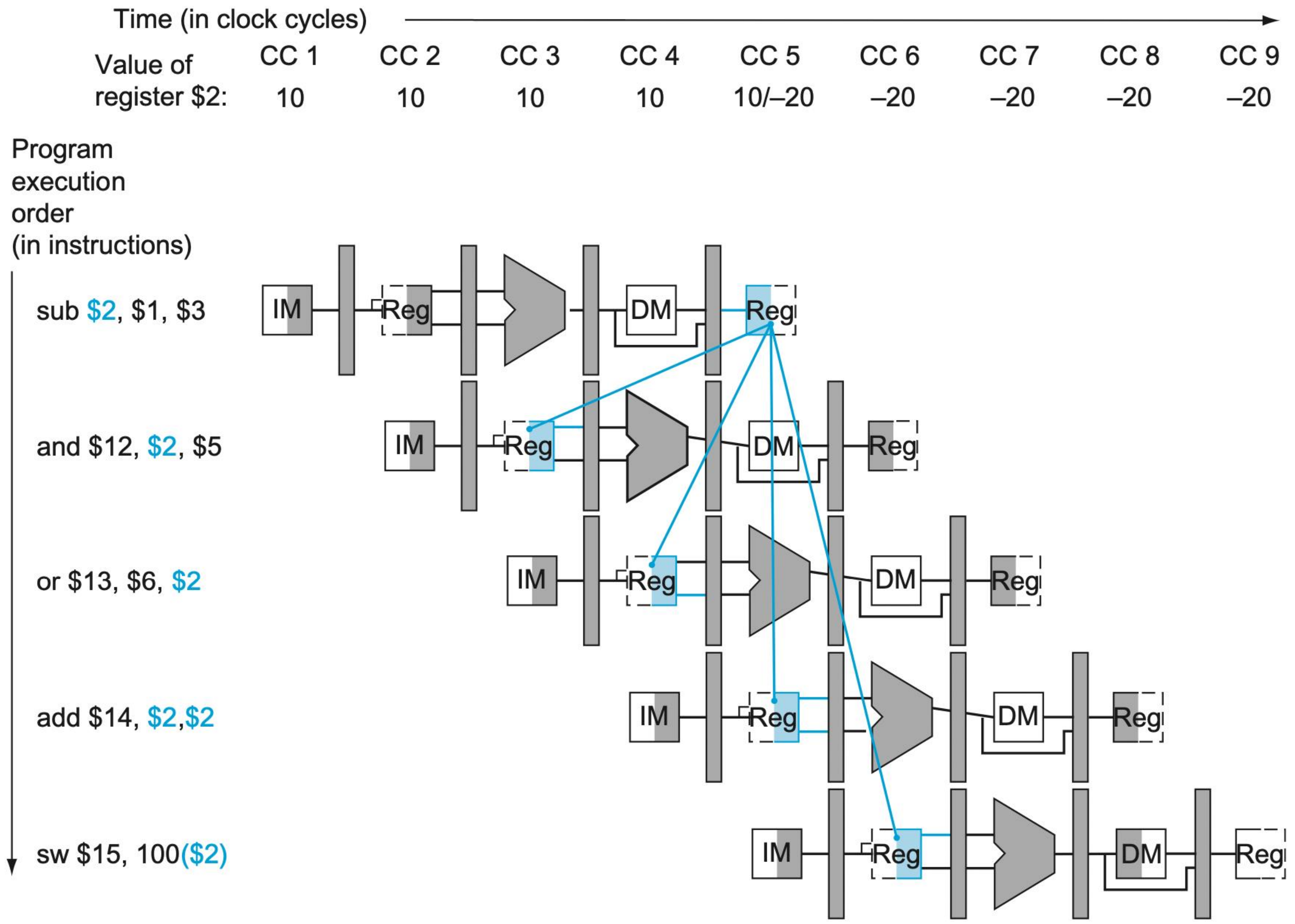
Computer Architecture (ENE1004)

Lec – 18: The Processor (Chapter 4) – 9

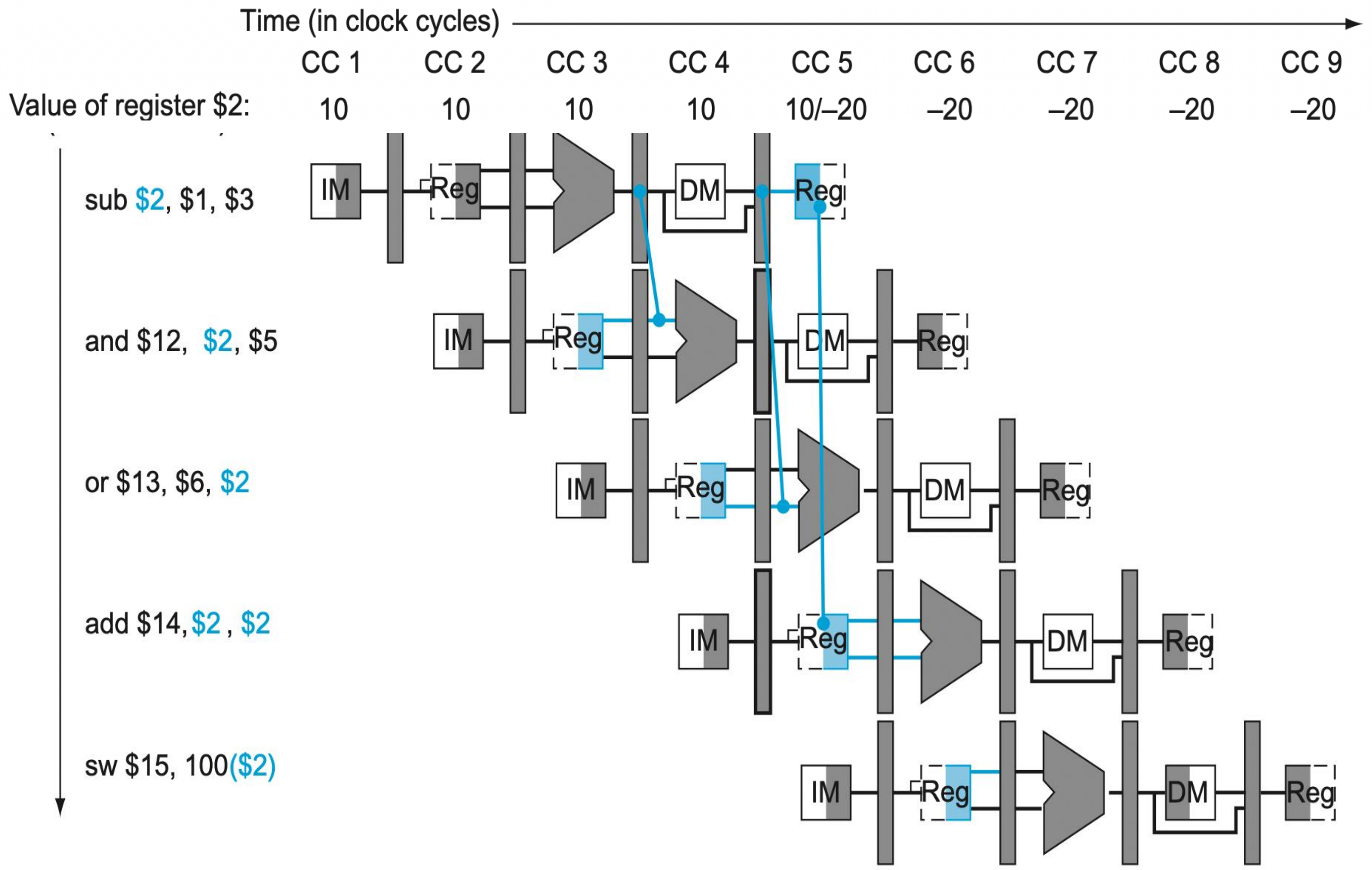
Notice

- The deadline of Assignment #1 has been extended to May 24 (Wed)
  - However, try to complete it as soon as possible
  - Add .text before the beginning of your code
- Assignment #2 has been announced
  - Tentative deadline: Jun. 11
  - Implement *shuffle32* as a function that calls *shuffle16* twice
  - Implement *shuffle16* as a function that calls *shuffle8* twice
  - Implement *shuffle8* as a function that calls *shuffle4* twice
  - Implement *shuffle4*
  - It must pass the 17 test cases
- May 29 (Mon) – No class
  - A short recorded video will be uploaded this week later for backup
  - It will provide a hint for Assignment #2

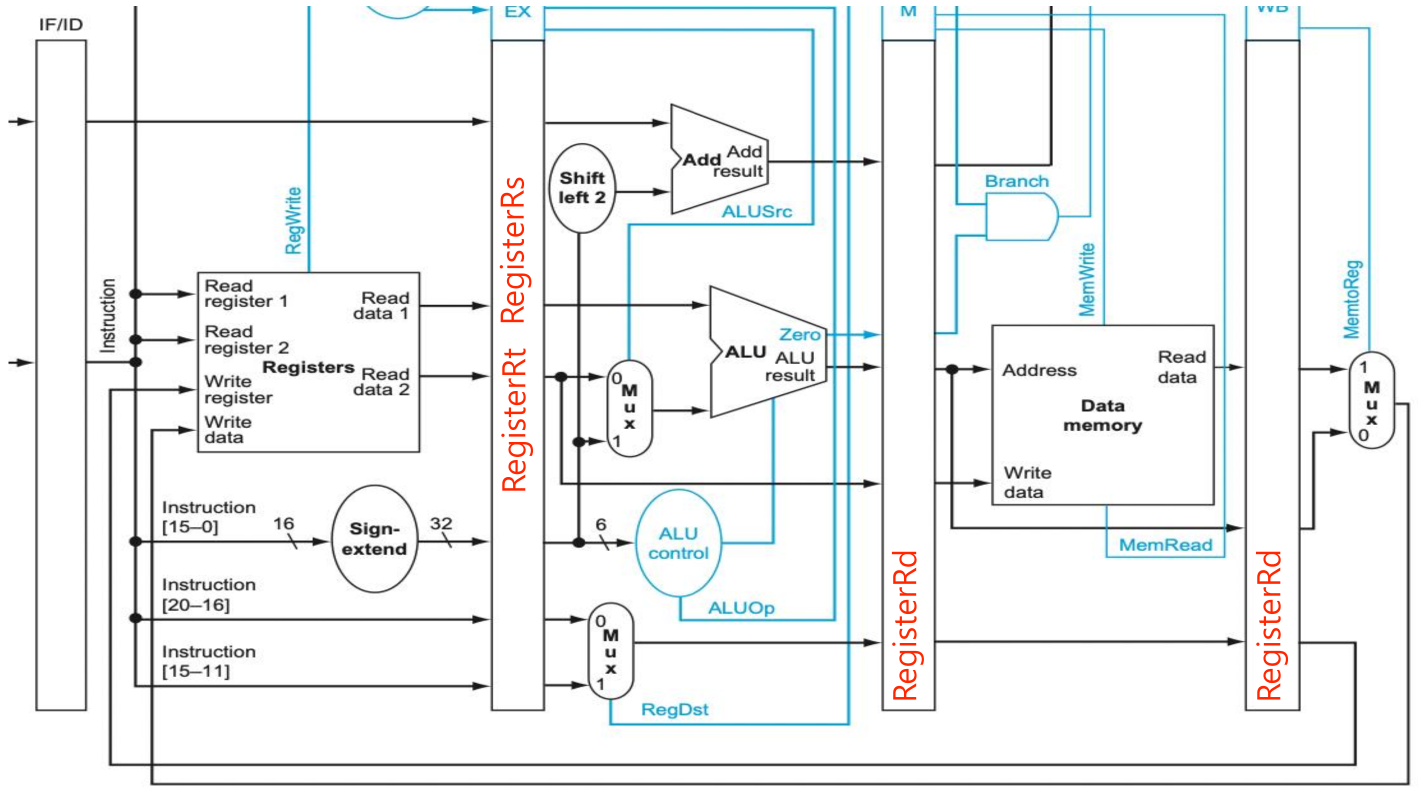
Data Hazard – RegWrite Scenarios



Data Forwarding (Solution) – RegWrite Scenarios



Data Hazard Detection – RegWrite Scenarios



- Four different cases where the data hazard occurs
  - *EX/MEM.RegisterRd == ID/EX.RegisterRs ?* (btw dest of 1st instruction & 1st source of 2nd instruction)
  - *EX/MEM.RegisterRd == ID/EX.RegisterRt ?* (btw dest of 1st instruction & 2nd source of 2nd instruction)
  - *MEM/WB.RegisterRd == ID/EX.RegisterRs ?* (btw dest of 1st instruction & 1st source of 3rd instruction)
  - *MEM/WB.RegisterRd == ID/EX.RegisterRt ?* (btw dest of 1st instruction & 2nd source of 3rd instruction)

Data Hazard Detection – RegWrite Scenarios

```
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
```

[For the 2nd instruction]

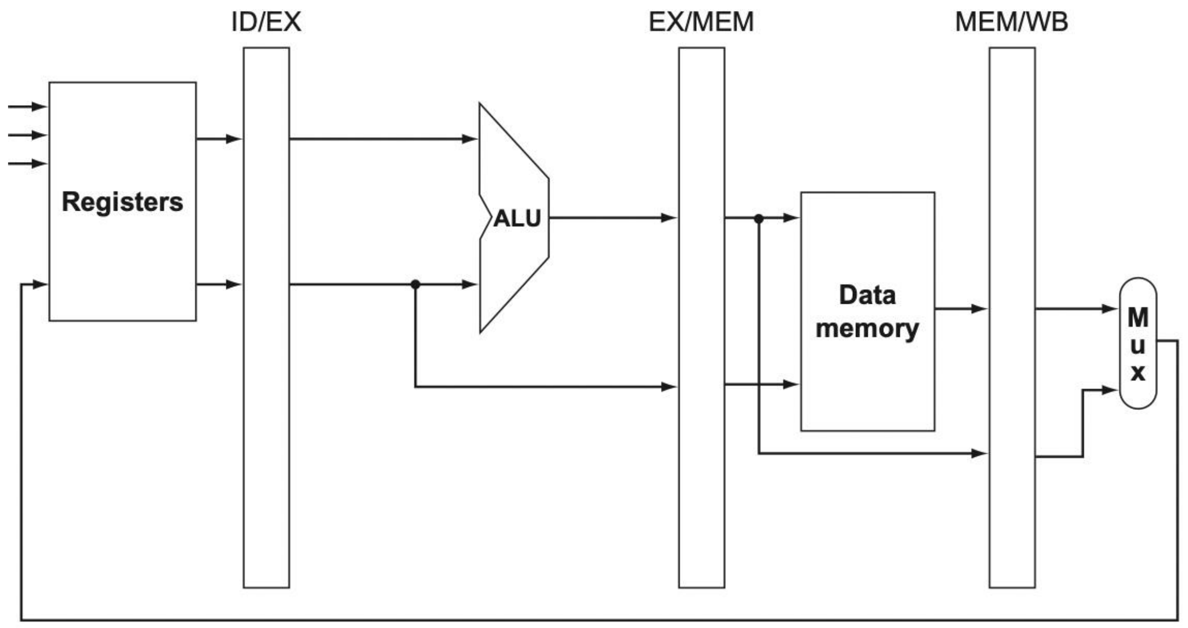
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠  0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠  0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
```

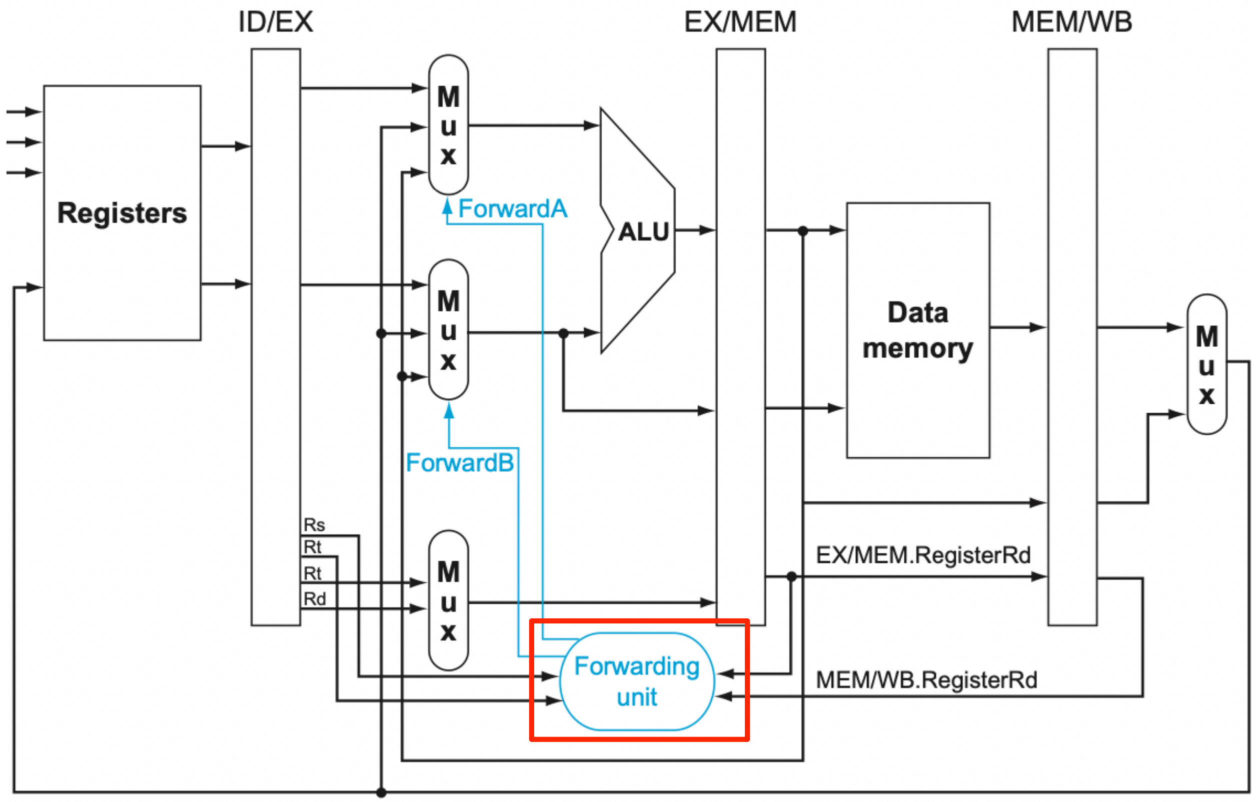
[For the 3rd instruction]

- (1) We need to check whether the first instruction writes register or not
  - If it does not write a register, we do not have to worry about the data hazard
  - Check EX/MEM.RegWrite and MEM/WB.RegWrite for the 2nd and 3rd instructions
- (2) We need to check whether the write register number is "0" or not
  - **sll $0, $1, 2**; this does not make any sense; but, programmers/compiler may make a mistake
  - In such a case, we do not have to worry about the data hazard
  - Check whether EX/MEM.RegisterRd and MEM/WB.RegisterRd for the 2nd and 3rd instructions
- (3) We need to check whether the write register number is used as a source register
  - Check the four conditions in the previous slide

Data Forwarding - Implementation (1)



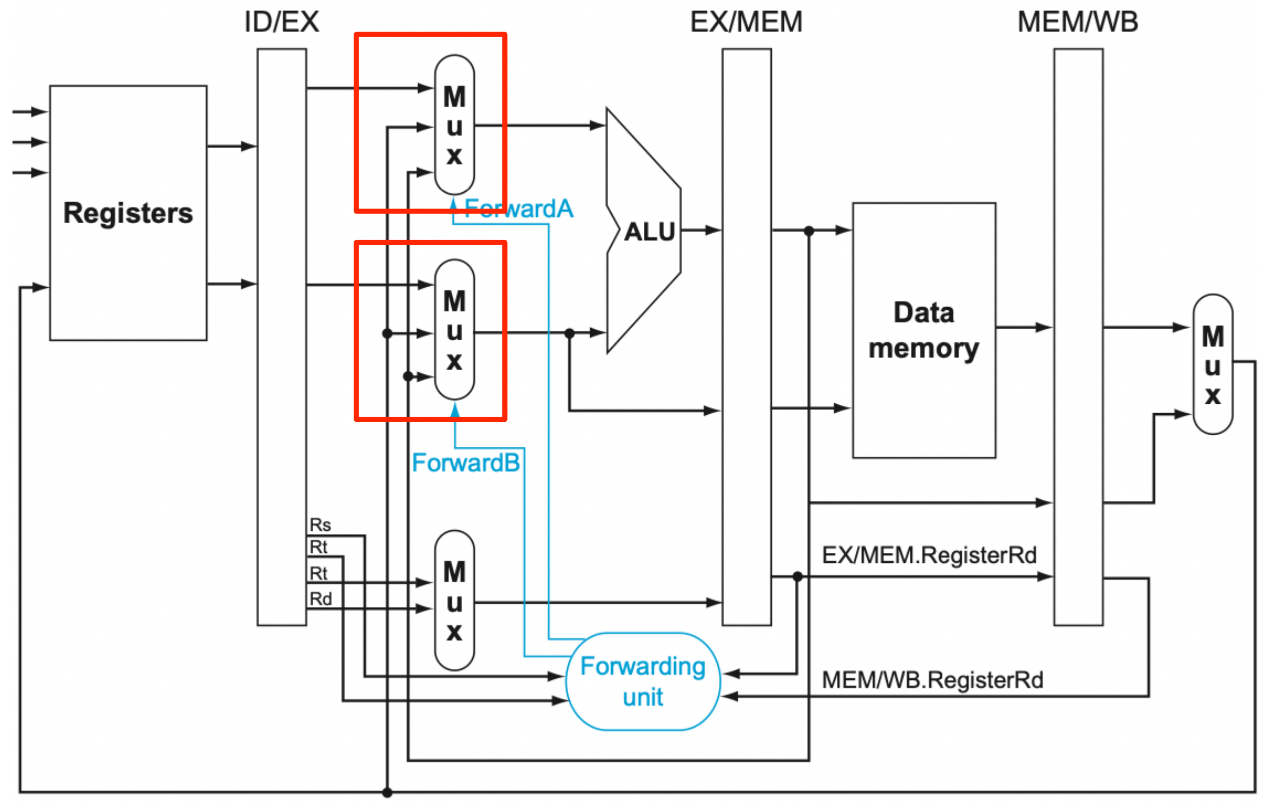
a. No forwarding



b. With forwarding

- We place the "forwarding unit"
- (i) It detects the data hazard based on the following values (inputs)
  - EX/MEM.RegisterRd, MEM/WB.RegisterRd, ID/EX.RegisterRs, ID/EX.RegisterRt
  - RegWrite signal value (this figure does not show it)

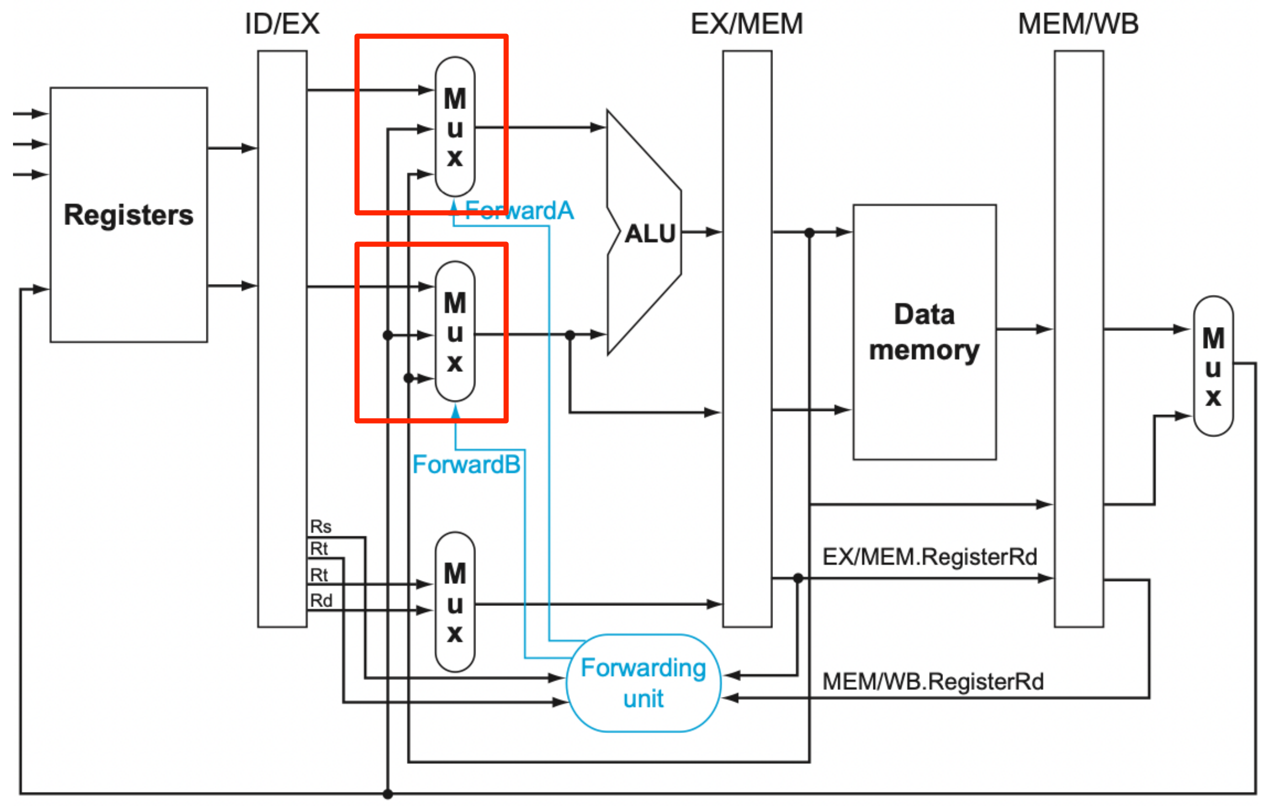
Data Forwarding - Implementation (2)



b. With forwarding

- We place "multiplexers" for two sources of ALU; the inputs of the multiplexers are
  - (1) from register file (or ID/EX register) - no data hazard
  - (2) from MEM/WB register – forwarding to the 3rd instruction
  - (3) from EX/MEM register – forwarding to the 2nd instruction

Data Forwarding - Implementation (3)

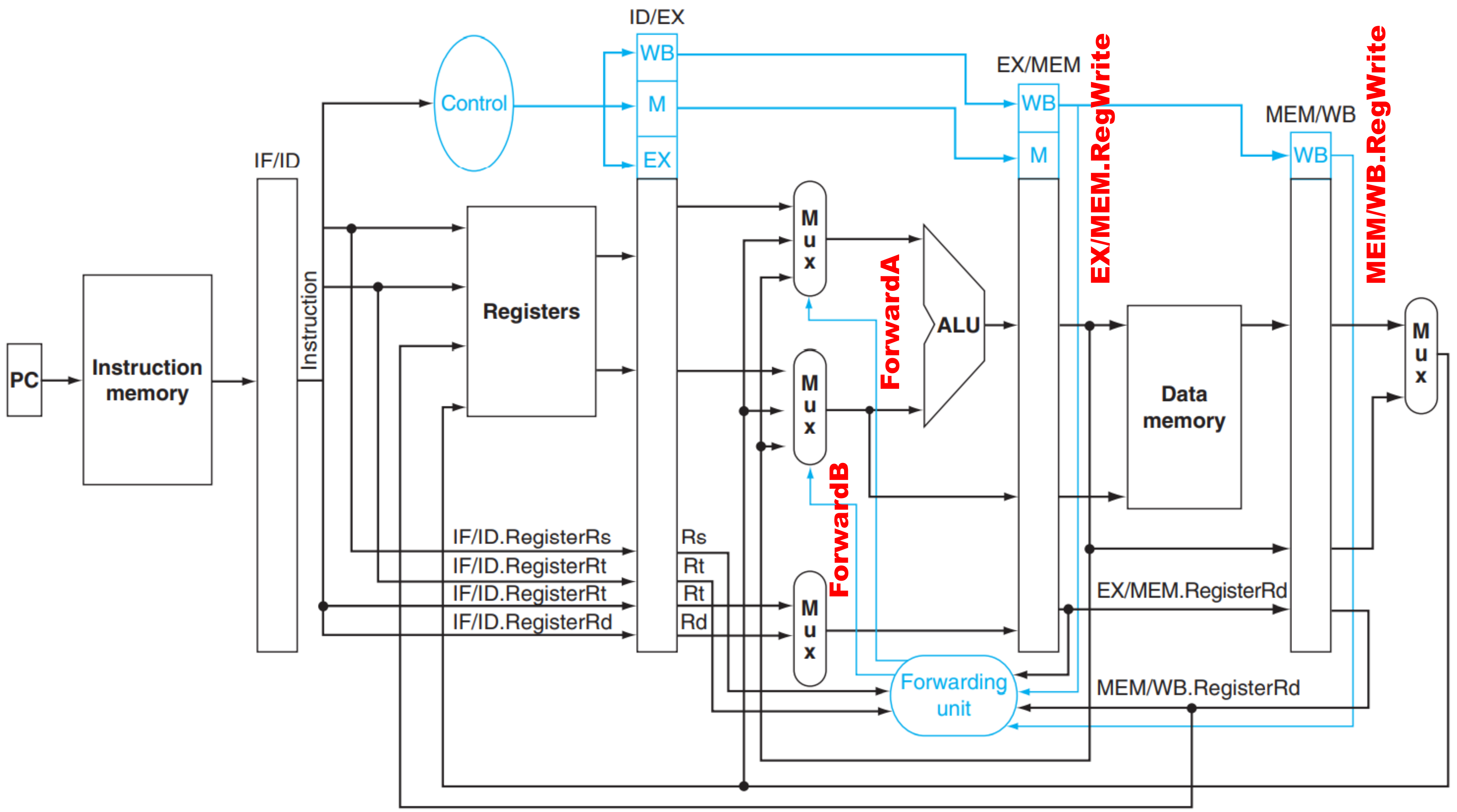


b. With forwarding

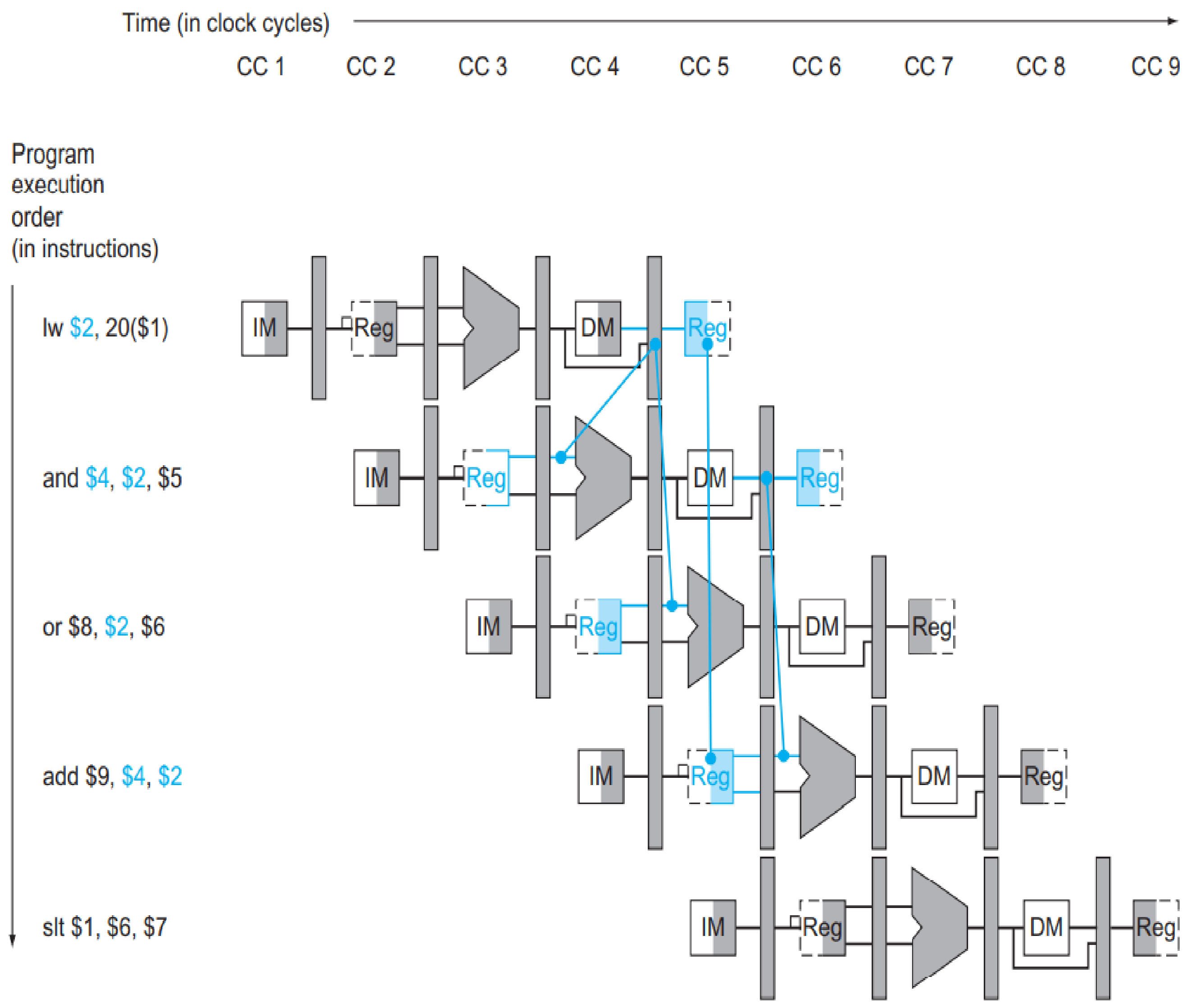
Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX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

- (ii) The forwarding unit determines one of the three inputs using 2-bit control signal
  - **ForwardA** & **ForwardB** determine the 1st and the 2nd source of the ALU
  - “00” selects the 1st input (no data hazard)
  - “01” selects the 2nd input (forwarding from MEM/WB for the 3rd instruction)
  - “10” selects the 3rd input (forwarding from EX/MEM for the 2nd instruction)

Data Forwarding - Implementation (4)

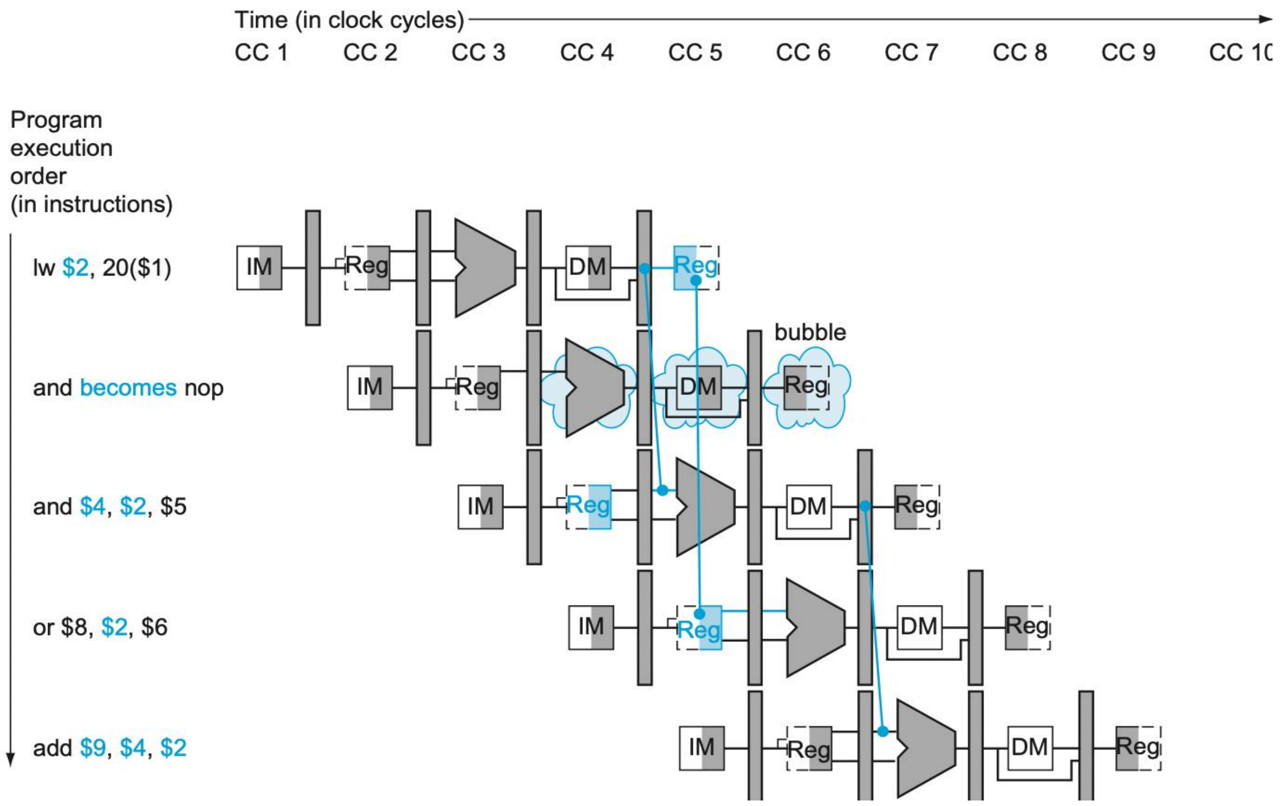


Data Hazard – Stall Scenarios



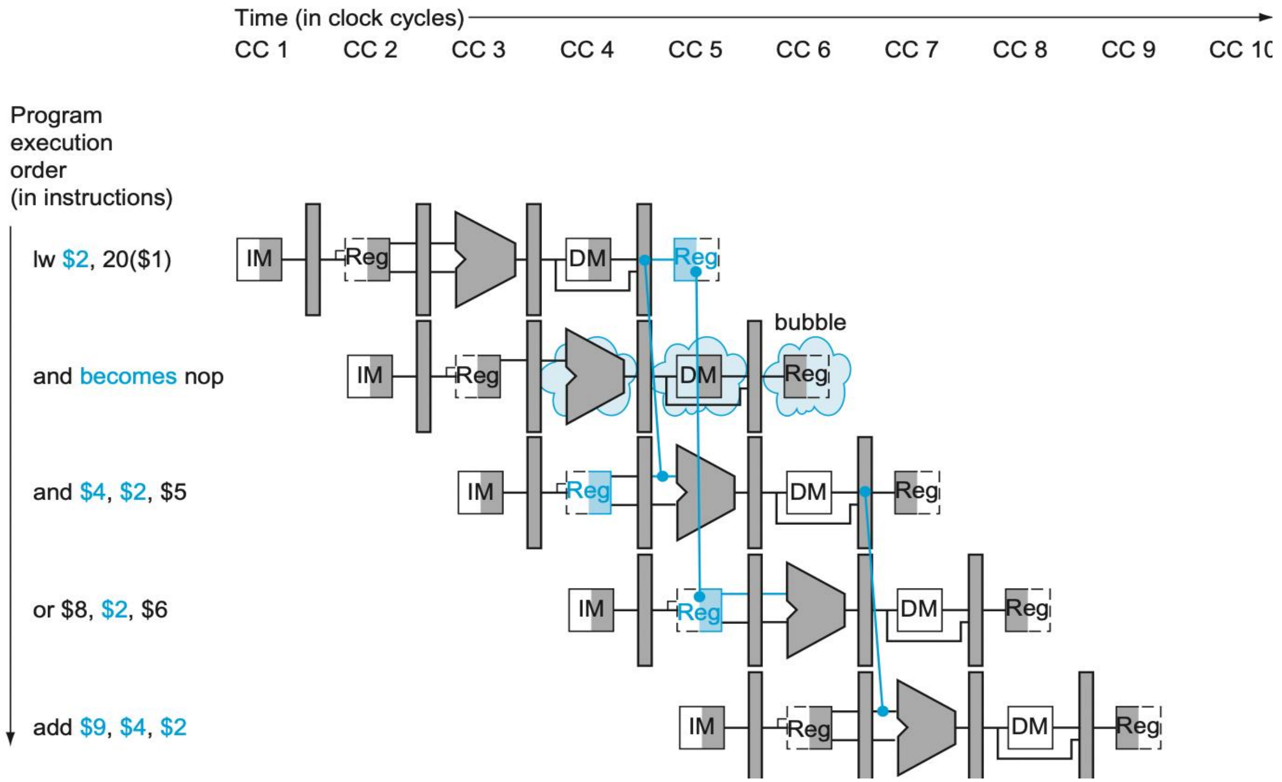
- There is a case where data forwarding cannot resolve the data hazard
  - Destination register of **lw** instruction
  - Next instruction attempts read the register
  - **$2** register in the example
- **lw $2, 20($1)**
  - The data read from the data memory in MEM stage is stored in MEM/WB register
- **and $4, $2, $5**
  - The data read from the data memory is needed at least in EX stage
- The data can be forwarded from *MEM/WB register* to *Rs port of the ALU*; but, it is too late
  - How can we handle this problem?

Stall (Solution) – Stall Scenarios (1)



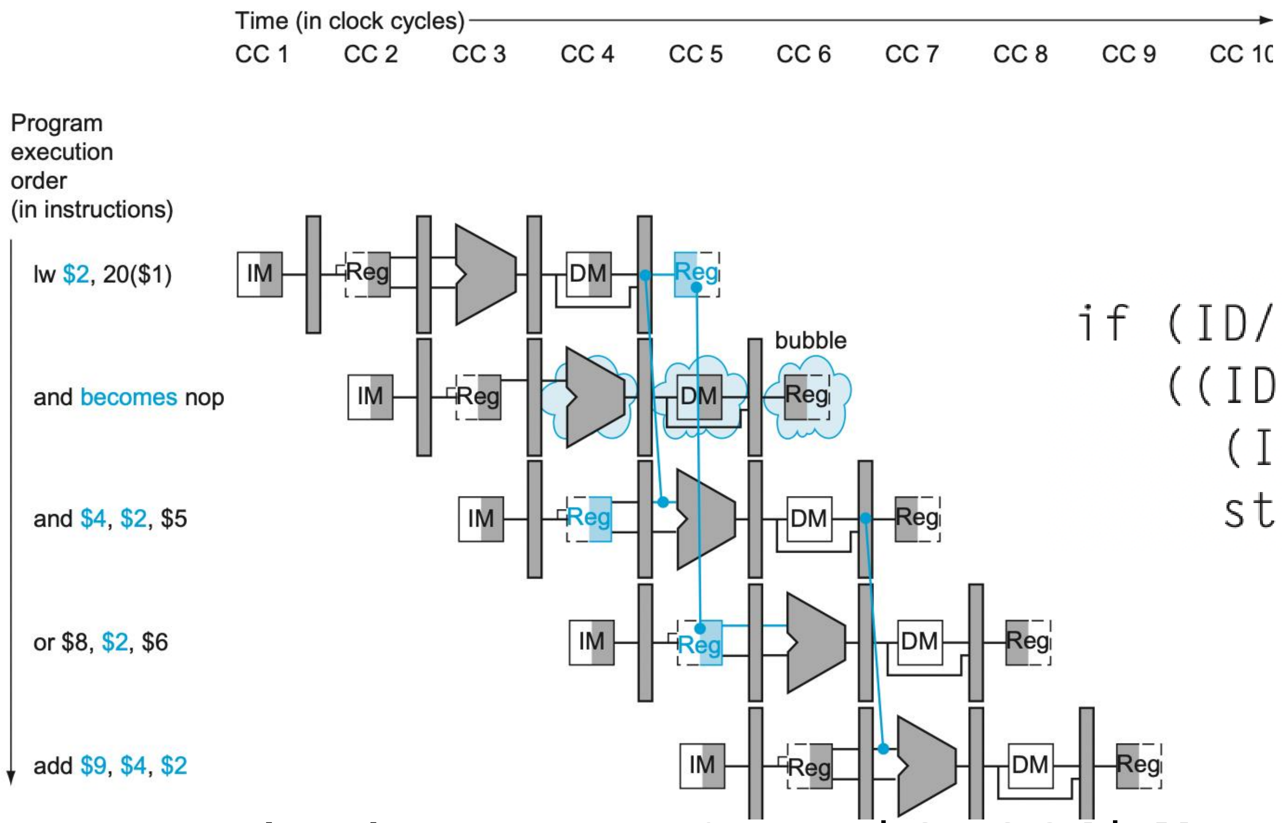
- The only way to resolve this is to stall the pipeline (wasting time by doing nothing)
  - (1) We prevent **and** (in ID stage) & **or** (in IF stage) from making progress for one CC
  - (2) We insert **nop** (an instruction that does nothing) into the pipeline and let it make progress
- After waiting for a CC, data forwarding (from MEM/WB to Rs of ALU) can resolve this

Stall (Solution) – Stall Scenarios (2)



- (1) We prevent **and** (in ID stage) & **or** (in IF stage) from making progress for one CC
  - If we prevent IF/ID pipeline register from changing, **and** will enter the ID stage again
  - If we prevent PC from changing, **or** will enter the IF stage again
- (2) We insert **nop** into the pipeline (see bubble) and let it make progress
  - If we set the nine control signals to "0", registers or memory are not written

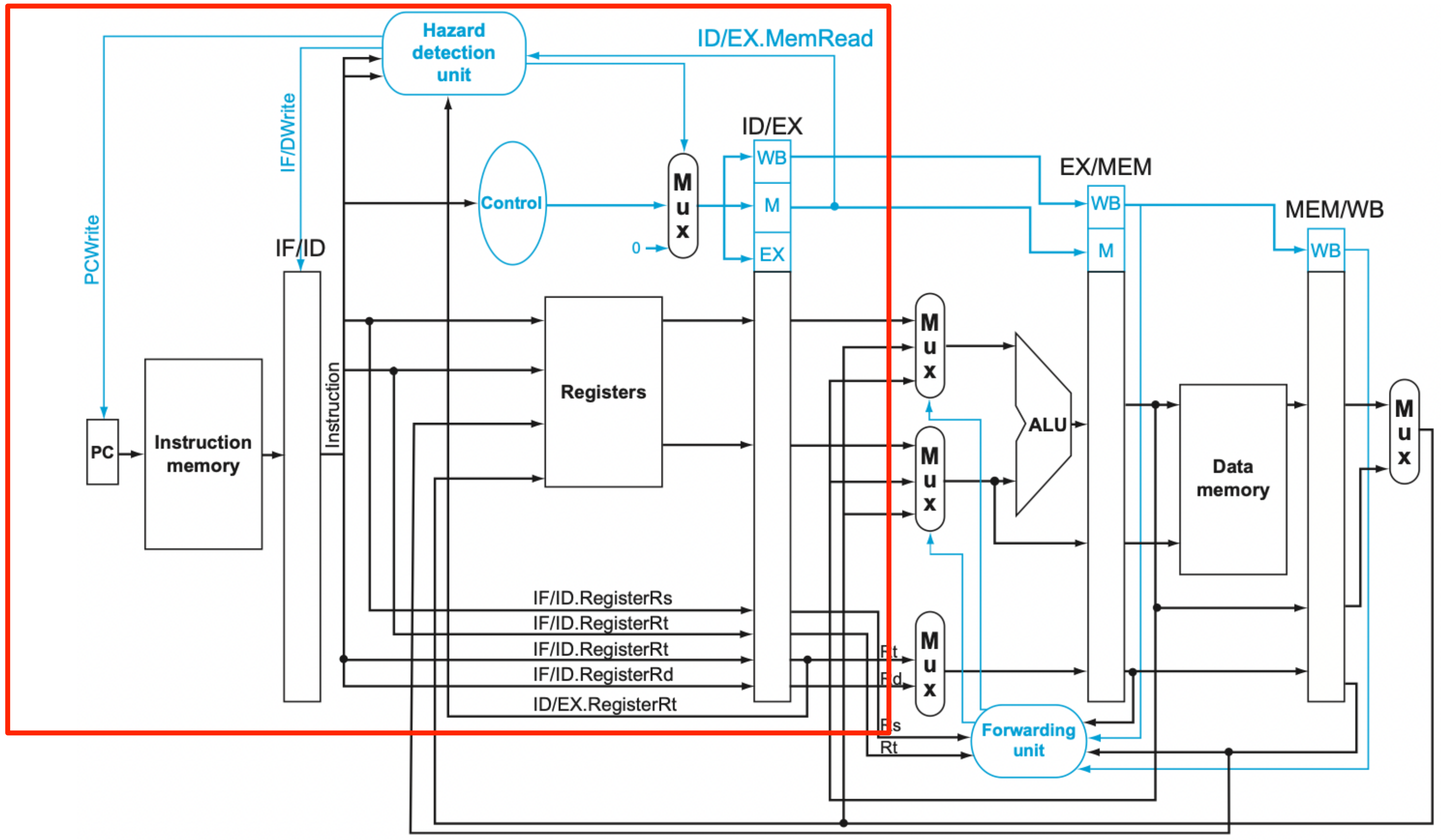
Data Hazard Detection for Stall



```
if (ID/EX.MemRead and
   ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
     (ID/EX.RegisterRt = IF/ID.RegisterRt)))
     stall the pipeline
```

- (1) Is it a load instruction? (**lw $2, 20($1)**)
  - *ID/EX.MemRead* == 1?
- (2) Is destination of the load instruction equal to a source of the next instruction? (**$2**)
  - *ID/EX.RegisterRt* == *IF/ID.RegisterRs* ?
  - *ID/EX.RegisterRt* == *IF/ID.RegsiterRt* ?

Stall - Implementation



- To emphasize hazard-related components, branch/sign-extension units are missing
- Hazard detection unit
  - Evaluates the conditions in the previous slide (ID/EX.MemRead, ID/EX.RegisterRt, IF/ID.RegisterRs/Rt)
  - (If it needs a stall) prevents PC & IF/ID register from changing (PCWrite=0 , IF/IDWrite=0)
  - (If it needs a stall) selects “0” to all 9 control signals