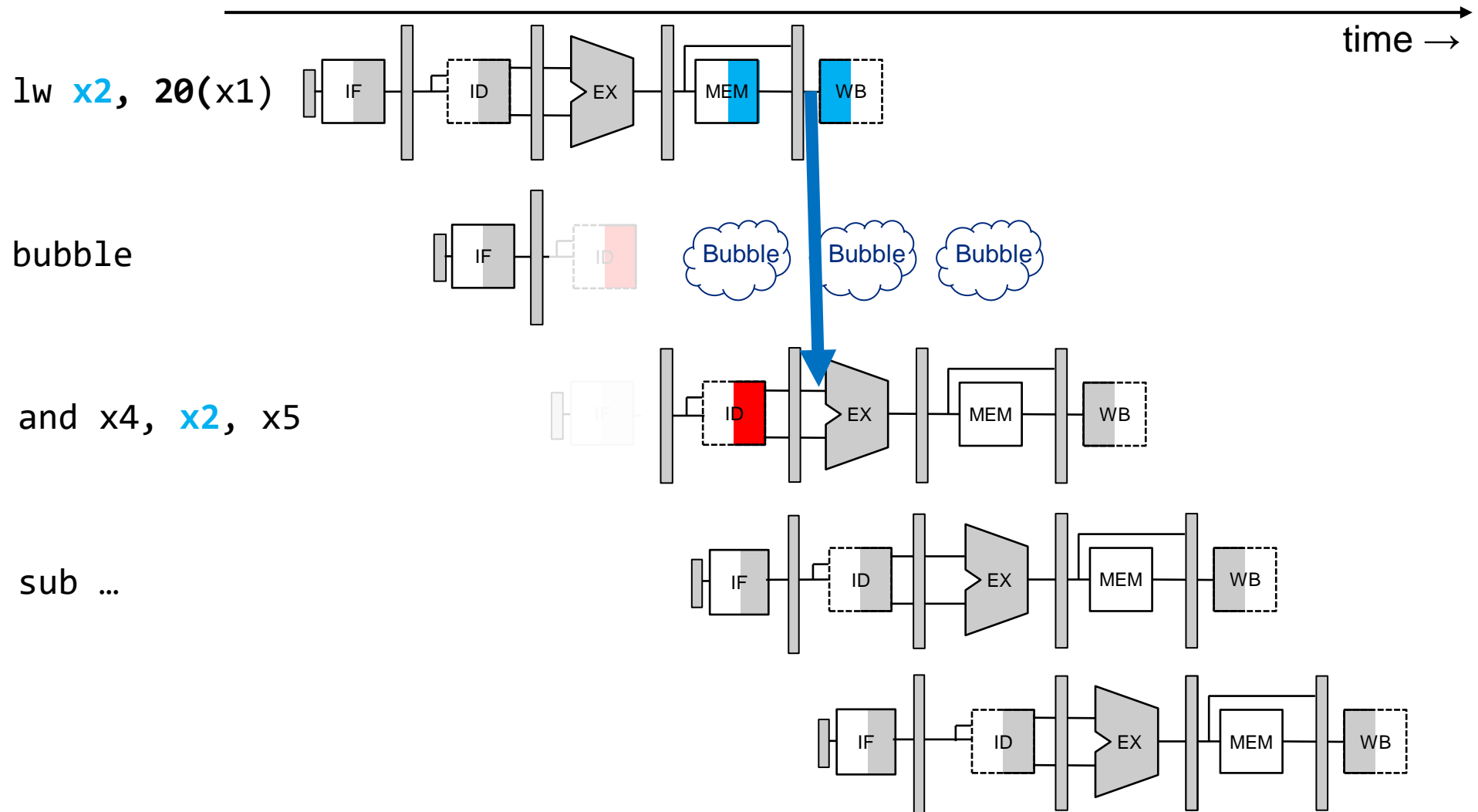
Stall for Load-Use Data Hazard (3)



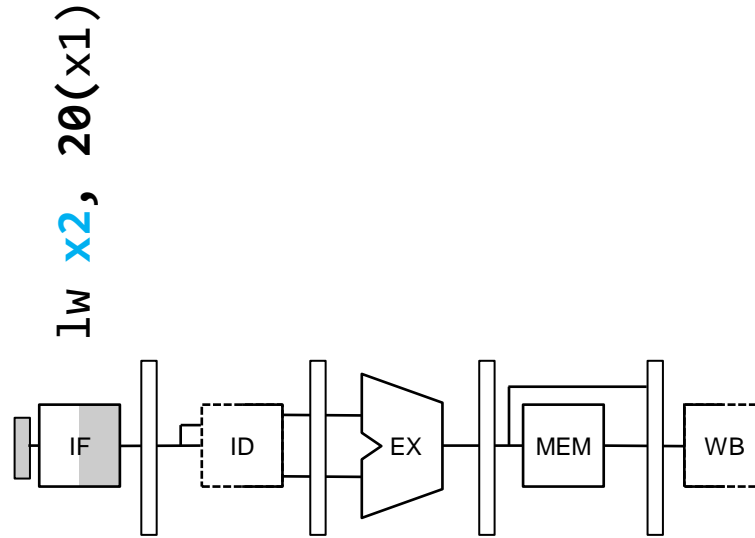
Stall for Load-Use Data Hazard (4)



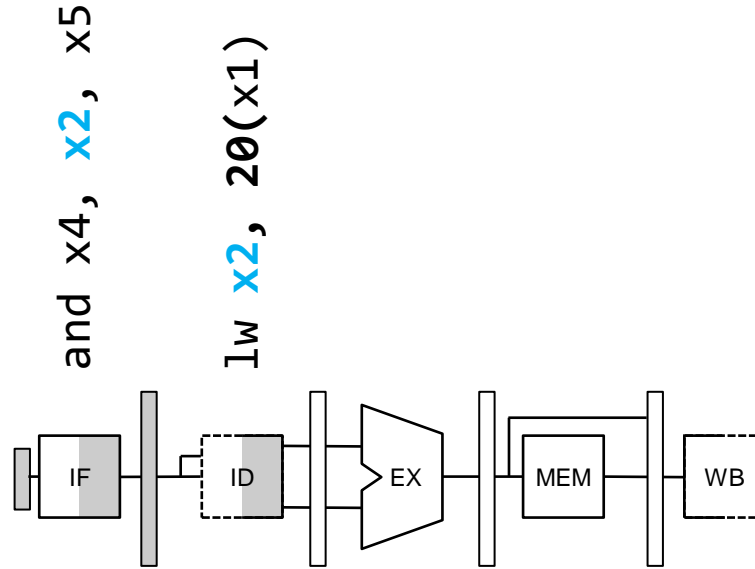
Stall for Load-Use Data Hazard (5)



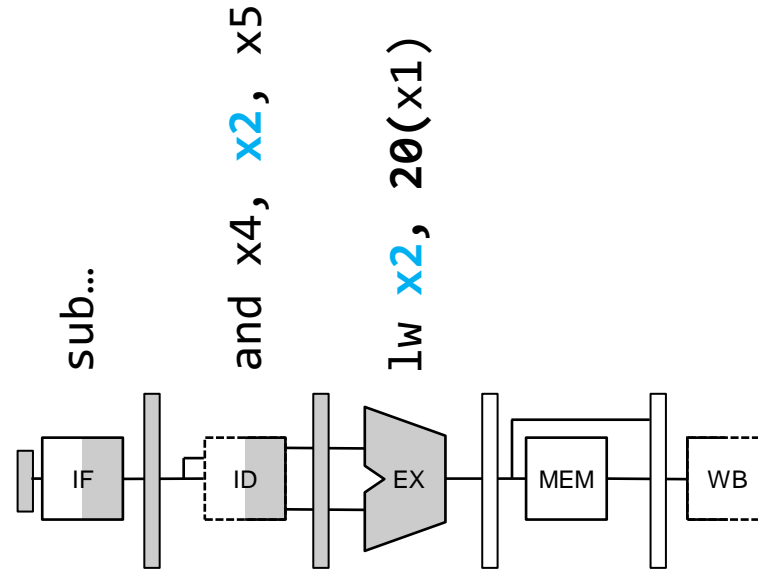
Stall for Load-Use Data Hazard – Different View: Cycle 1



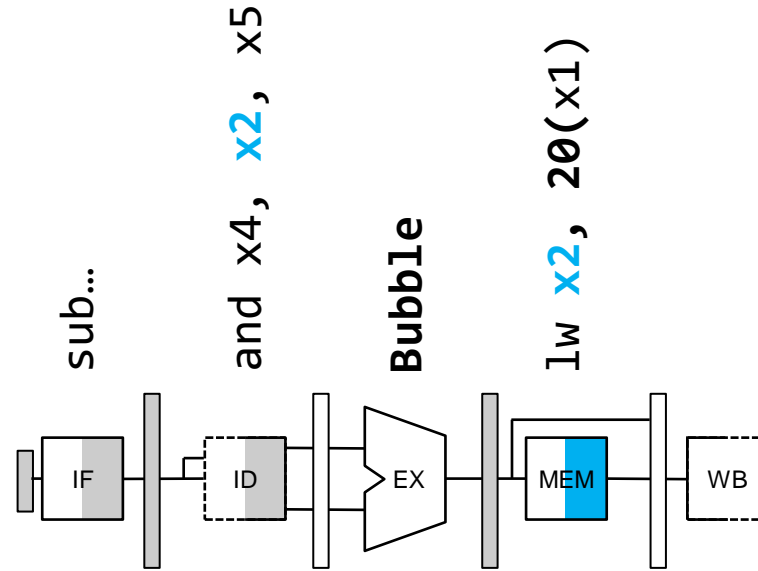
Stall for Load-Use Data Hazard – Different View: Cycle 2



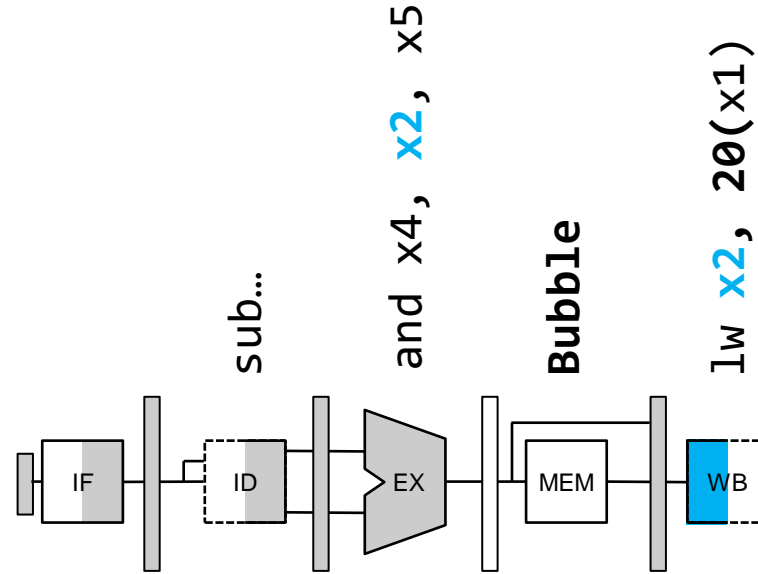
Stall for Load-Use Data Hazard – Different View: Cycle 3



Stall for Load-Use Data Hazard – Different View: Cycle 4



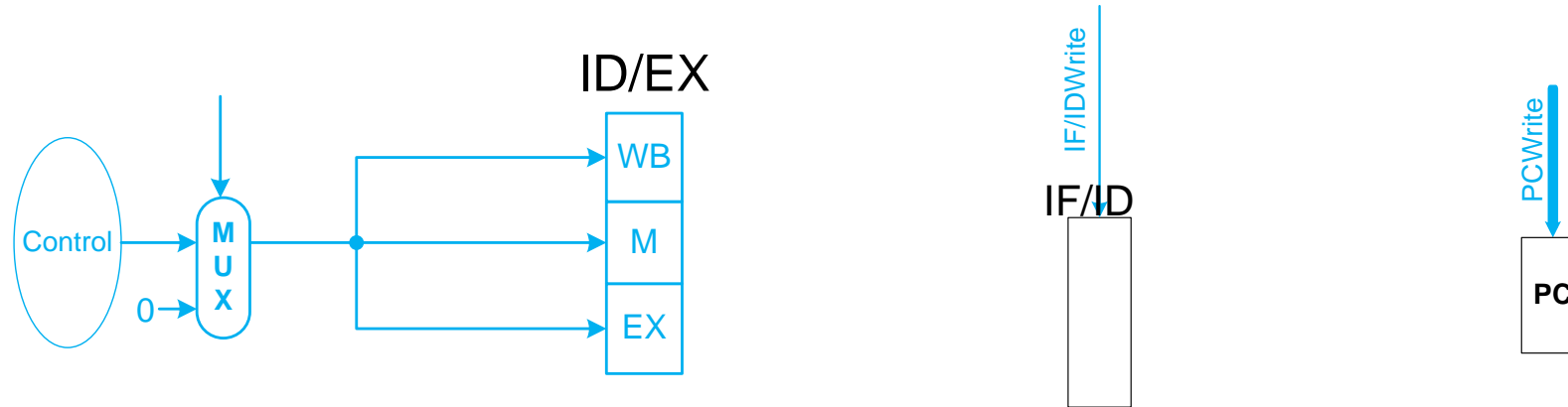
Stall for Load-Use Data Hazard – Different View: Cycle 5



Adding a Bubble – What We Want

1. Pipeline Bubble in EX, MEM, WB stages: Skip
2. Repeat the ID stage 1 more cycle
3. Repeat the IF stage 1 more cycle

Adding a Bubble – How We Do It

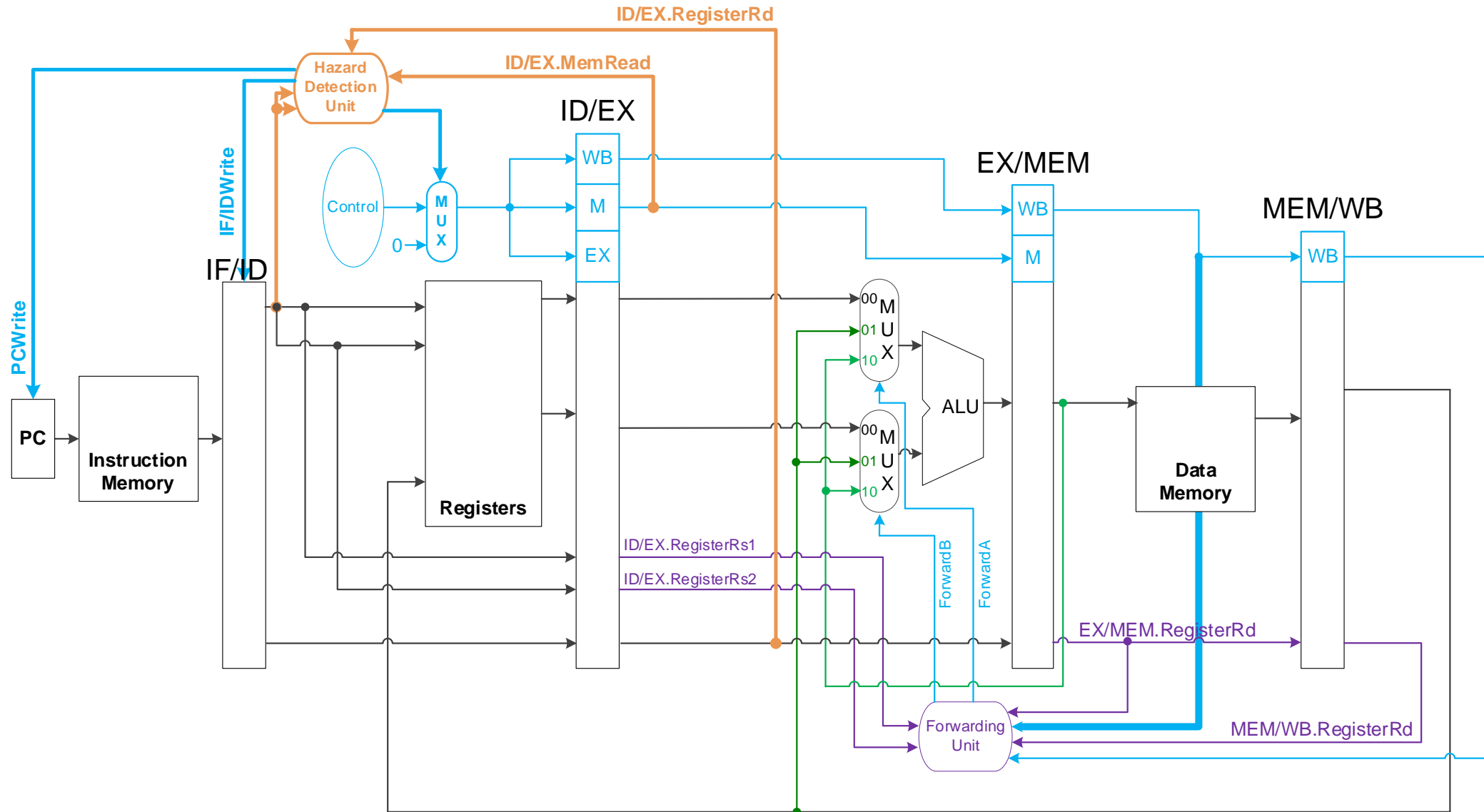


1. EX, MEM, WB stages: Skip
 - All control signals ([RegWrite](#), [MemWrite](#), [MemRead](#),...) : '0'
2. Repeat the ID stage 1 more cycle
 - Do not change the IF/ID pipeline register → [IF/IDWrite](#) signal: '0'
3. Repeat the IF stage 1 more cycle
 - Do not change the PC → [PCWrite](#) signal: '0'

Load-Use Hazard Detection

- Check in ID stage
 - ❖ ALU operand register numbers in ID stage are...
 - `IF/ID.RegisterRs1`, `IF/ID.RegisterRs2`
 - ❖ Destination register for load instruction is
 - `ID/EX.RegisterRd`
- Load-use hazard when
 - ❖ `ID/EX.MemRead` and
 - `((ID/EX.RegisterRd = IF/ID.RegisterRs1) or`
`(ID/EX.RegisterRd = IF/ID.RegisterRs2))`
- If detected, stall and insert bubble

Datapath with Hazard Detection



Code Scheduling to Avoid Stalls

- Reorder code to avoid use of load result in the next instruction
- $A = B + E$; $C = B + F$;

