Topic 7

Data Hazards

Hazards

- Situations that prevent starting the next instruction in the next cycle
- Data hazard
  - Need to wait for previous instruction to complete its data read/write
- Control hazard
  - Decision on control action depends on previous instruction
- Structure hazards
  - A required resource is busy

Structure Hazards

- Conflict for use of a resource
- If common memory for program and data
  - 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
  - separate instruction/data caches (Harvard)
  - Common low-level memory (Von Neumann)

Data Hazards in ALU Instructions

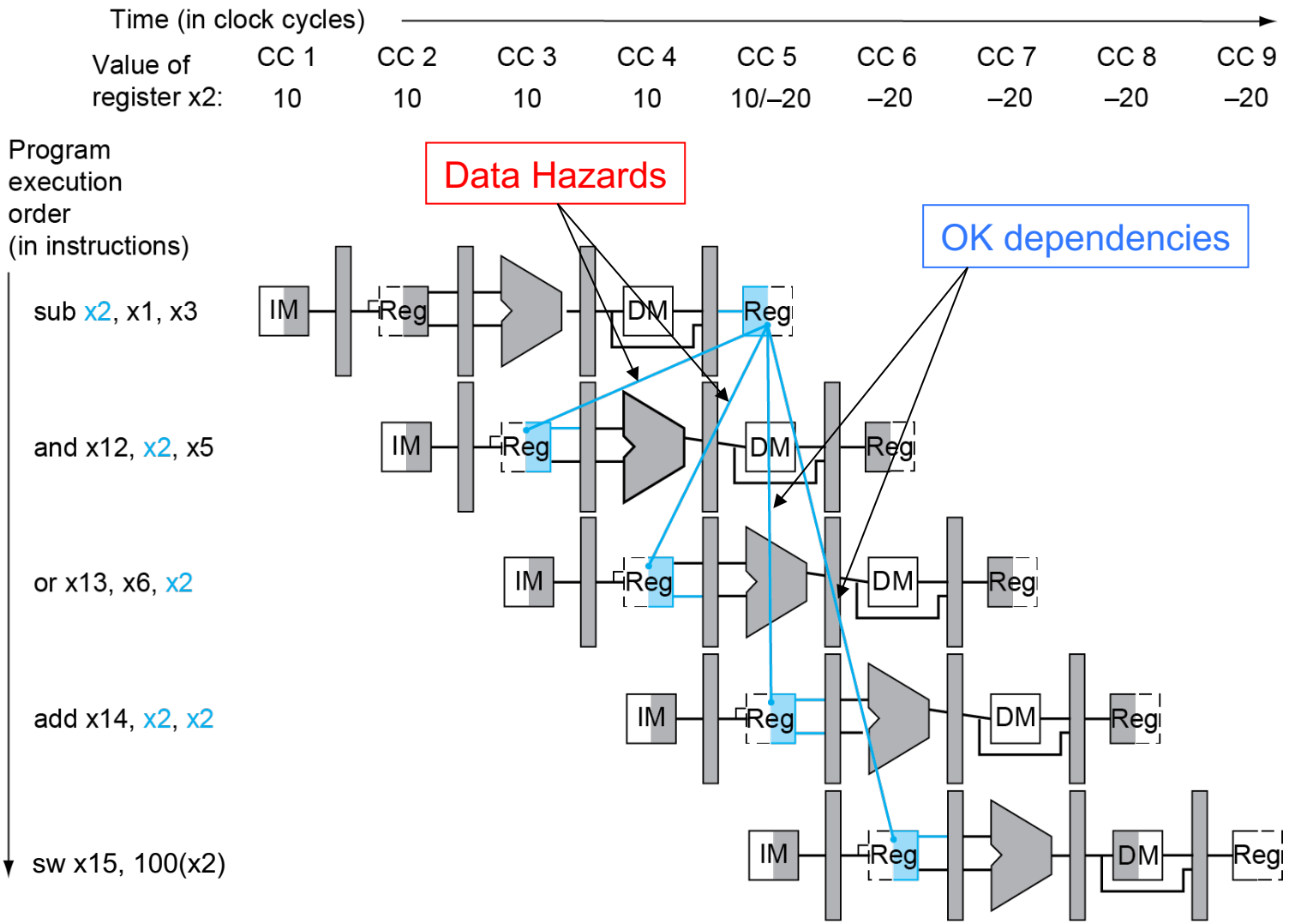
- Consider this sequence:

```
(1)  sub   x2, x1,x3
(2)  and   x12,x2,x5
(3)  or    x13,x6,x2
(4)  add   x14,x2,x2
(5)  sw    x15,100(x2)
```

- Data dependencies between:

```
(1)  &  (2)
(1)  &  (3)
(1)  &  (4)
(1)  &  (5)
```

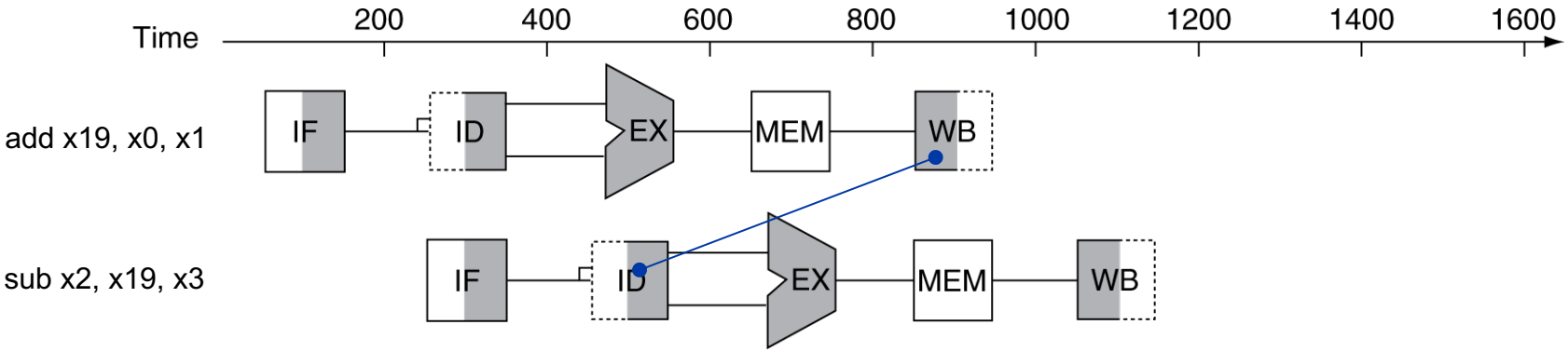
Data Dependency



Data Hazards

- If an instruction depends on completion of a data by a previous instruction
  - Then the instruction must be delayed
  - Example

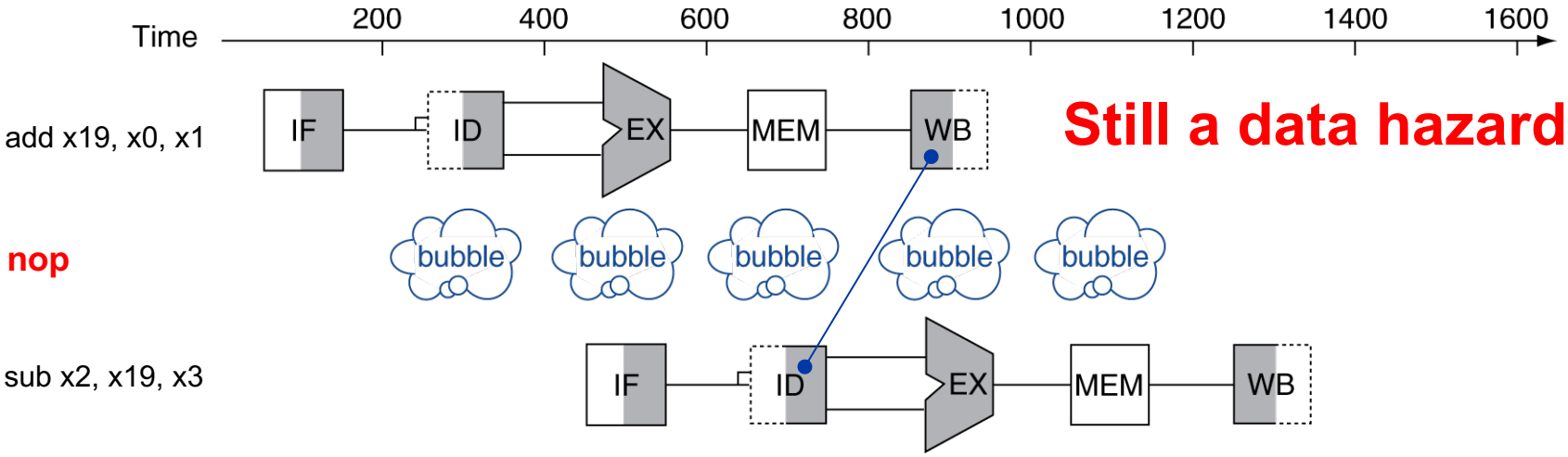
```
add  x19, x0, x1
sub  x2, x19, x3
```



Data Hazards

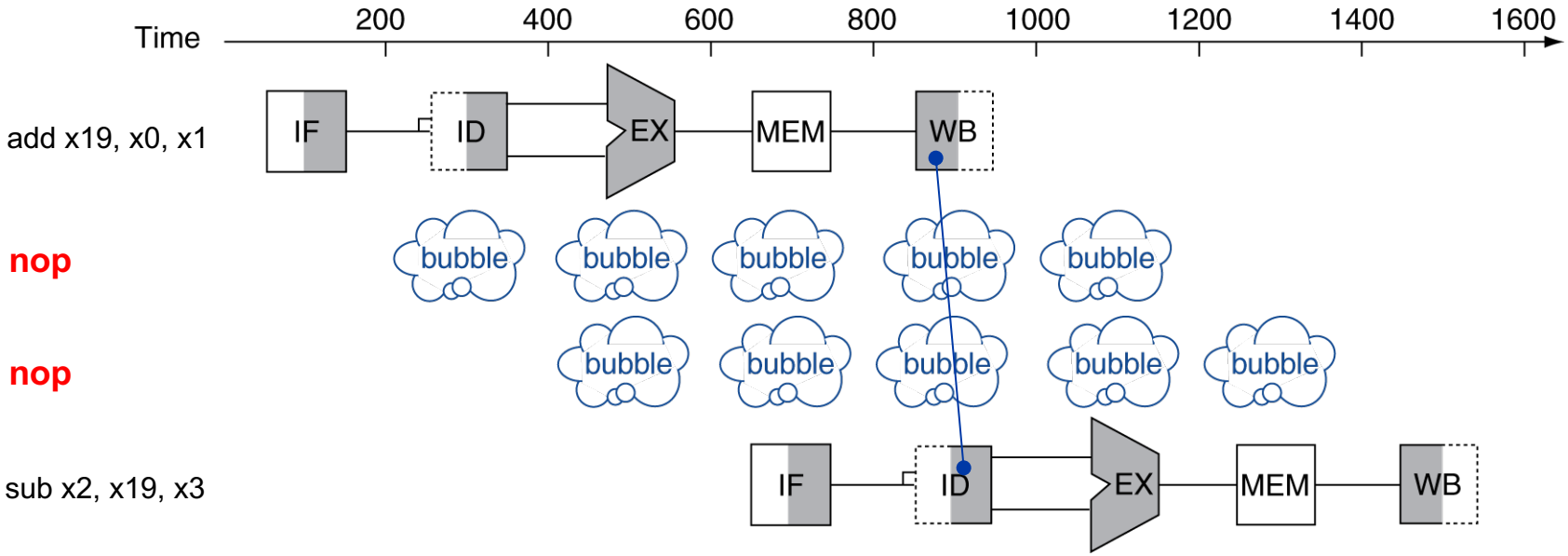
- The instruction must be delayed by
  - Inserting ***bubbles*** or ***stalls*** with a `nop`

```
nop   # no operation, pseudo-instruction
      # addi x0, x0, 0
```



Data Hazards

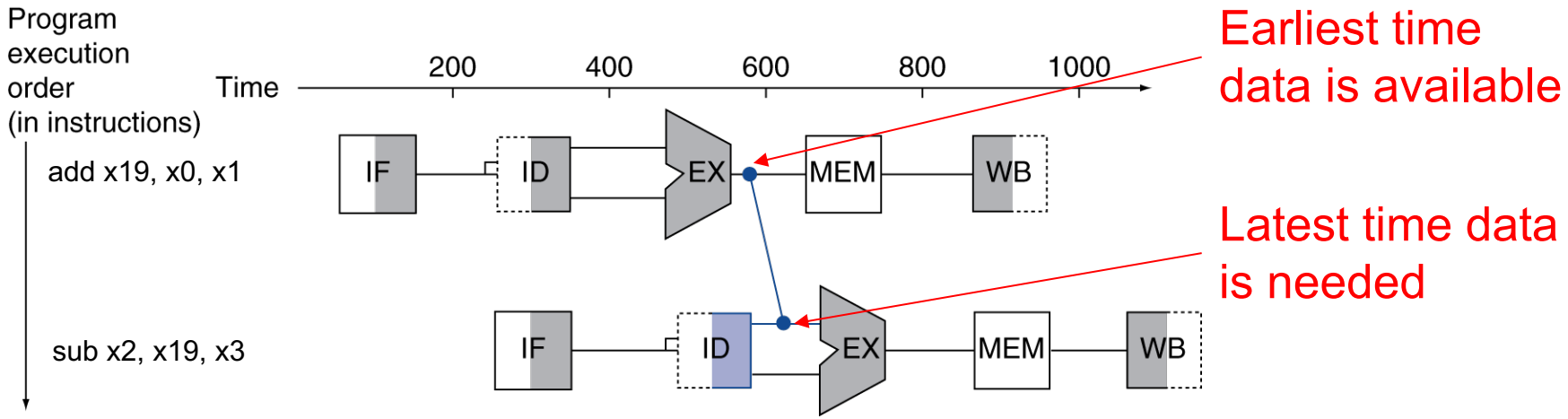
- Keep adding bubbles until the hazard is resolved



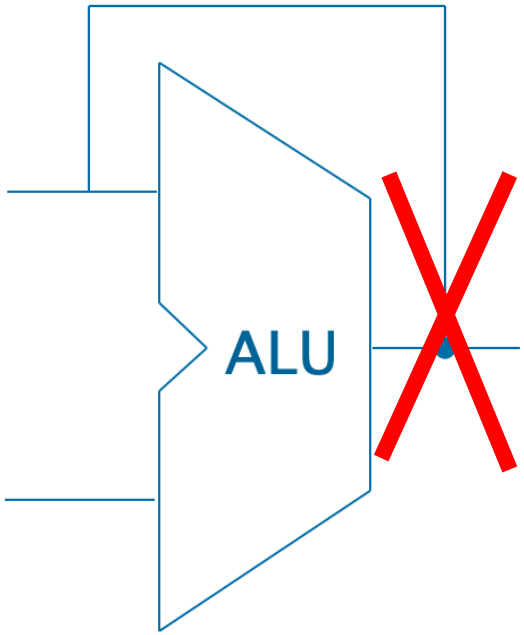
Data hazard resolved

Forwarding (aka Bypassing)

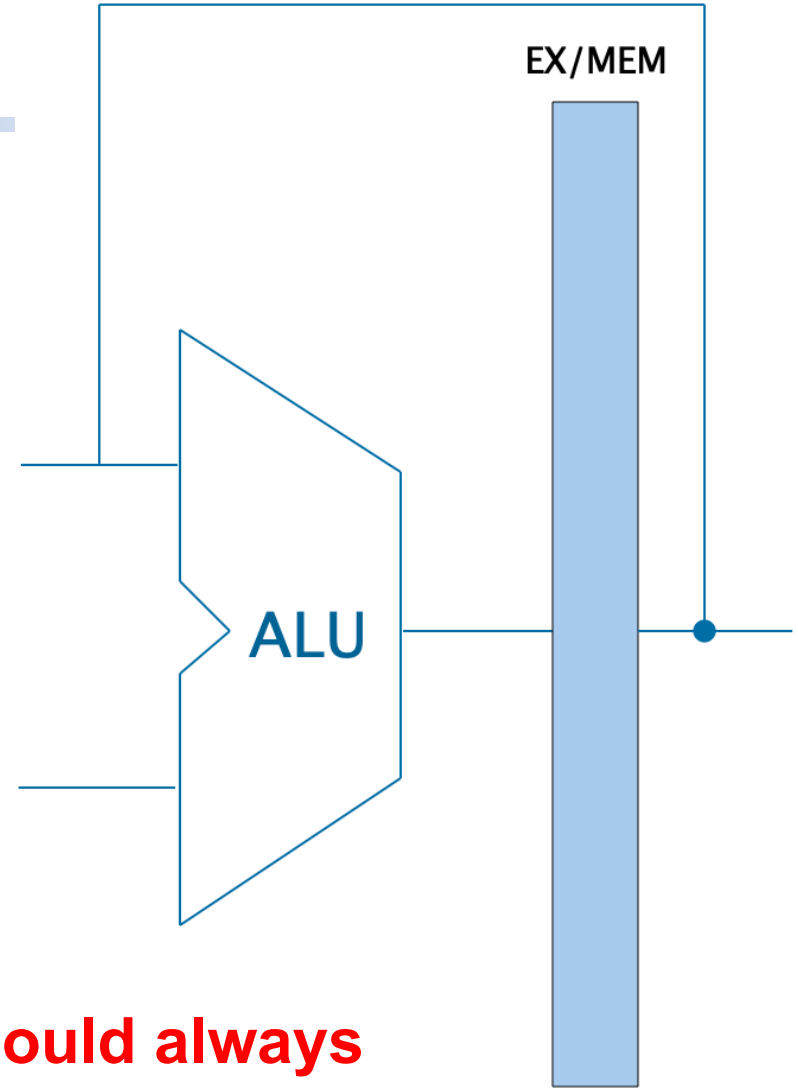
- Inserting nop (delay) compromises performance
- Data hazard **may** be solved with **Forwarding**
- Use result as soon as it is available
  - Don't wait for it to be stored in a register
  - Requires extra hardware in the datapath



Forwarding Path

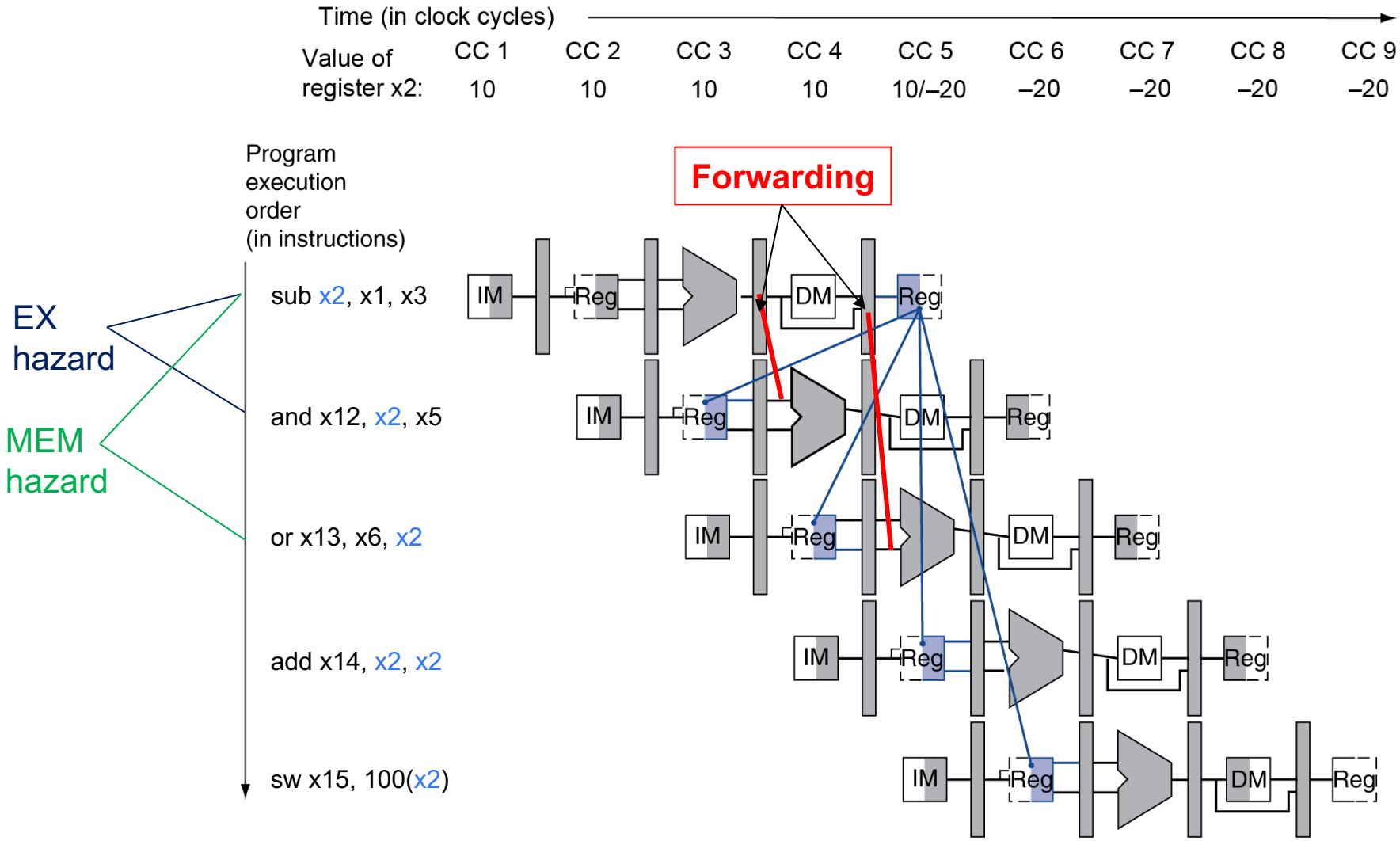


Can’t connect input and output of a component directly



Should always connect data from a pipeline register

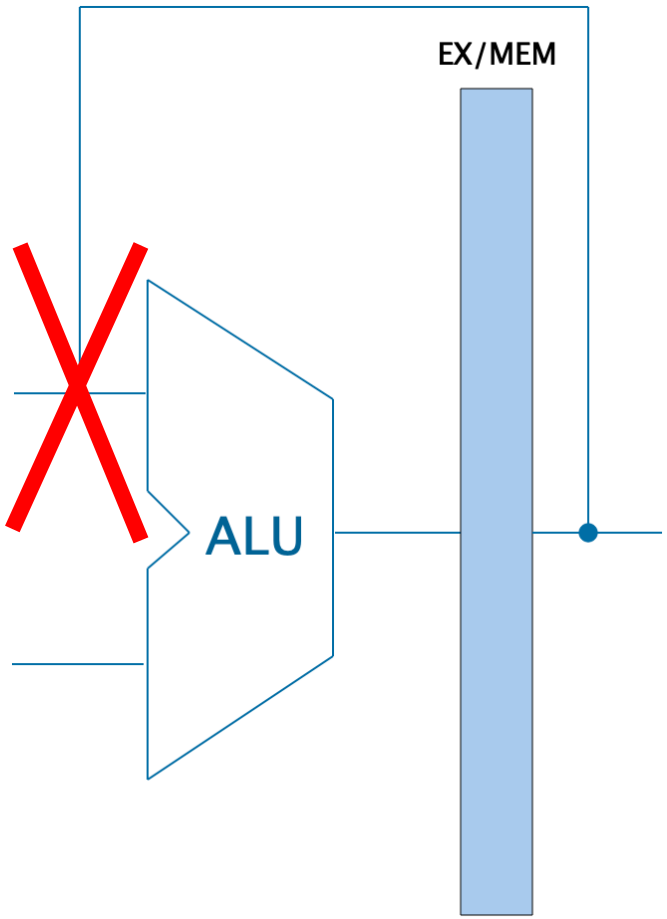
Forwarding from Registers

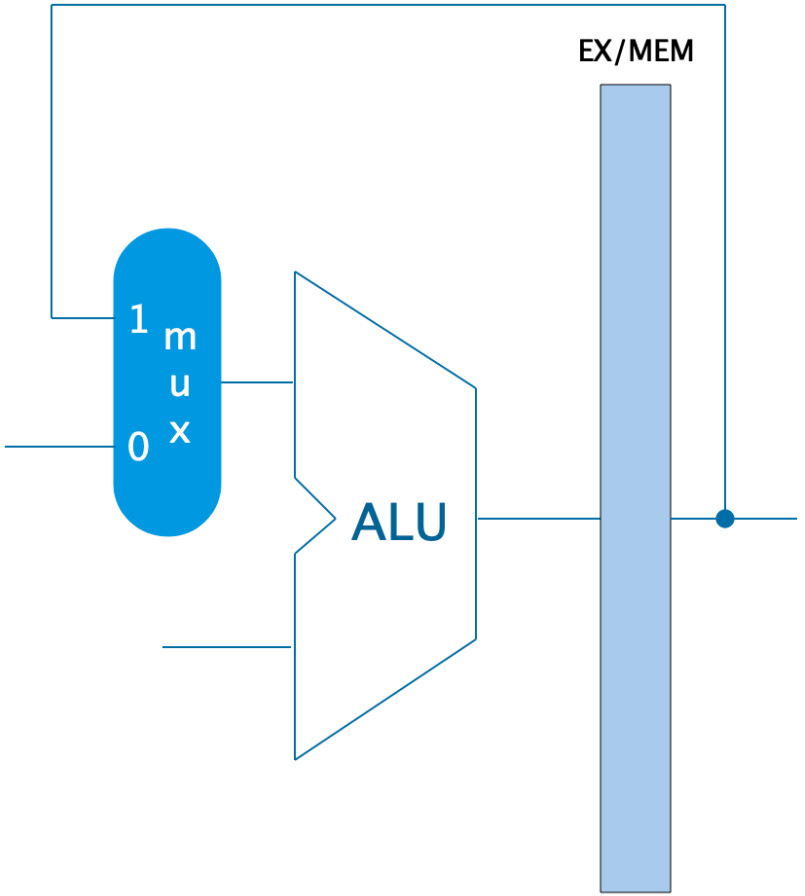


Type of data hazards

- EX hazard
  - Data hazard between two adjacent instructions
  - Forwarding path from EX/MEM pipeline register to ALU
- MEM hazard
  - Data hazard between two instructions with one more instruction in between
  - Forwarding path from MEM/WB pipeline register to ALU

Forwarding Path



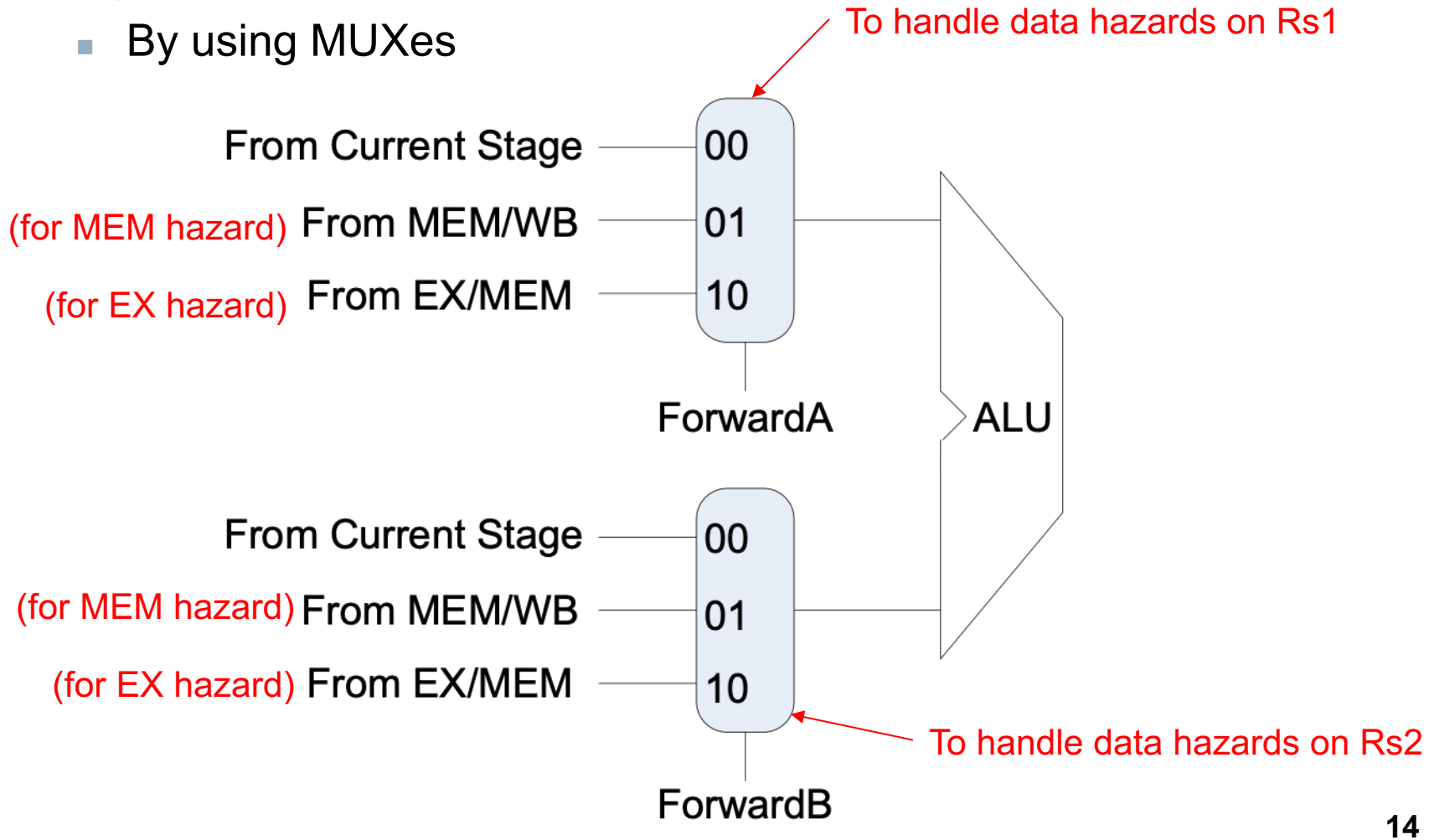


Can't just join 2 wires

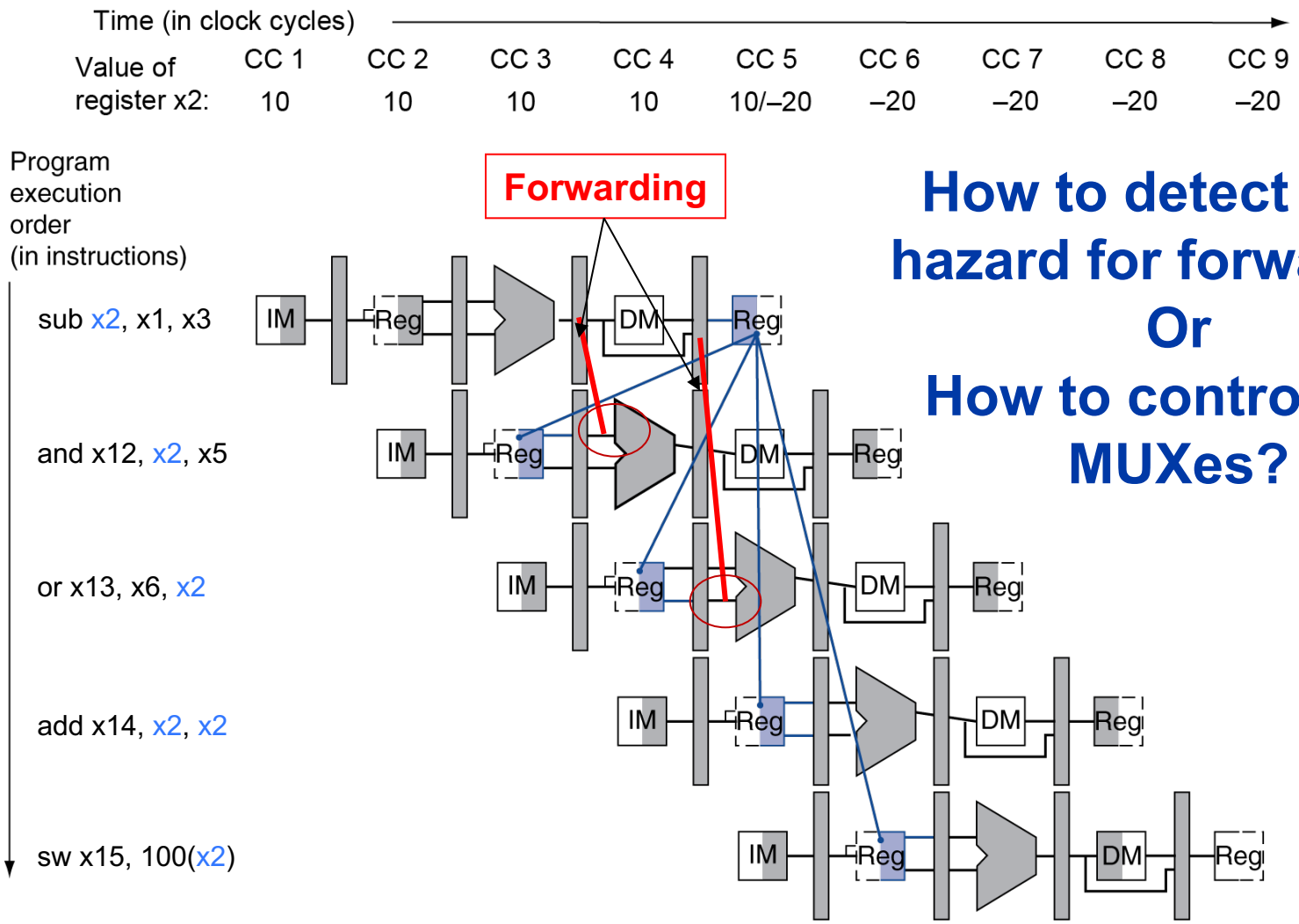
Use mux

Forwarding Paths

- Forwarding paths are created between stage pipeline registers and ALU inputs
  - By using MUXes



Dependencies & Forwarding



How to detect data hazard for forwarding
Or
How to control the MUXes?

Detecting the Need to Forward

- Denotation:
  - e.g., ID/EX.RegisterRs1 = register number for Rs1 sitting in ID/EX pipeline register
- ALU operand register numbers in EX stage are given by
  - ID/EX.RegisterRs1, ID/EX.RegisterRs2
- Data hazards when

EX Hazard

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

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

Fwd from EX/MEM pipeline reg

MEM Hazard

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

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

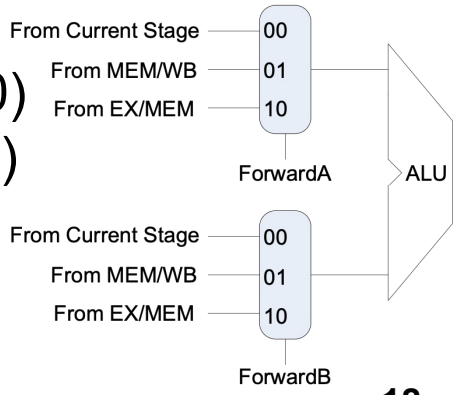
Fwd from MEM/WB pipeline reg

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 x0
  - EX/MEM.RegisterRd ≠ 0, MEM/WB.RegisterRd ≠ 0

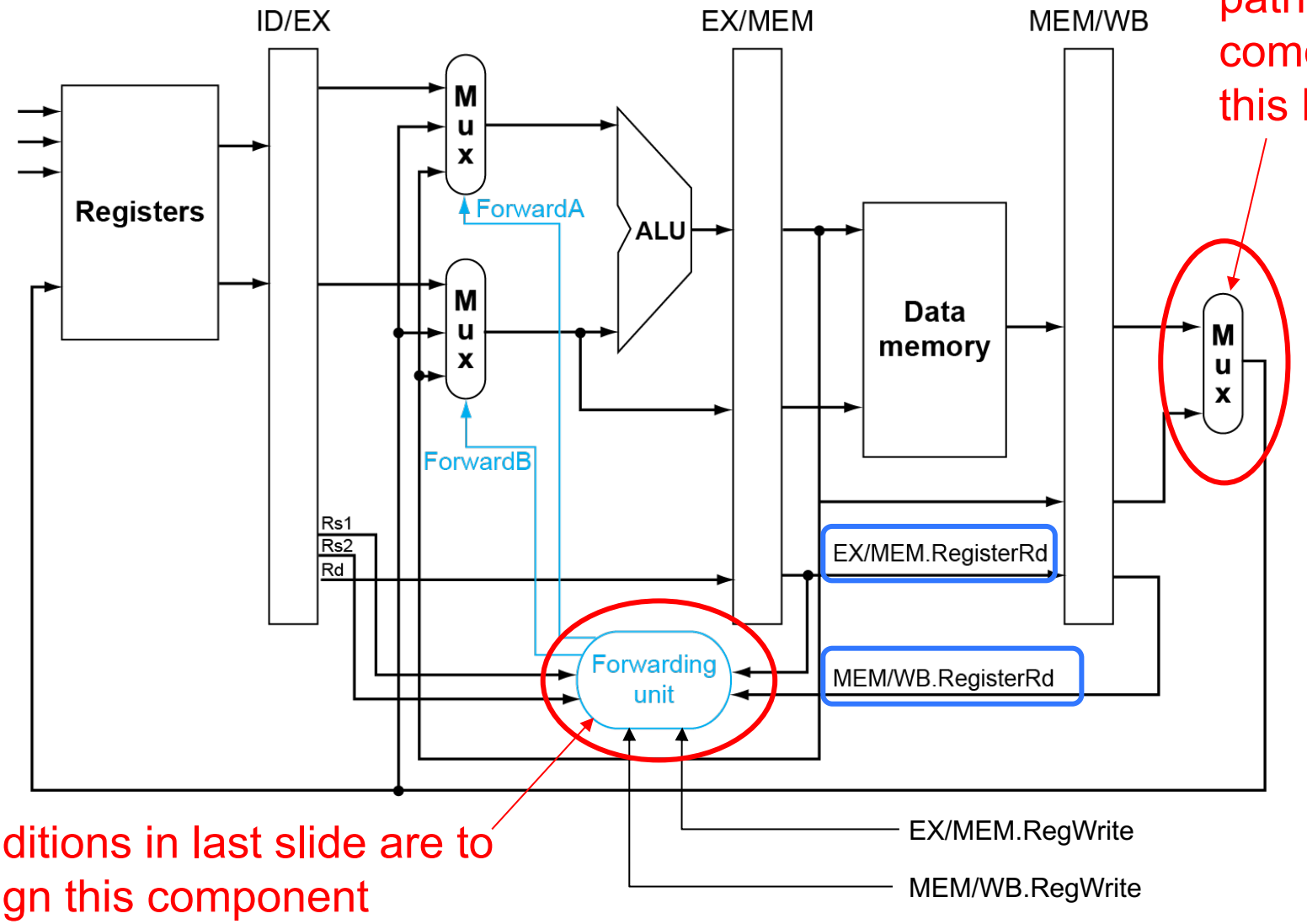
Revised Forwarding Conditions

- EX hazard
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd == ID/EX.RegisterRs1))
    MUX select signal ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd == ID/EX.RegisterRs2))
    MUX select signal ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd == ID/EX.RegisterRs1))
    MUX select signal ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd == ID/EX.RegisterRs2))
    MUX select signal ForwardB = 01



Forwarding Paths

Forwarding Unit in **EX** stage



MEM/WB forwarding path actually comes from this MUX

Conditions in last slide are to design this component

Double Data Hazard

- Consider the sequence:

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

- Conditions for both hazards are satisfied
  - Want to use the most recent
  - Forwarding between 1st and 3rd – MEM hazard condition should be disabled
- Revise MEM hazard condition
  - Only fwd if EX hazard condition isn't true

Revised Forwarding Condition

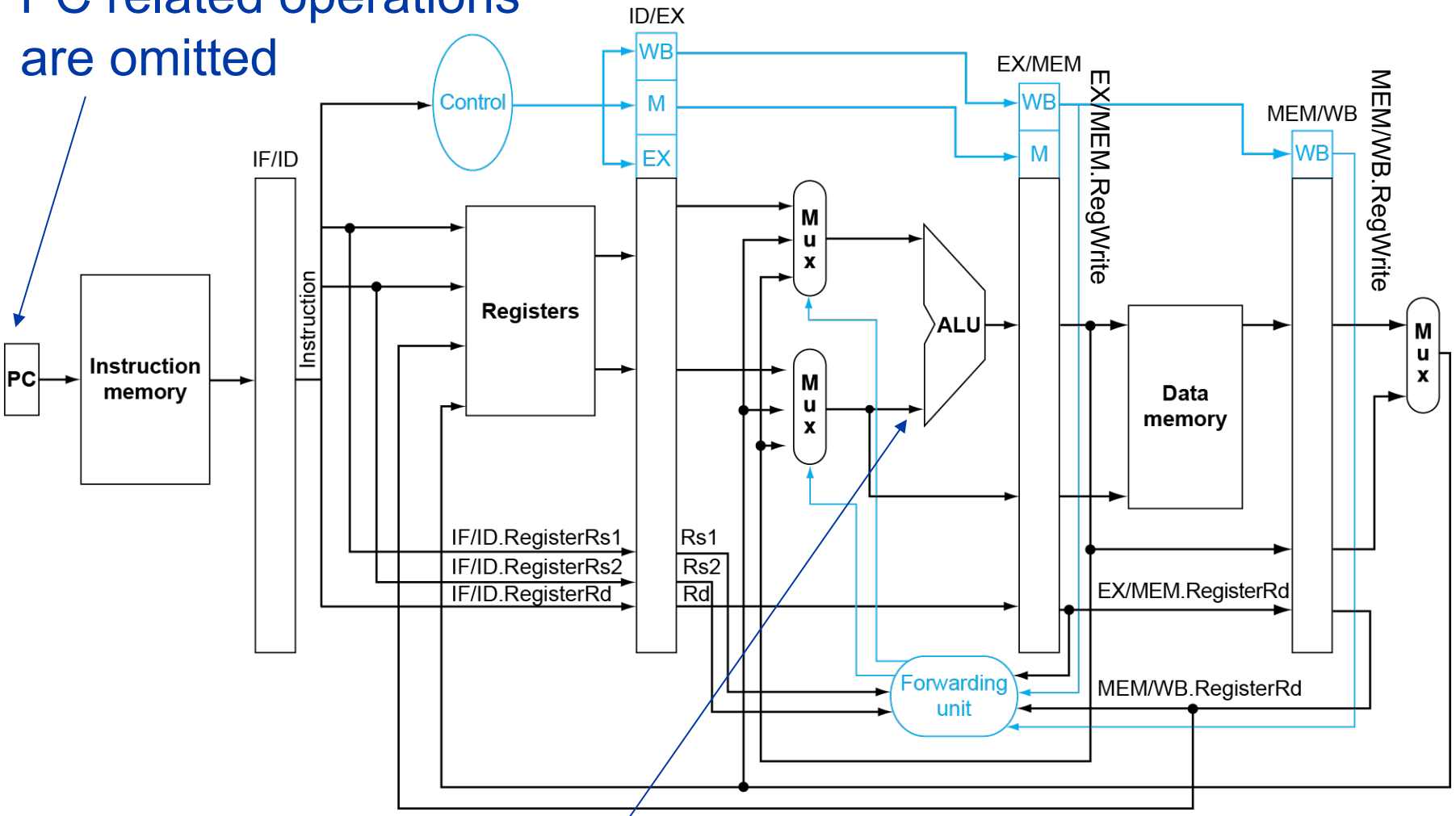
- MEM hazard (not EX hazard)
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs1)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs1)))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs2)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs2)))
    ForwardB = 01

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.

Datapath with Forwarding

PC related operations are omitted

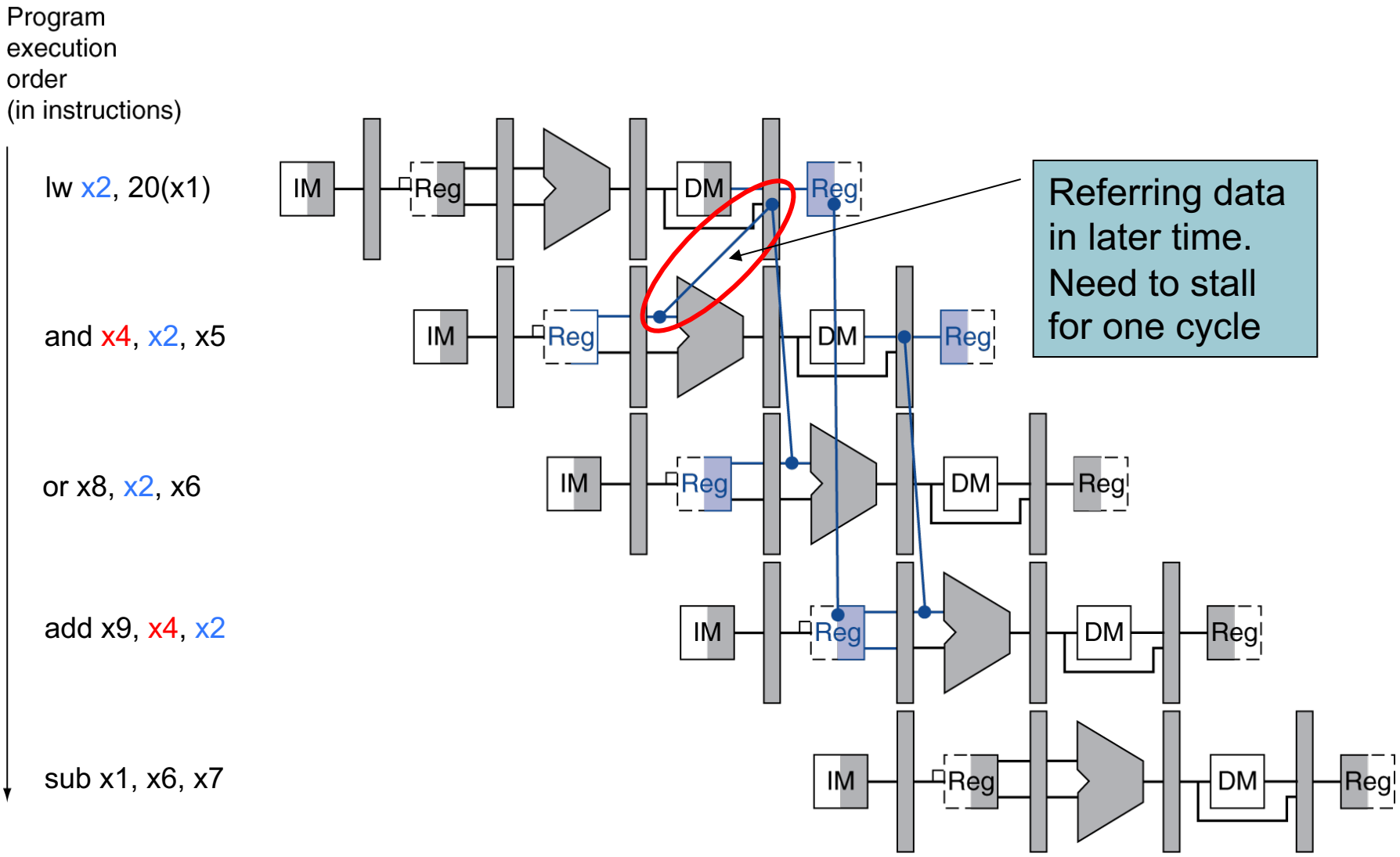


Note: Imm Gen and ALUSrc mux are omitted for simplicity purpose

Forwarding May Fail

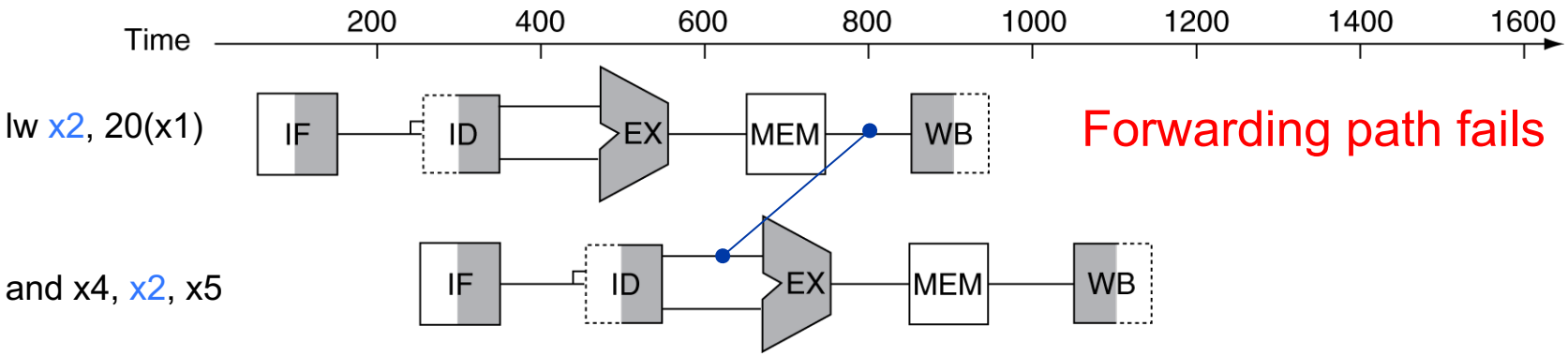
- Can’t always avoid stalls by forwarding
  - If value not computed when needed
  - Can’t forward back in time!

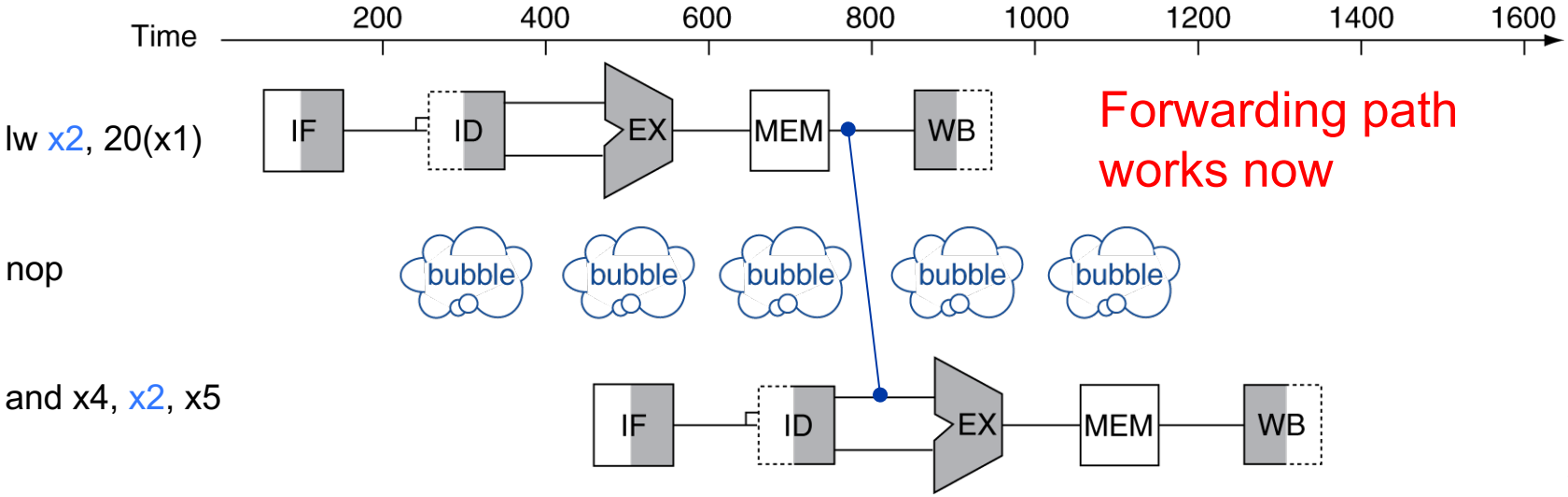
Load-Use Data Hazard



Load-Use Data Hazard

- Resolve by inserting stalls

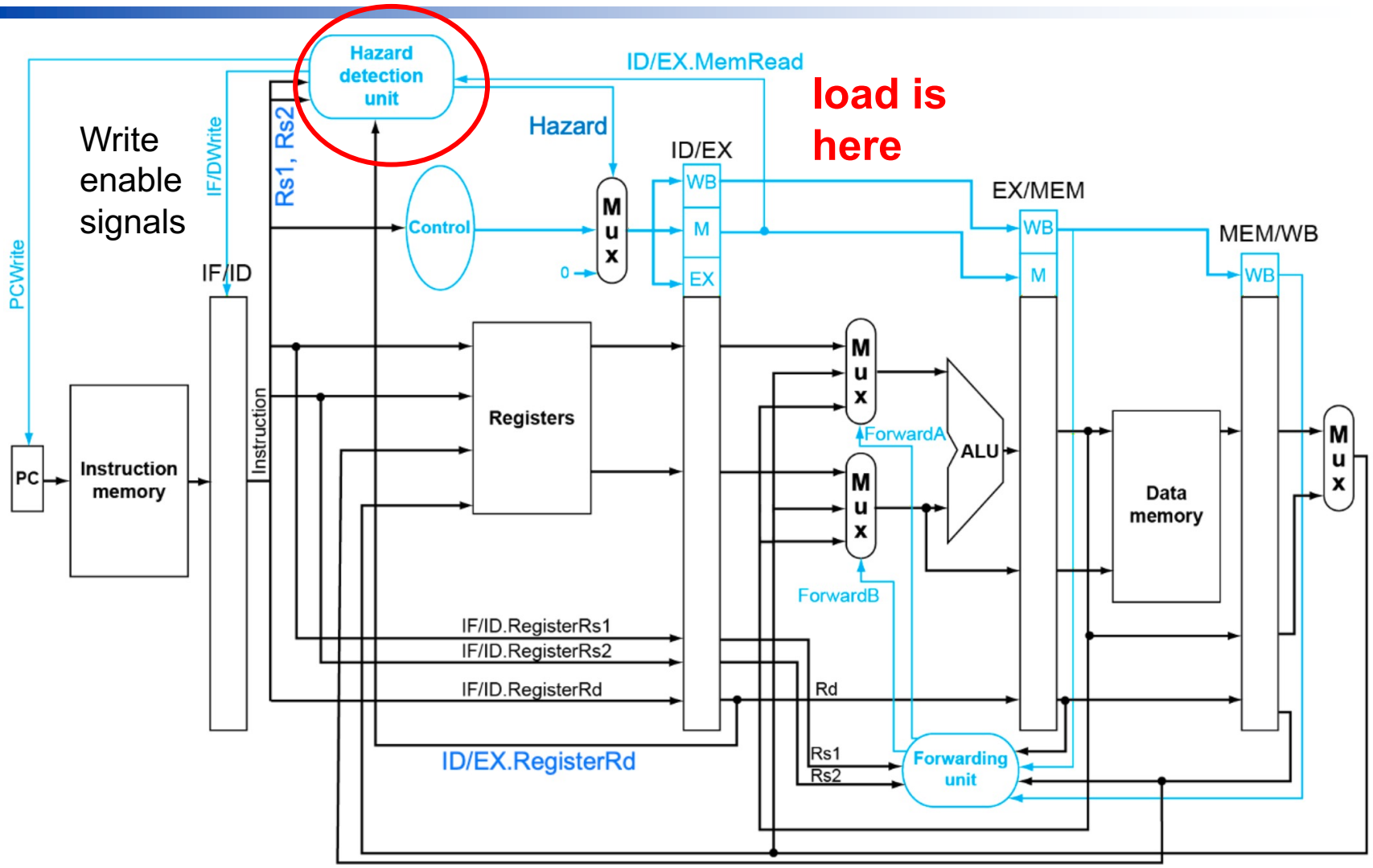




Load-Use Hazard Detection

- Load-use hazard is checked when a ***load*** instruction is in the **EX** stage, indicated by
  - ID/EX.MemRead
- Load-use hazard happens when
  - ID/EX.MemRead and
    ((ID/EX.RegisterRd = IF/ID.RegisterRs1) or
    (ID/EX.RegisterRd = IF/ID.RegisterRs2))
- **Hazard Detection** unit is added in **ID** stage to detect the load-use hazard
- If detected, stall and insert bubble

Datapath with Hazard Detection

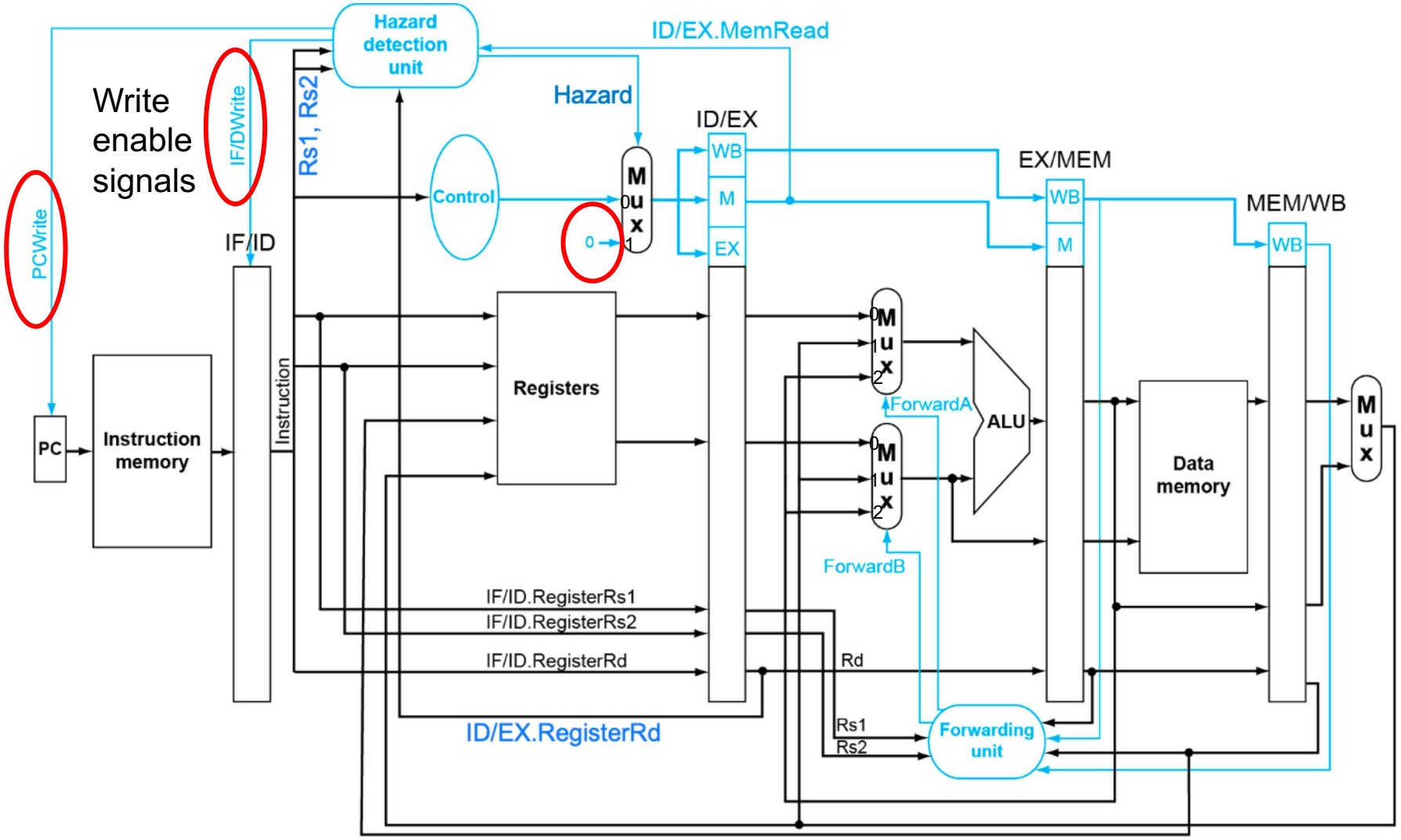


How to Stall the Pipeline?

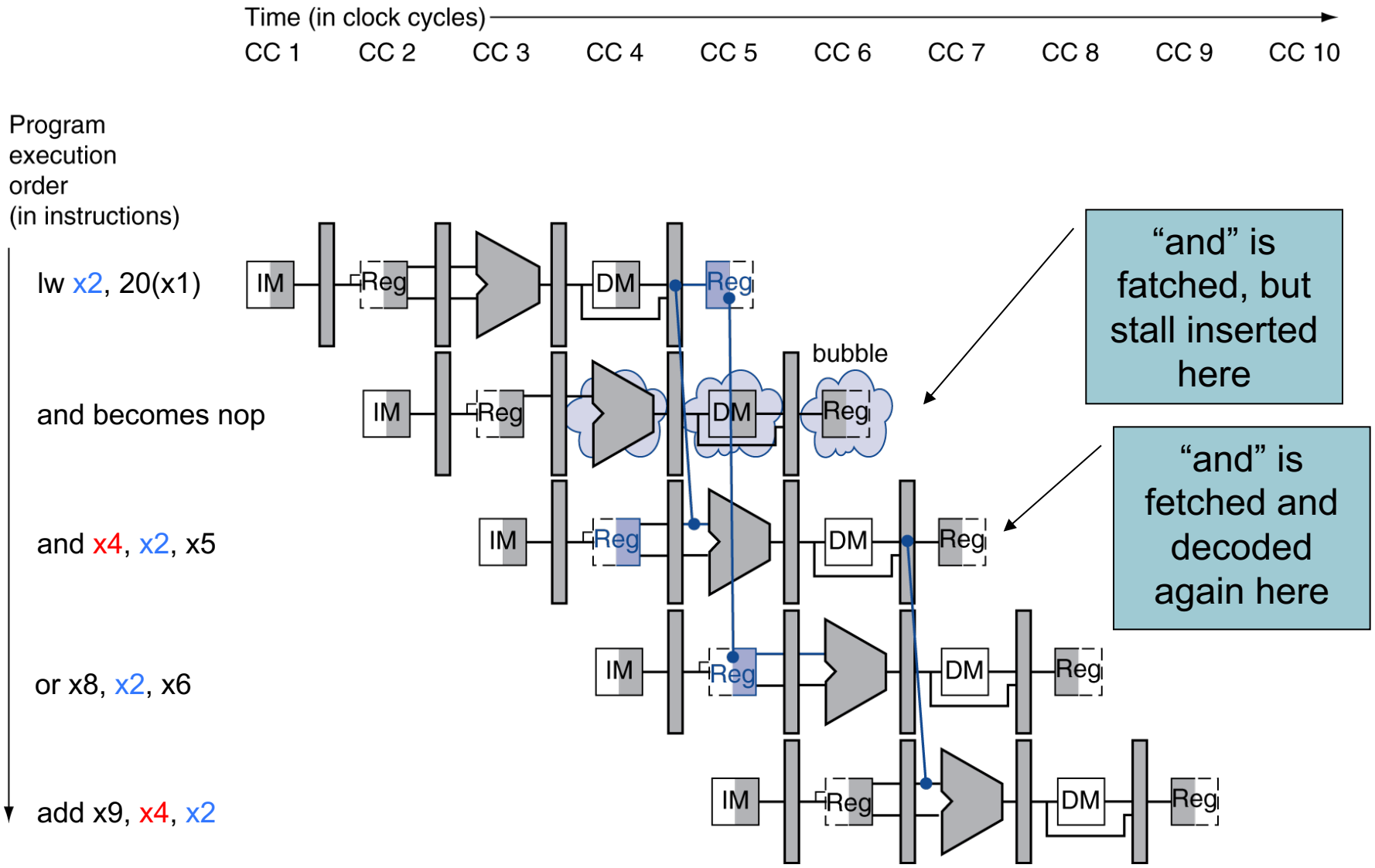
- (How to insert bubbles when load use hazard is detected?)

1. Force control signals in ID/EX pipeline register to 0's
   - EX, MEM and WB in following cycles do no-operation (nop) with those 0 control signals
2. Prevent updates of PC and IF/ID registers
   - Instruction in IF/ID is held and decoded again
   - PC is held, so the same instruction is fetched again

Datapath with Hazard Detection



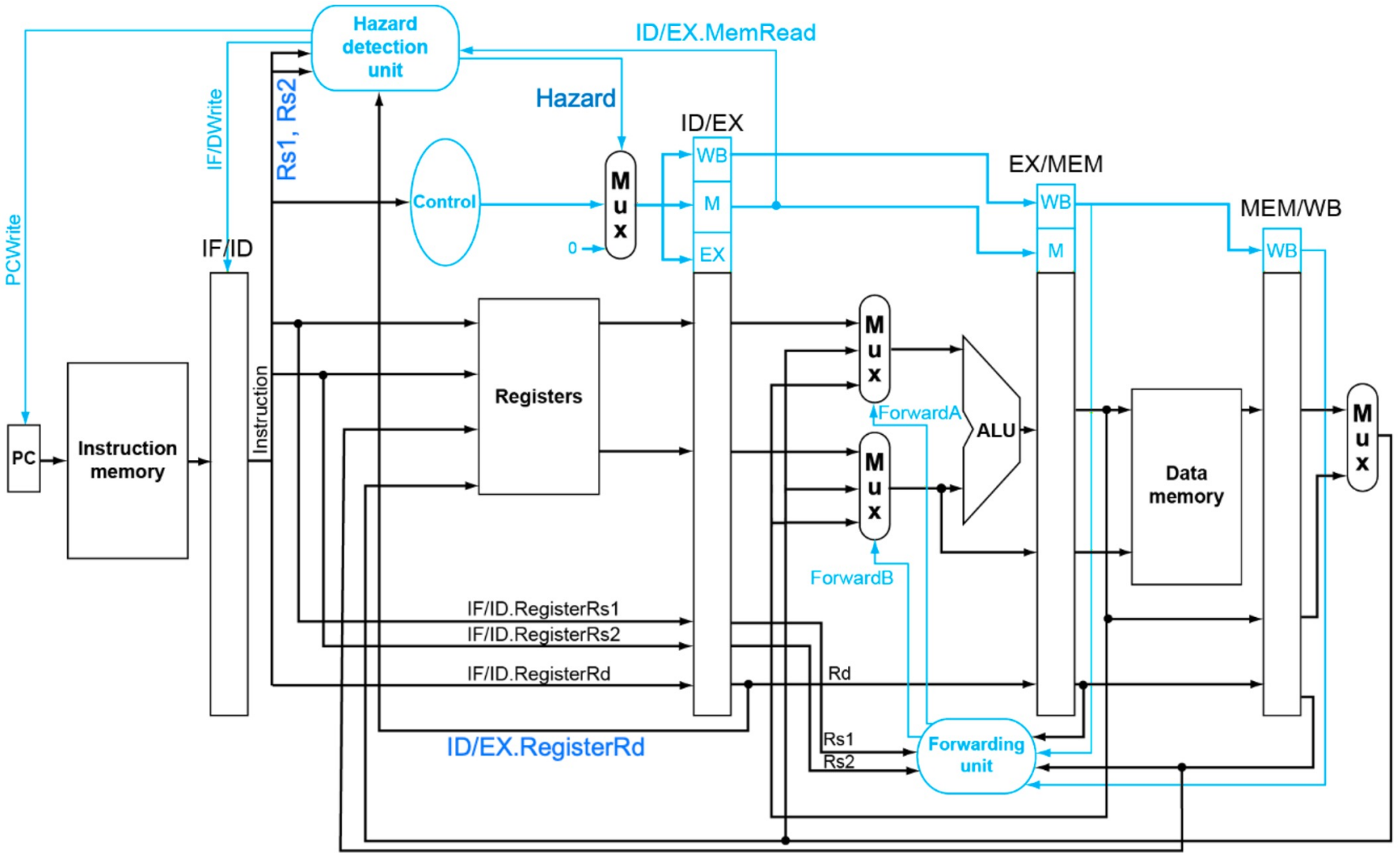
Stall/Bubble in the Pipeline



CC1

lw x2, 20(x1)

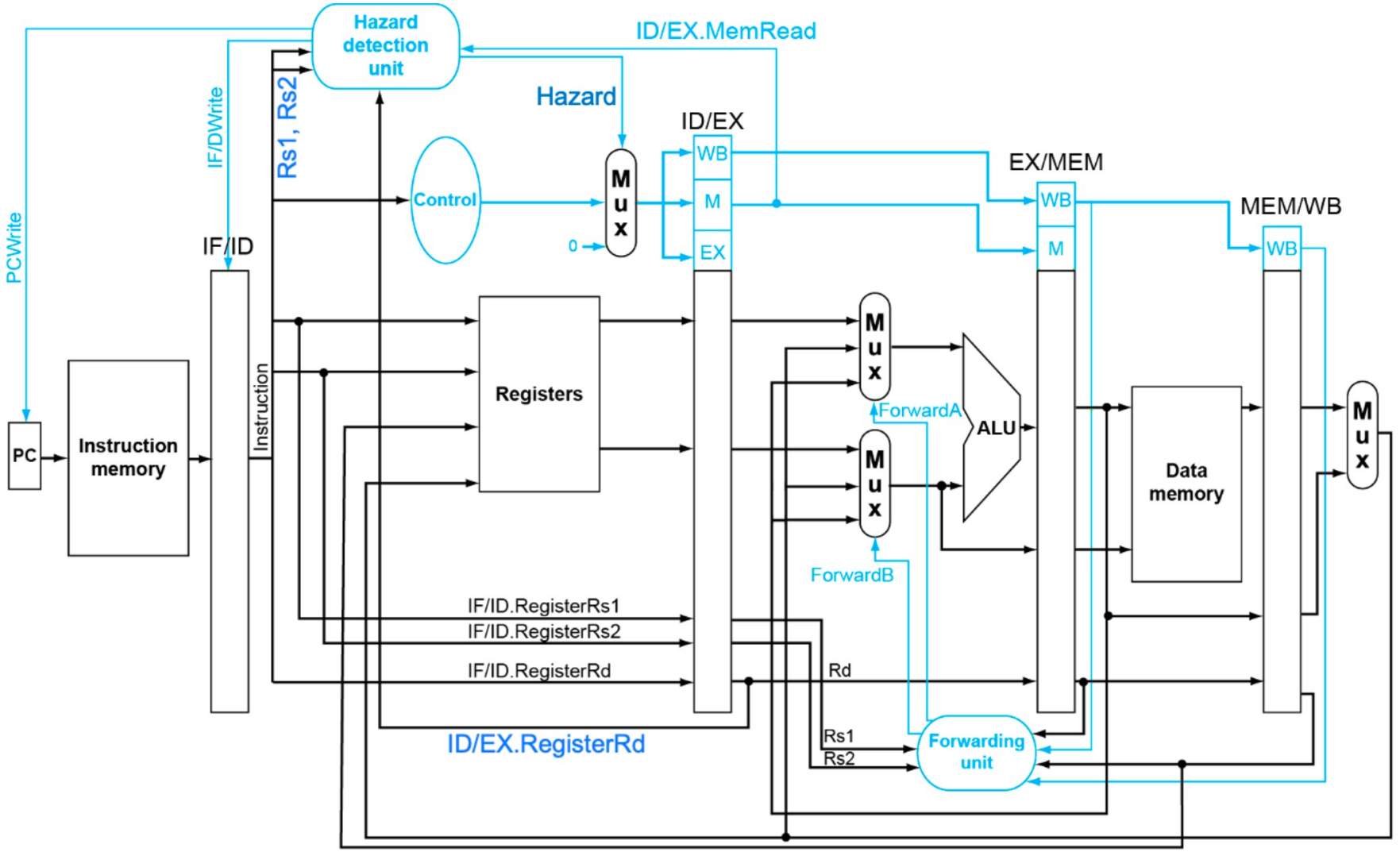
PC Write	IF/ID Write	Hazard	ID/EX. MemRead	IF/ID .Rs1	IF/ID .Rs2	ID/EX. Rd
1	1	x	x	x	x	x



CC2

PC Write	IF/ID Write	Hazard	ID/EX. MemRead	IF/ID. Rs1	IF/ID. Rs2	ID/EX. Rd
1	1	x	x	1	20	x

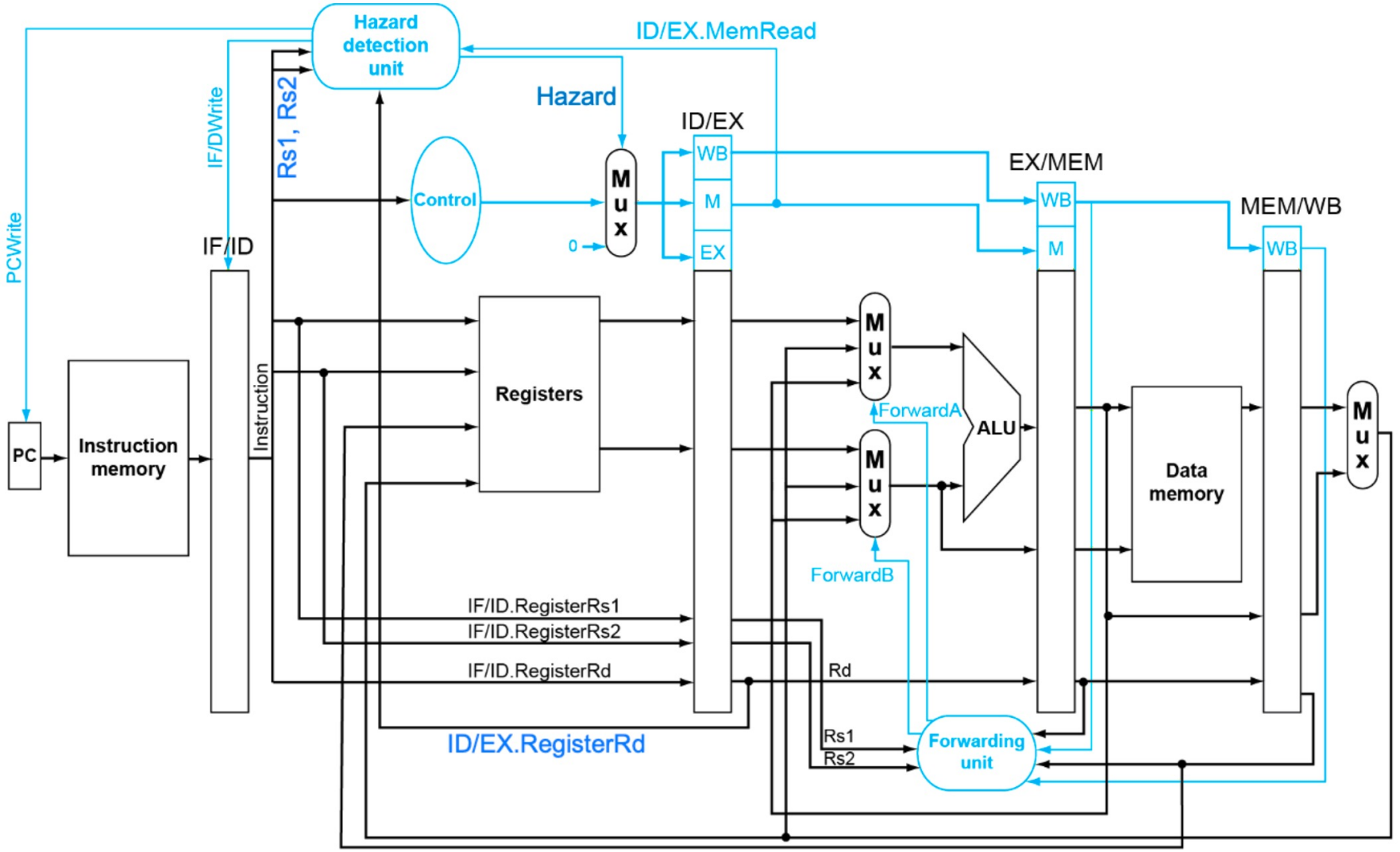
and x4,x2,x5 lw x2, 20(x1)



CC3

PC Write	IF/ID Write	Hazard	ID/EX. MemRead	IF/ID. Rs1	IF/ID. Rs2	ID/EX. Rd
0	0	1	1	2	5	2

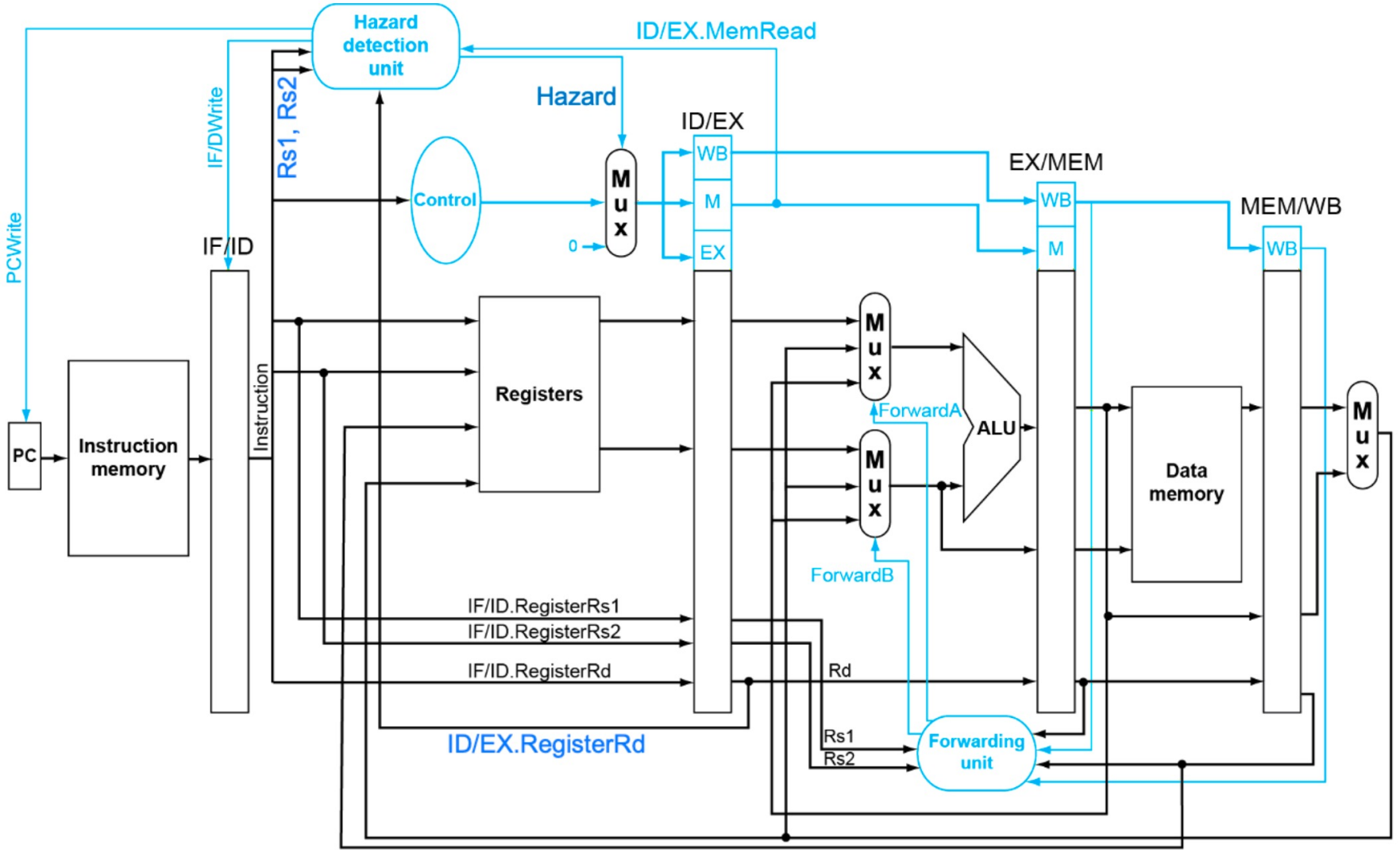
or x8,x2,x6 and x4,x2,x5 lw x2, 20(x1)



CC4

PC Write	IF/ID Write	Hazard	ID/EX. MemRead	IF/ID. Rs1	IF/ID. Rs2
1	1	0	0	2	5

or x8,x2,x6 and x4,x2,x5 nop (and) lw x2, 20(x1)



CC4

ID/EX.Rs1	ID/EX.Rs2	ID/EX.Rd	EX/MEM.Rd	MEM/WB.Rd
2	5	4	2	x

EX/MEM.RegWrite	MEM/WB.RegWrite	ForwardA	ForwardB
1	x	10	00

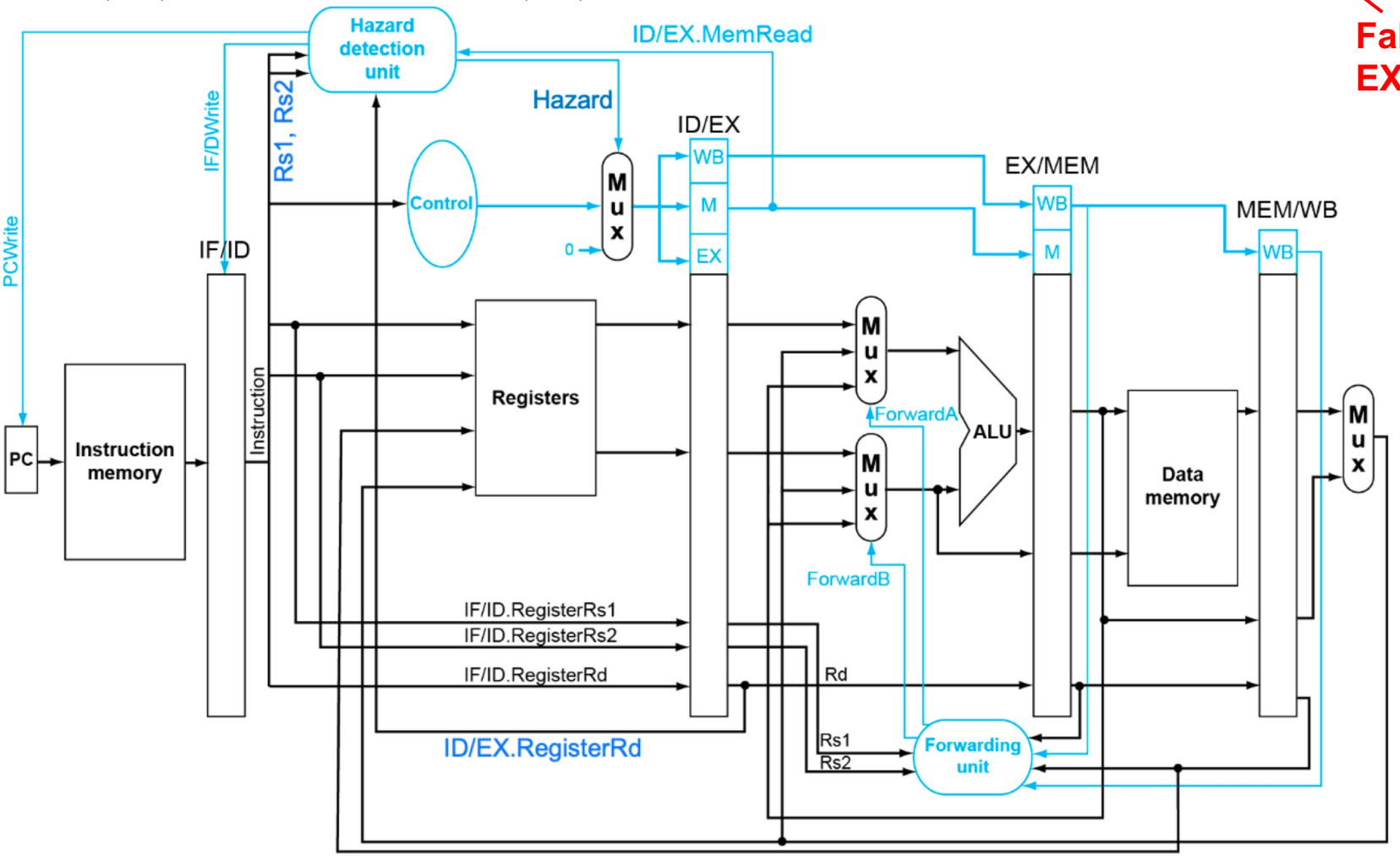
or x8,x2,x6

and x4,x2,x5

nop (and)

lw x2, 20(x1)

Fake
EX hazard

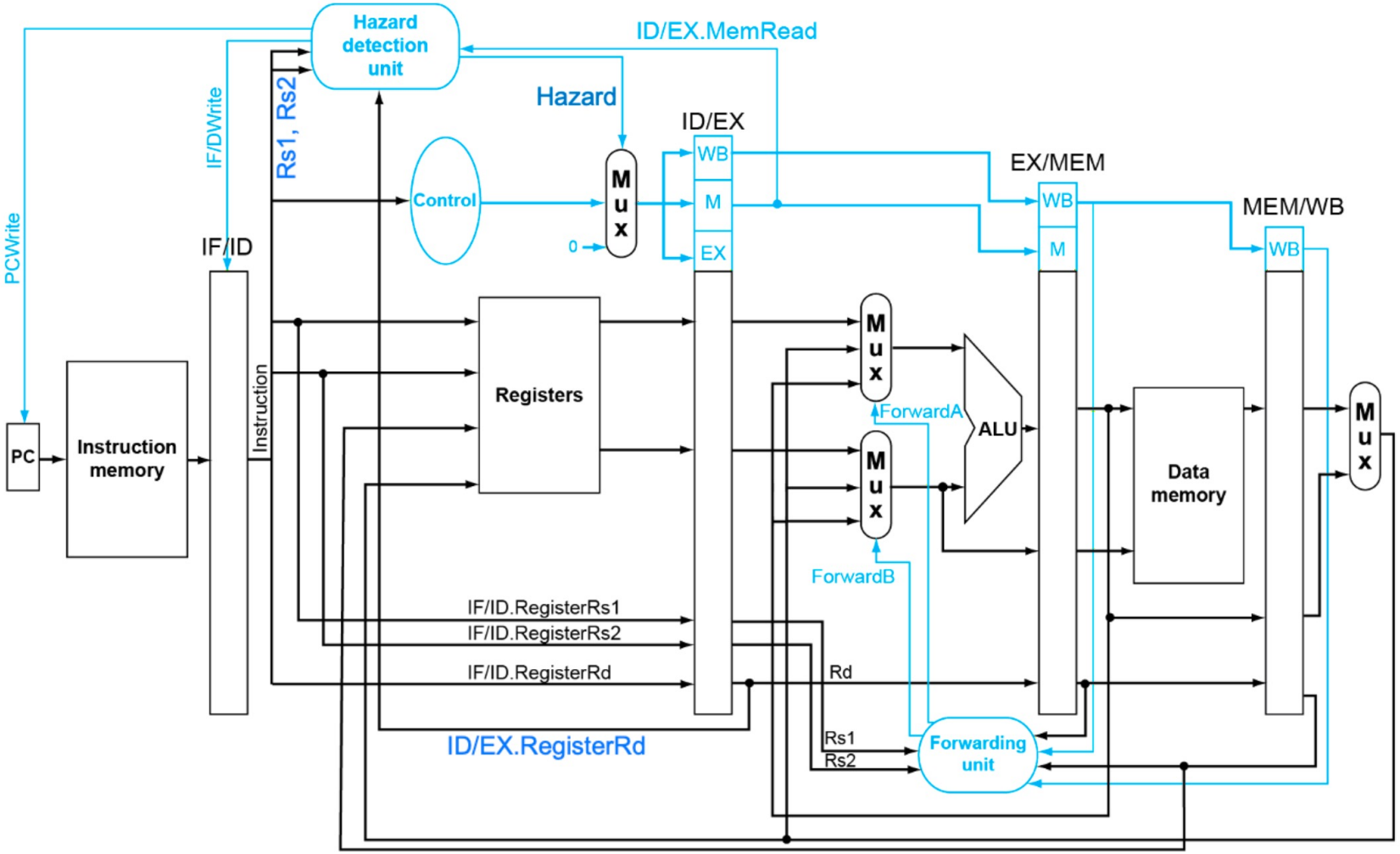


CC5

PC Write	IF/ID Write	Hazard	EX/MEM. MemRead	EX/MEM. MemWrite
1	1	0	**0**	**0**

No effective operation

add x9,x4,x2 or x8,x2,x6 and x4,x2,x5 **nop (and)** lw x2, 20(x1)



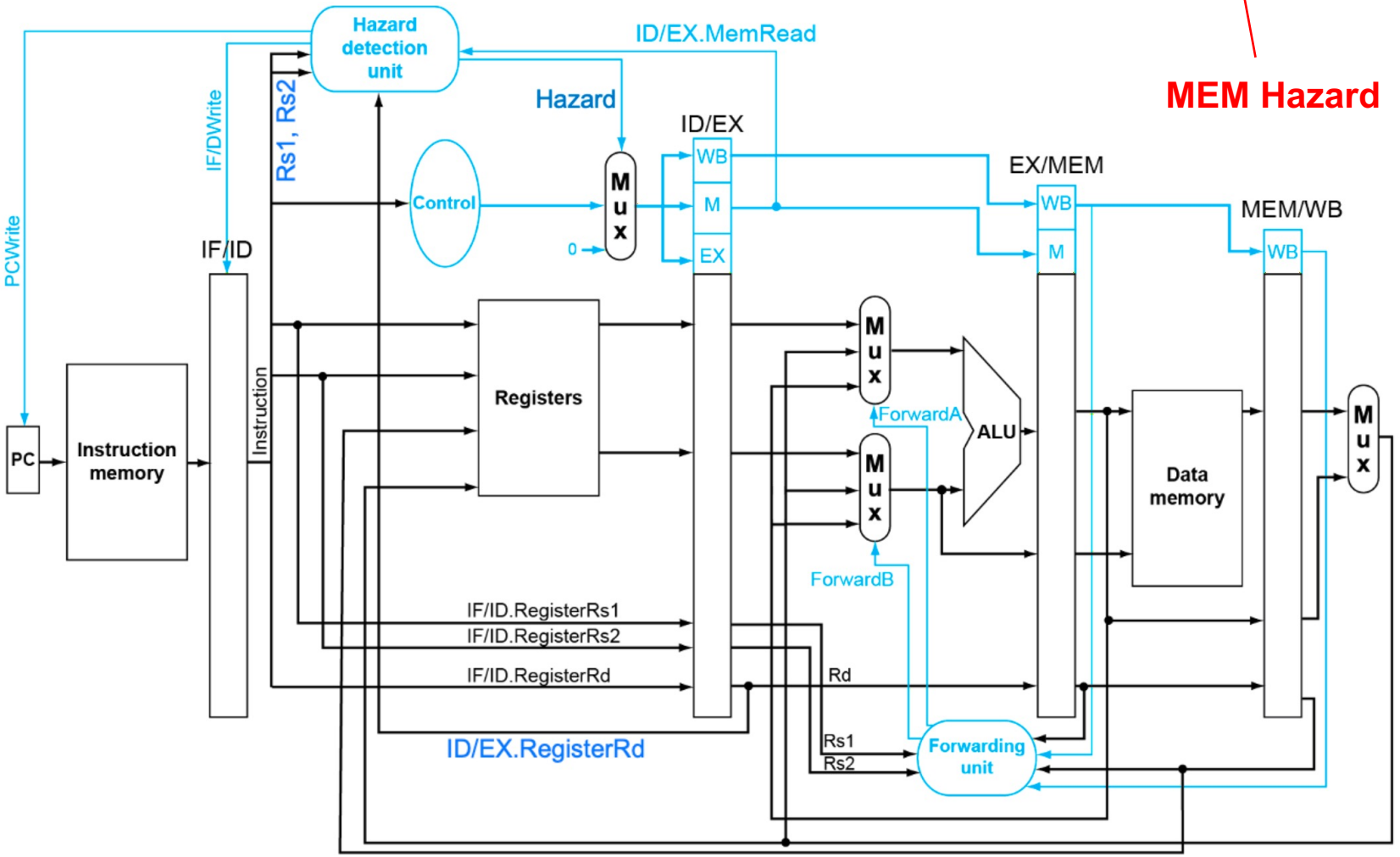
CC5

ID/EX.Rs1	ID/EX.Rs2	ID/EX.Rd	EX/MEM.Rd	MEM/WB.Rd
2	5	4	4	2

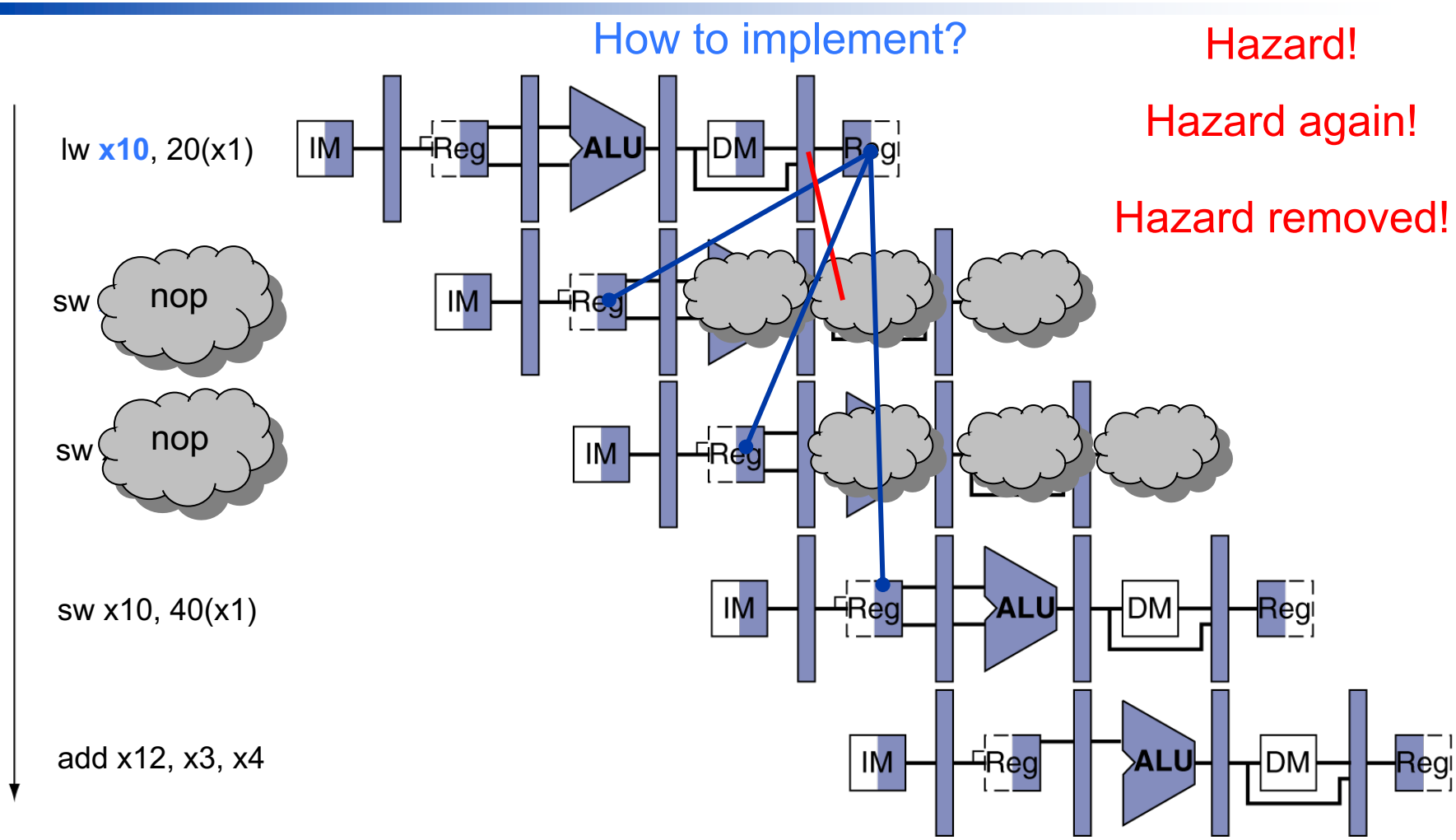
EX/MEM.RegWrite	MEM/WB.RegWrite	ForwardA	ForwardB
0	1	01	00

add x9,x4,x2 | or x8,x2,x6 | **and x4,x2,x5** | nop | **lw x2, 20(x1)**

MEM Hazard

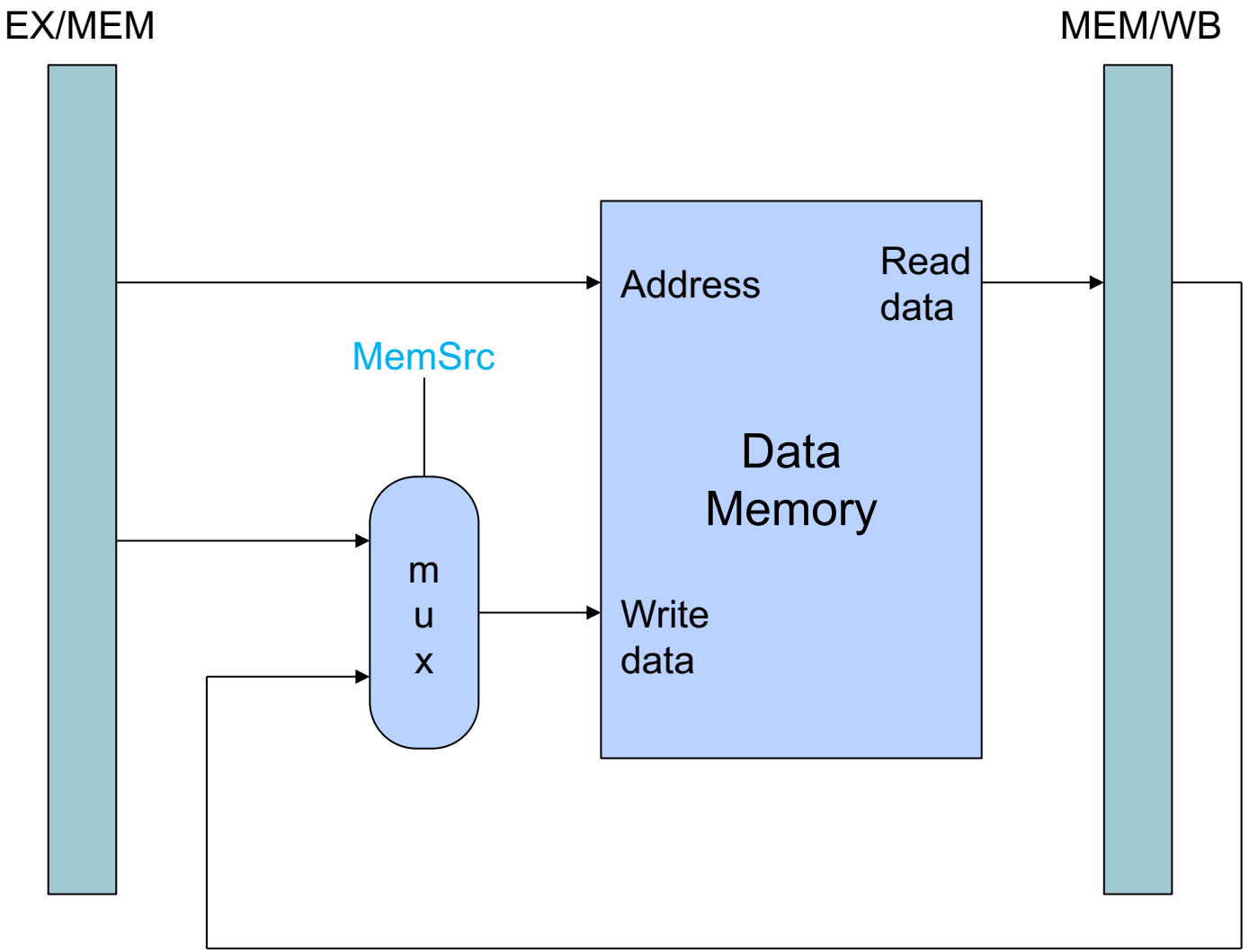


Example: Another Hazard



This delays the execution too much, can it be fixed with forwarding?

Question: how to implement



Question: how to decide MemSrc?

- MemSrc = 1 when
  (MEM/WB.Rd == EX/MEM.Rs2) and
  MEM/WB.MemRead (*already used*) and
  EX/MEM.MemWrite

Example: Is There a Hazard?

- Consider the sequence:

```
add  x3,x2,x1
addi x5,x4,3
```

An EX hazard!!!

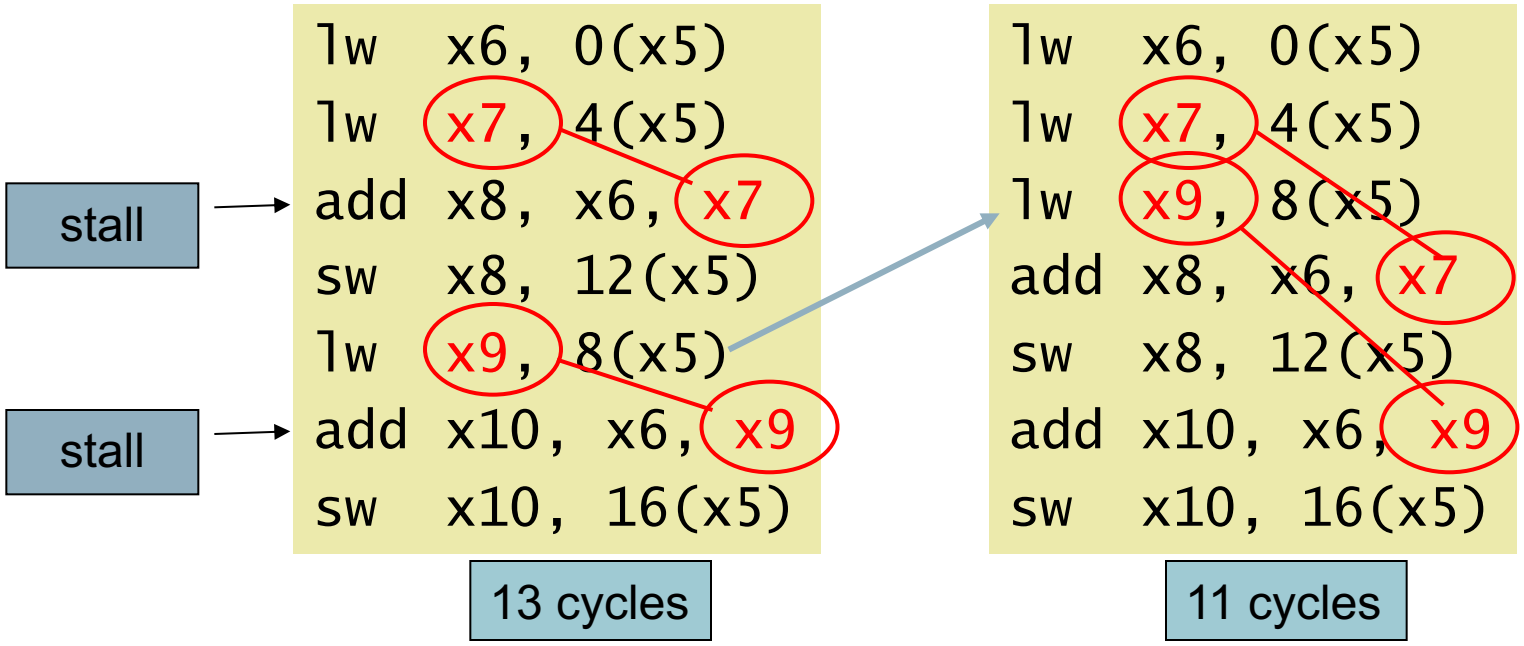
- When `add` is in MEM, `addi` in EX
  - EX/MEM.RegWrite == 1
  - EX/MEM.RegisterRd == 3 ≠ 0
  - ID/EX.RegisterRs2 == 3 (lower 5 bits of imm12) == EX/MEM.RegisterRd

But not really a hazard!!!

How to fix?

Code Scheduling to Avoid Stalls

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



Stalls and Performance

The BIG Picture

- Stalls reduce performance
  - But are required to get correct results
- Hardware can be improved to take care of the hazards automatically
  - forwarding
- Compiler can arrange code to avoid hazards and stalls
  - Requires knowledge of the pipeline structure