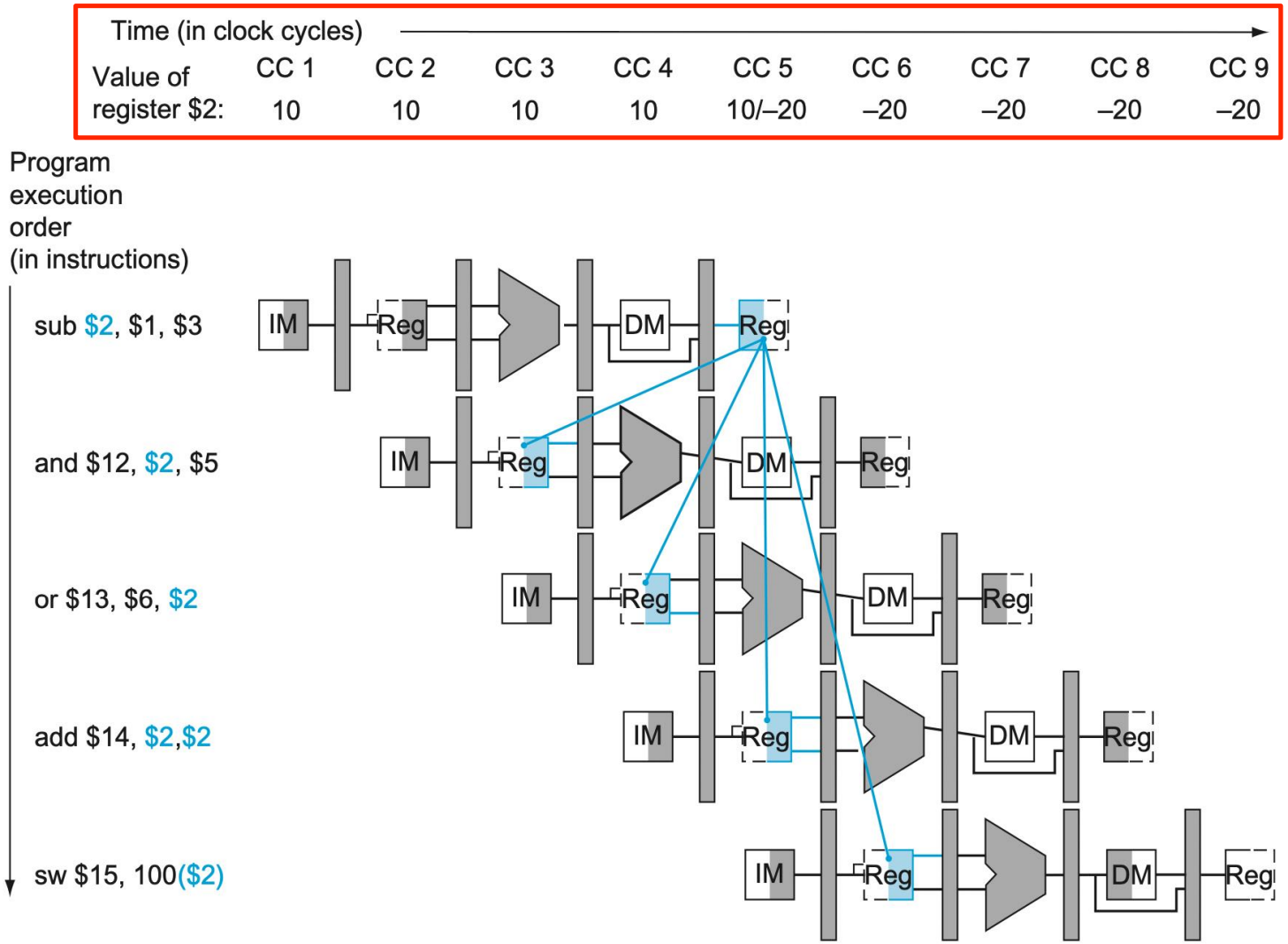
Computer Architecture
(ENE1004)

Lec – 17: The Processor (Chapter 4) – 8

Notice

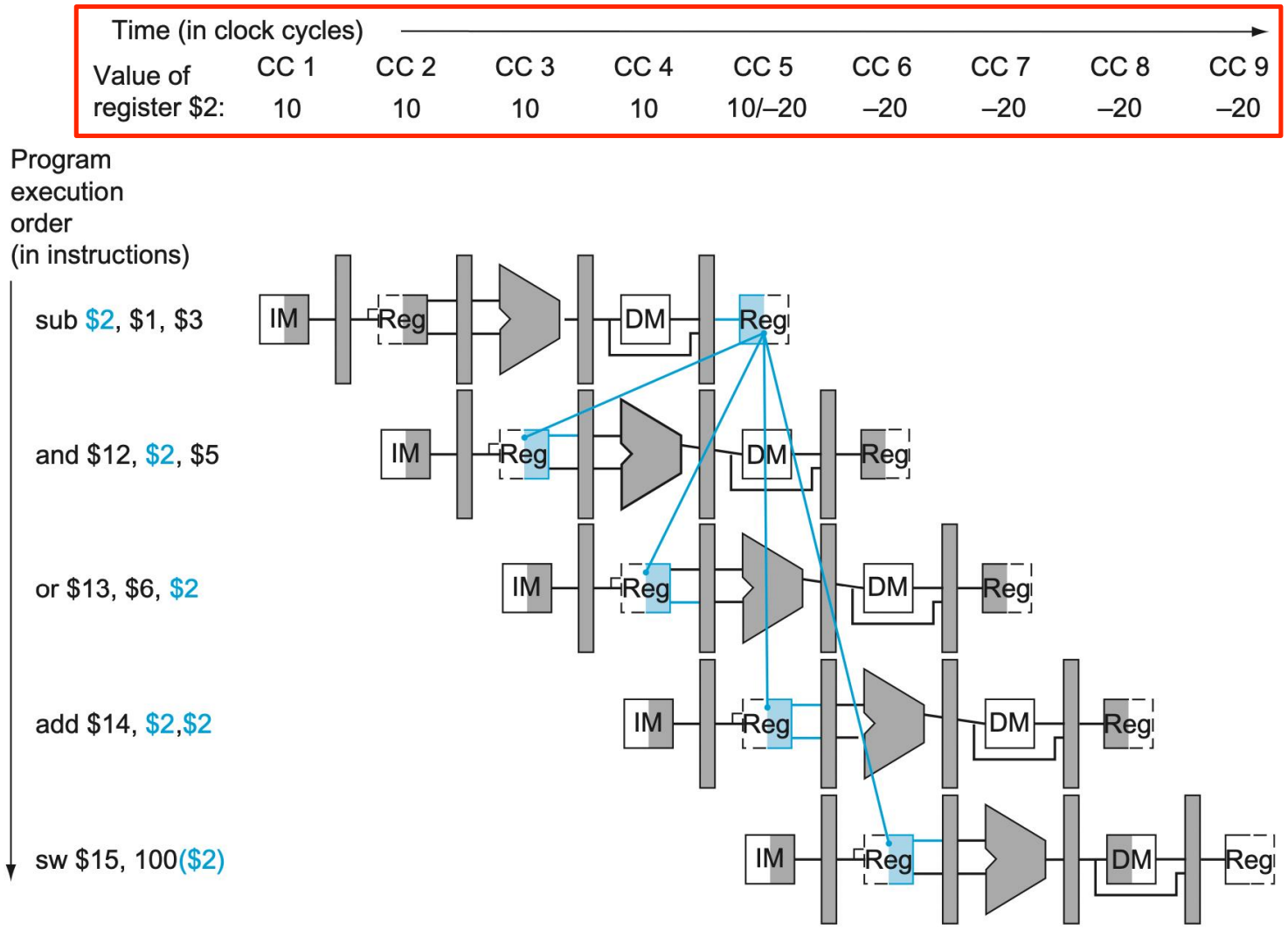
- The deadline of Assignment #1 has been extended to May 24 (Wed)
  - However, try to complete it as soon as possible
  - Add .text before the beginning of your code
- Assignment #2 has been announced
  - Tentative deadline: Jun. 11
  - Implement shuffle32 as a function that calls shuffle16 twice
  - Implement shuffle16 as a function that calls shuffle8 twice
  - Implement shuffle 8 as a function that calls shuffle4 twice
  - Implement shuffle4
  - It must pass the 17 test cases
- May 29 – No class
  - A recorded video will be uploaded later for backup

Data Hazard - An Example Scenario (1)



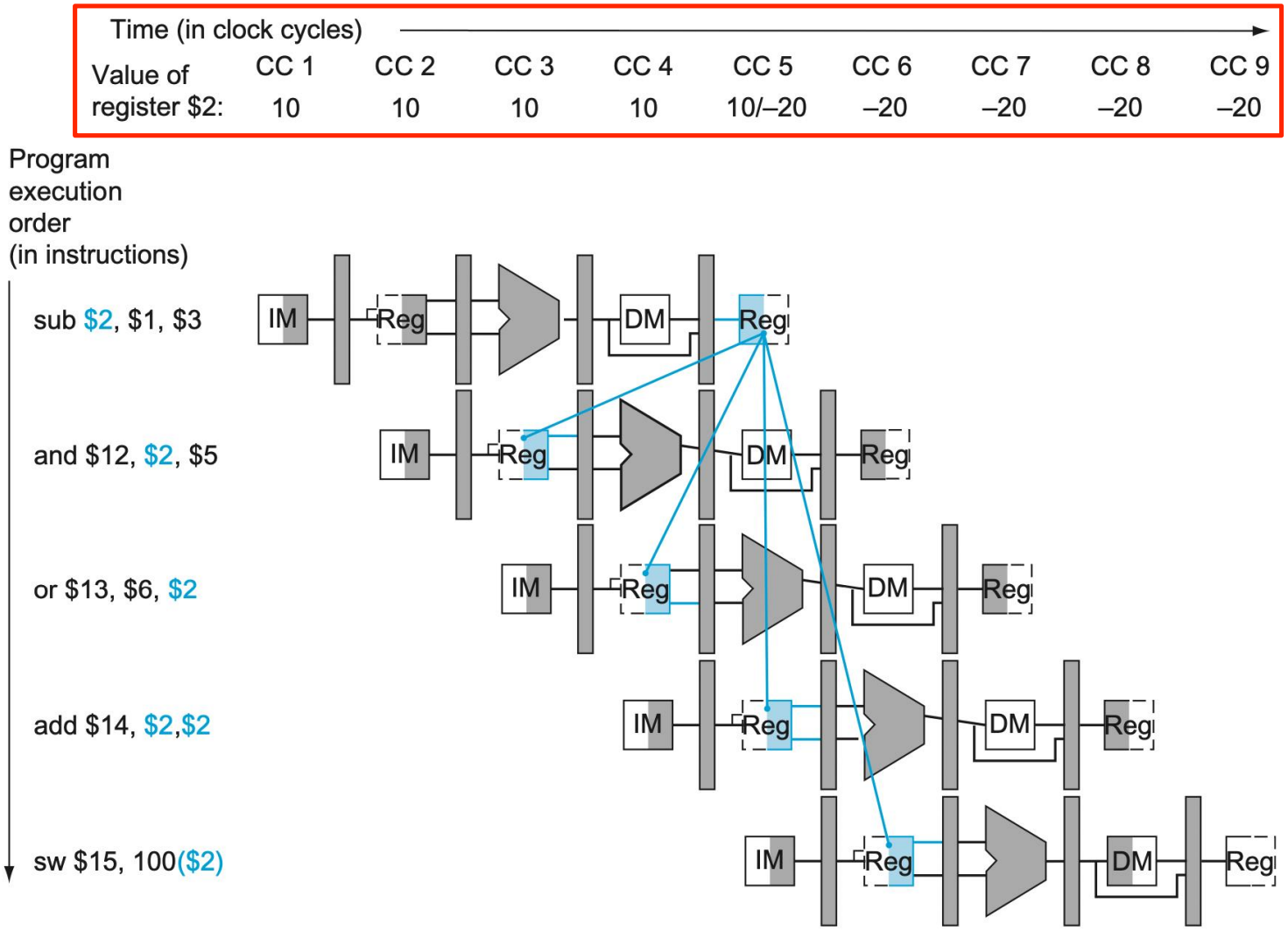
- Figure
  - Clock cycle (CC): left —> right
  - Instructions: top —> down
- Instructions scenario: The destination register of 1st instruction (**$2**) is the source registers of following instructions
- Assumption
  - **$2** holds "10" at first
  - **$2** is expected to hold "-20" after executing 1st instruction (sub $2,$1,$3)
- Let us focus on the value of **$2**
  - 1st instruction writes "-20" into **$2** in WB stage in CC5
  - So, **$2** holds "10" till CC4; **$2** holds "-20" from CC5

Data Hazard - An Example Scenario (2)



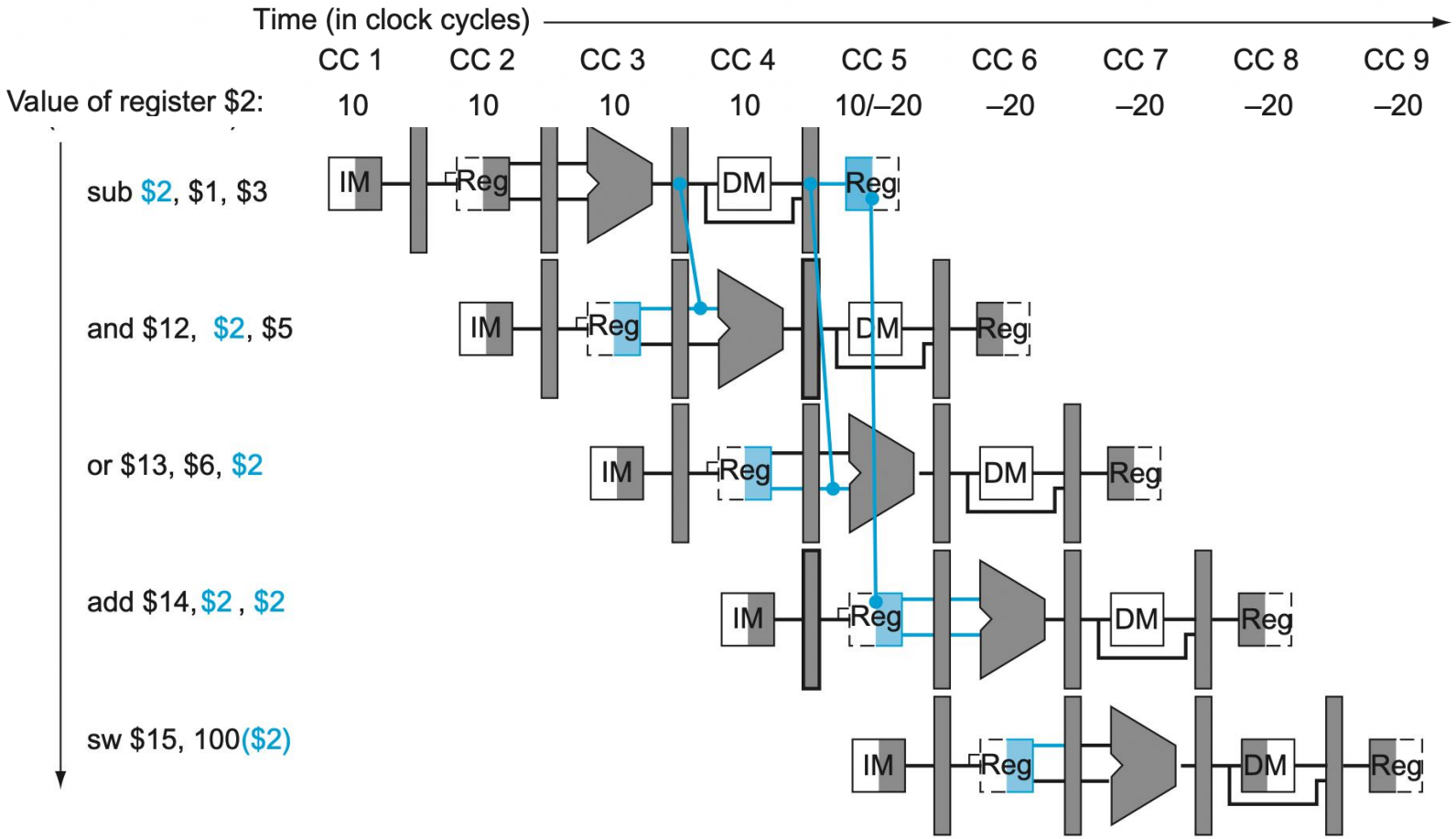
- **and $12, $2, $5**
  - The 2nd instruction expects to read the new value "-20" from **$2** in the ID stage at CC3
  - However, **$2** holds the old value "10" at CC3
- **or $13, $6, $2**
  - The 3rd instruction expects to read the new value "-20" from **$2** in the ID stage at CC4
  - However, **$2** holds the old value "10" at CC4
- We call these "data hazard"
  - A destination register of an instruction is read by following instructions as source registers

Data Hazard - An Example Scenario (3)



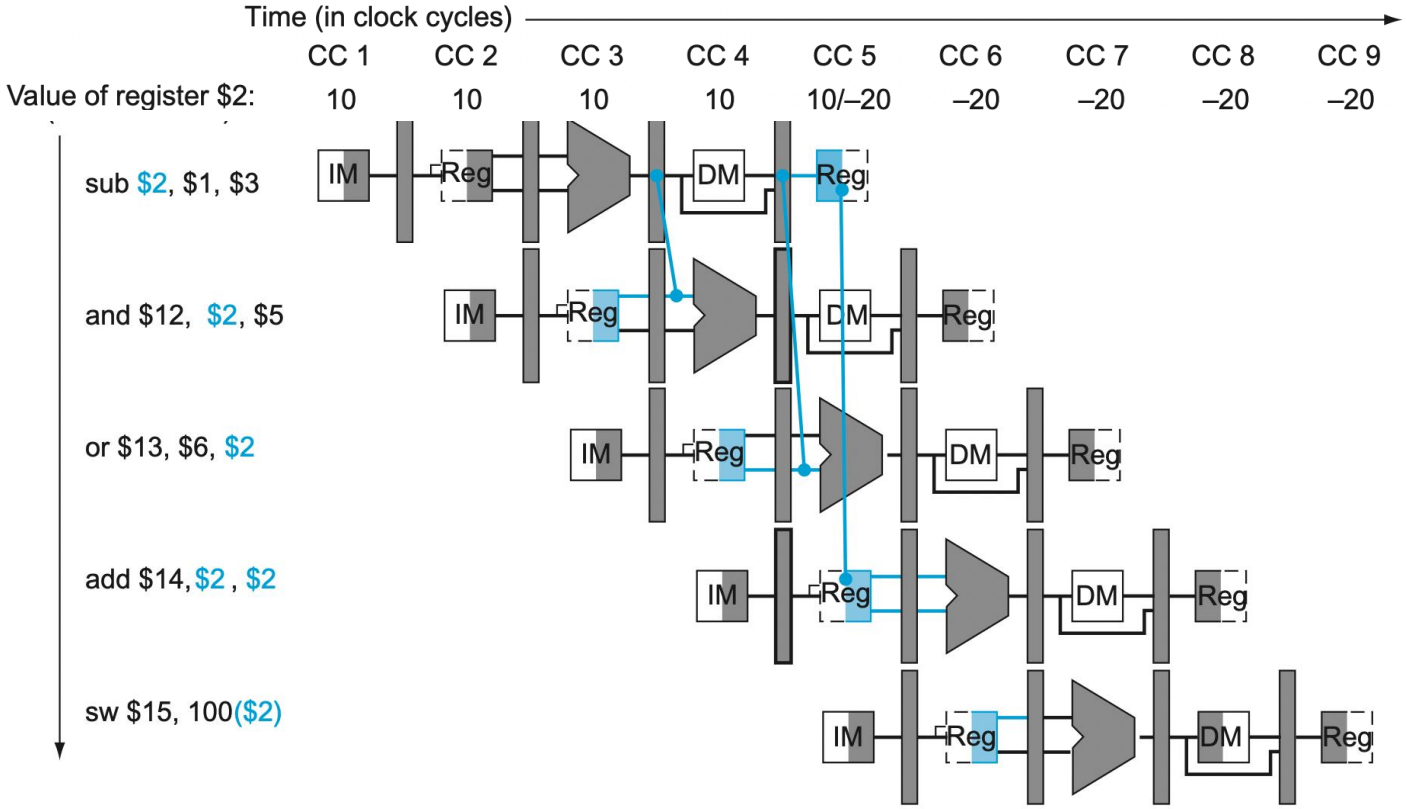
- **add $14, $2, $2**
  - The 3rd instruction expects to read the new value "-20" from **$2** in the ID stage at CC5
  - The new value "-20" is written into **$2** in the first half of CC5 by **sub**, which can be read by **add** in the second half of CC5
  - This is NOT a data hazard
- **sw $15, 100($2)**
  - The 4th instruction expects to read the new value "-20" from **$2** in the ID stage at CC6
  - The new value "-20" is written into **$2** at CC5, which can be read at CC6
  - There is NO problem at all

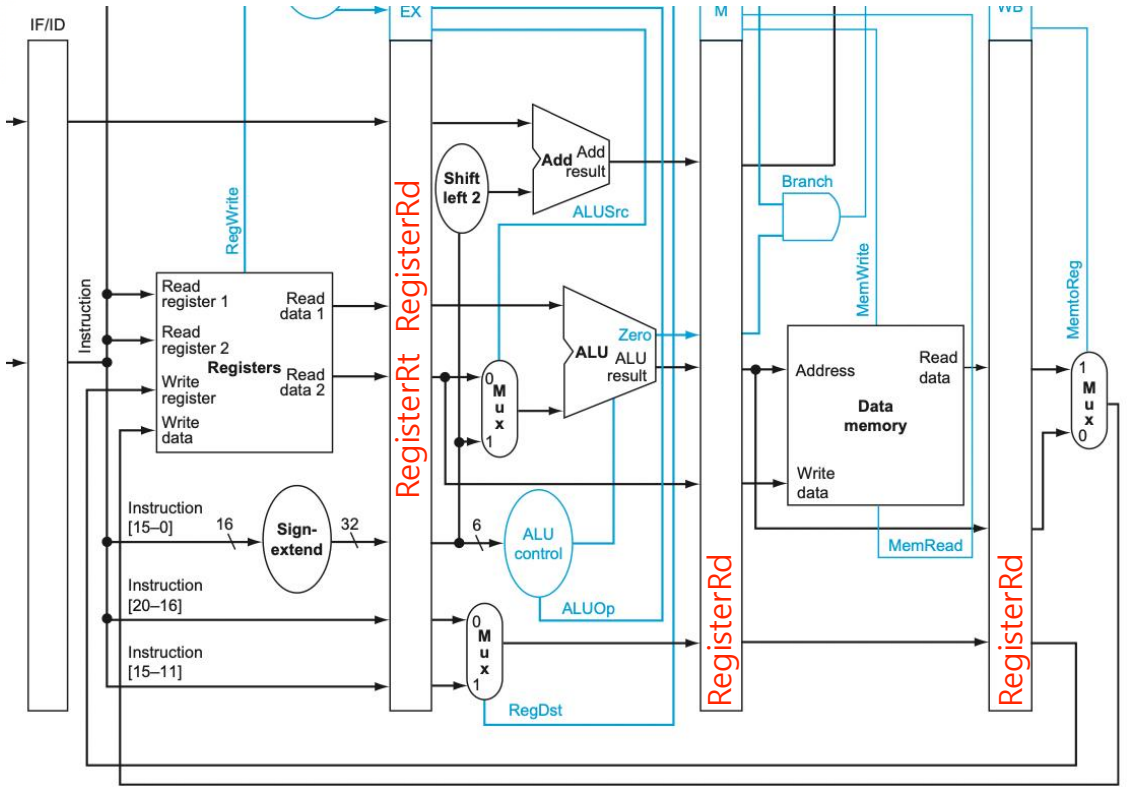
Data Forwarding (Solution) - An Example Scenario



- The new value "-20" is generated by **ALU** at the end of EX stage at CC3 (EX/MEM register)
- How about sending (forwarding) the new value to the sources of ALU?
  - For **and**, we can send the value from *EX/MEM register* to ***Rs (1st) port of the ALU*** at CC4
  - For **or**, we can send the value from *MEM/WB register* to ***Rt*** *(2nd) port of the ALU* at CC5

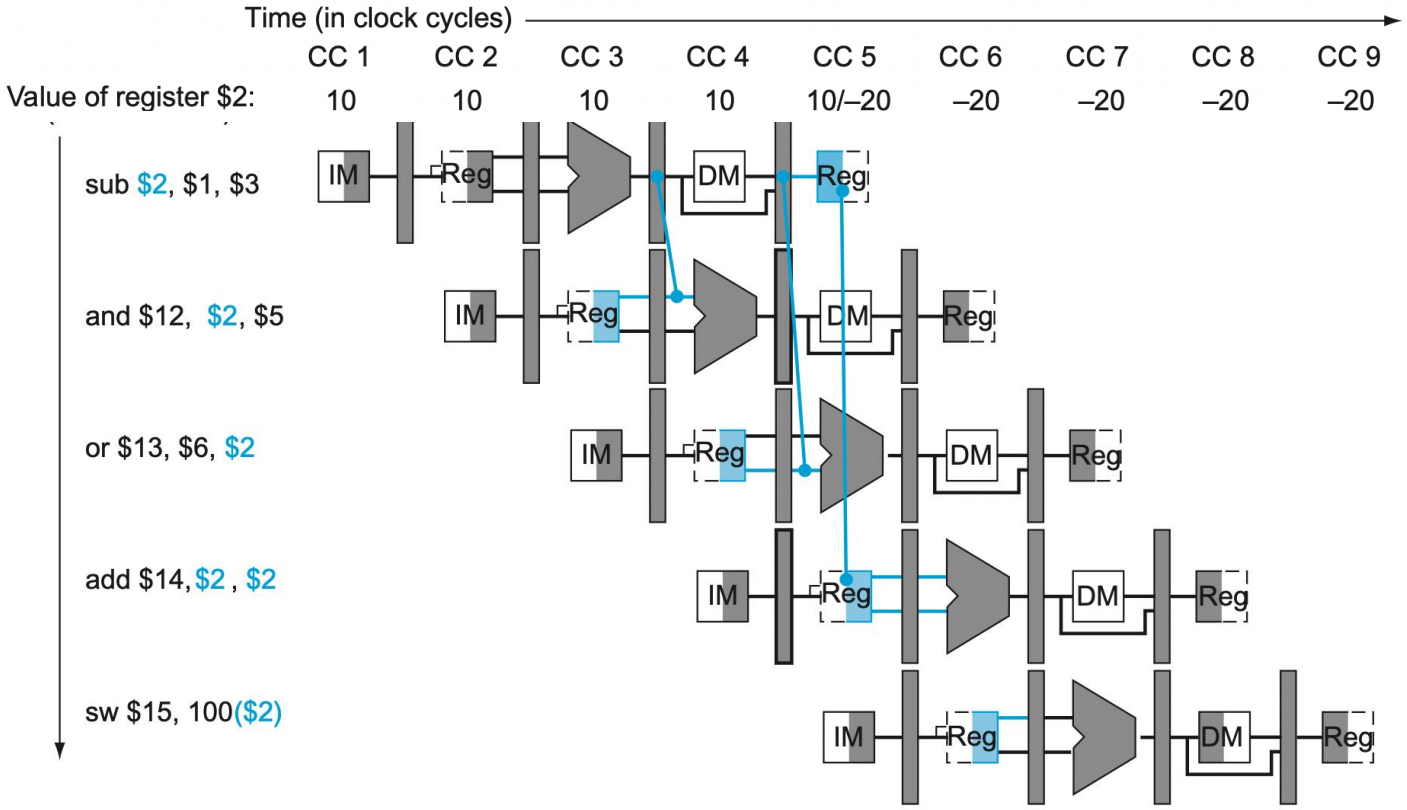
Data Hazard Detection - An Example Scenario

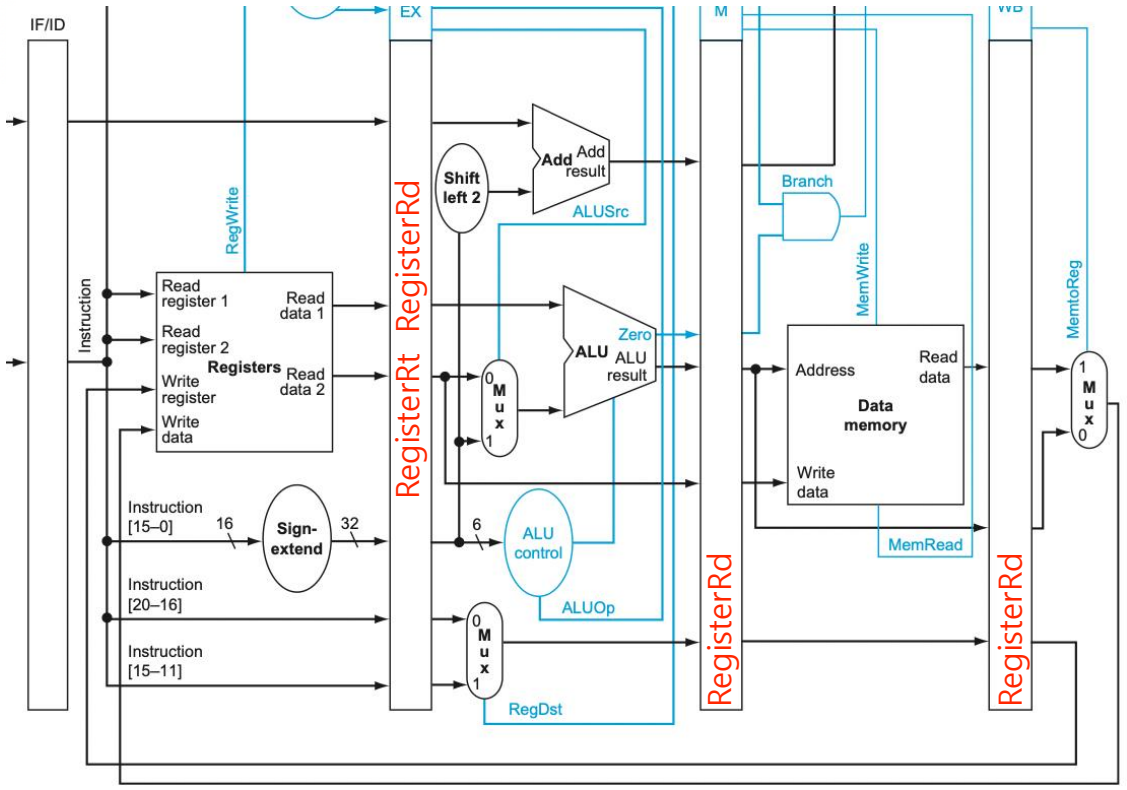




- Datapath should be able to detect when data hazard occurs (and forwarding is performed)
- We need to check whether the destination register (**$2**) is used as a source register (**$2**)
  - For **and**, is *EX/MEM.RegisterRd* (of **sub**) equal to *ID/EX.RegisterRs* (of **and**)?
  - For **or**, is *MEM/WB.RegisterRd* (of **sub**) equal to *ID/EX.RegisterRt* (of **or**)?

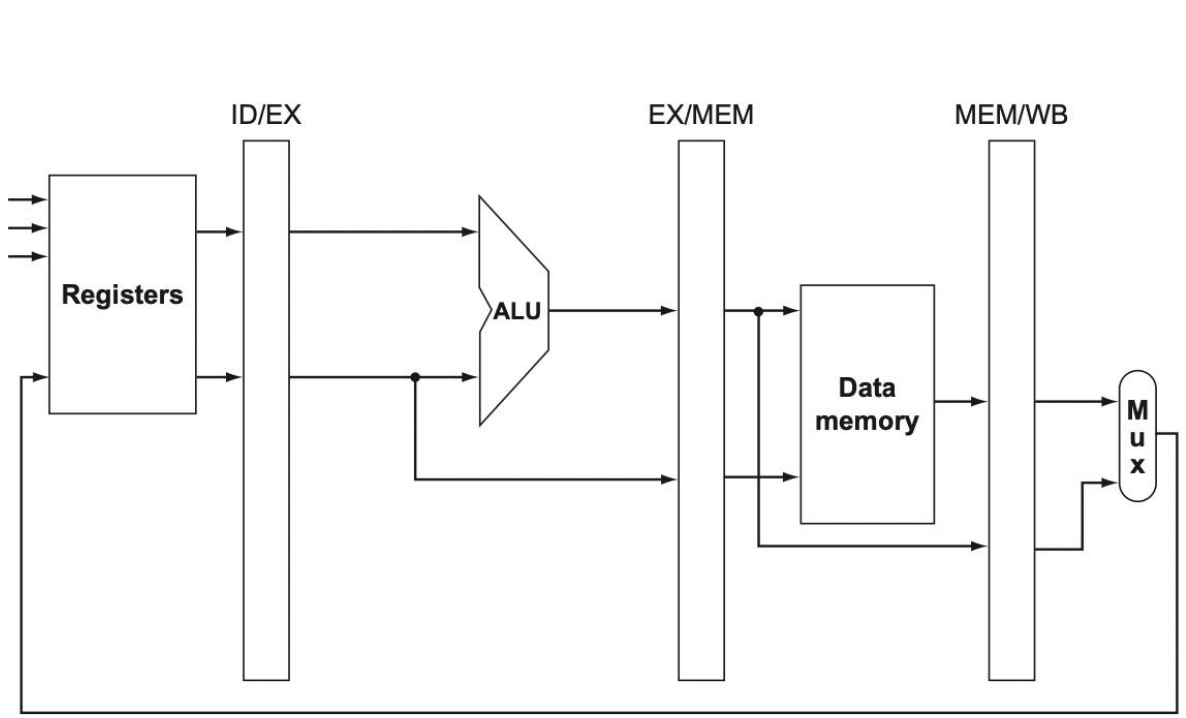
Data Hazard Detection - Generalization



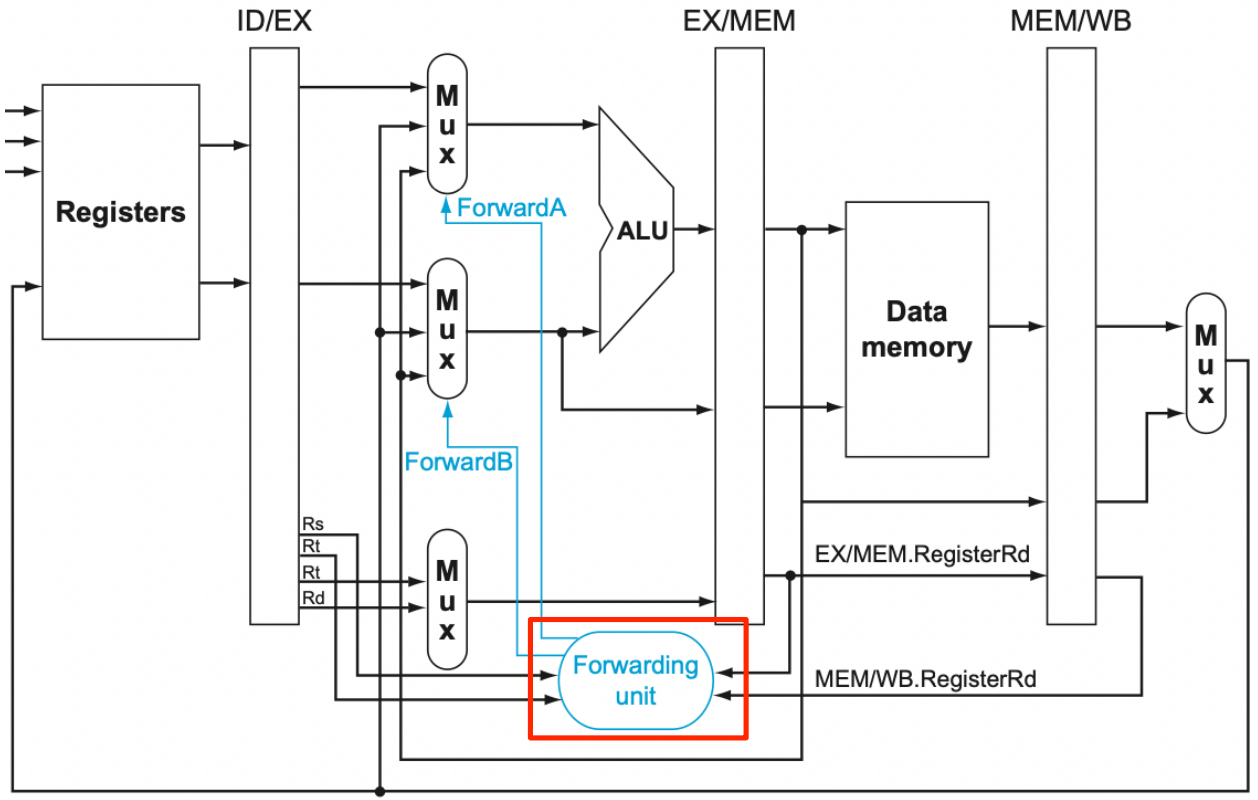


- Four different cases where the data hazards occur
  - *EX/MEM.RegisterRd* = *ID/EX.RegisterRs* (btw 1st and 2nd instructions; 1st source of 2nd instruction)
  - *EX/MEM.RegisterRd* = *ID/EX.RegisterRt* (btw 1st and 2nd instructions; 2nd source of 2nd instruction)
  - *MEM/WB.RegisterRd* = *ID/EX.RegisterRs* (btw 1st and 3rd instructions; 1st source of 3rd instruction)
  - *MEM/WB.RegisterRd* = *ID/EX.RegisterRt* (btw 1st and 3rd instructions; 2nd source of 3rd instruction)

Data Forwarding - Implementation (1)



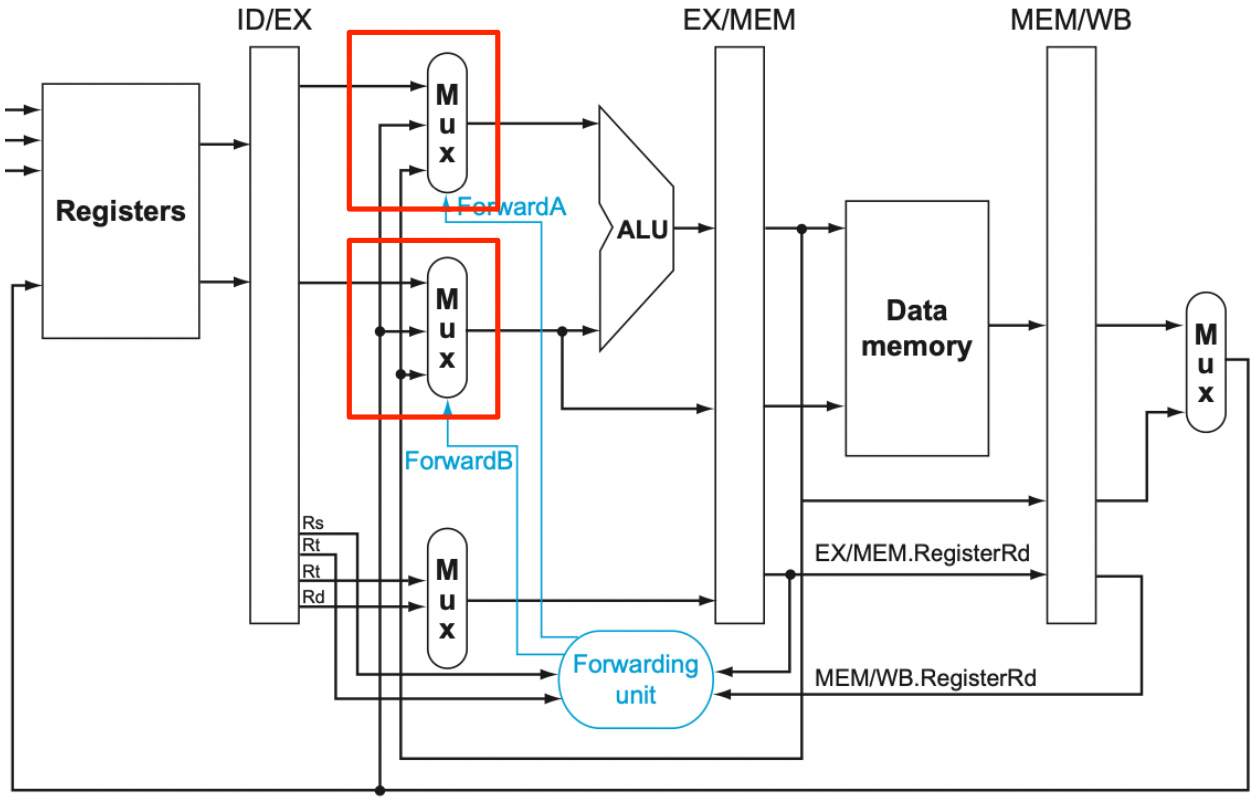
a. No forwarding



b. With forwarding

- We place the forwarding unit, which detects data hazard using the four inputs
  - EX/MEM.RegisterRd
  - MEM/WB.RegisterRd
  - ID/EX.RegisterRs
  - ID/EX.RegisterRd

Data Forwarding - Implementation (2)

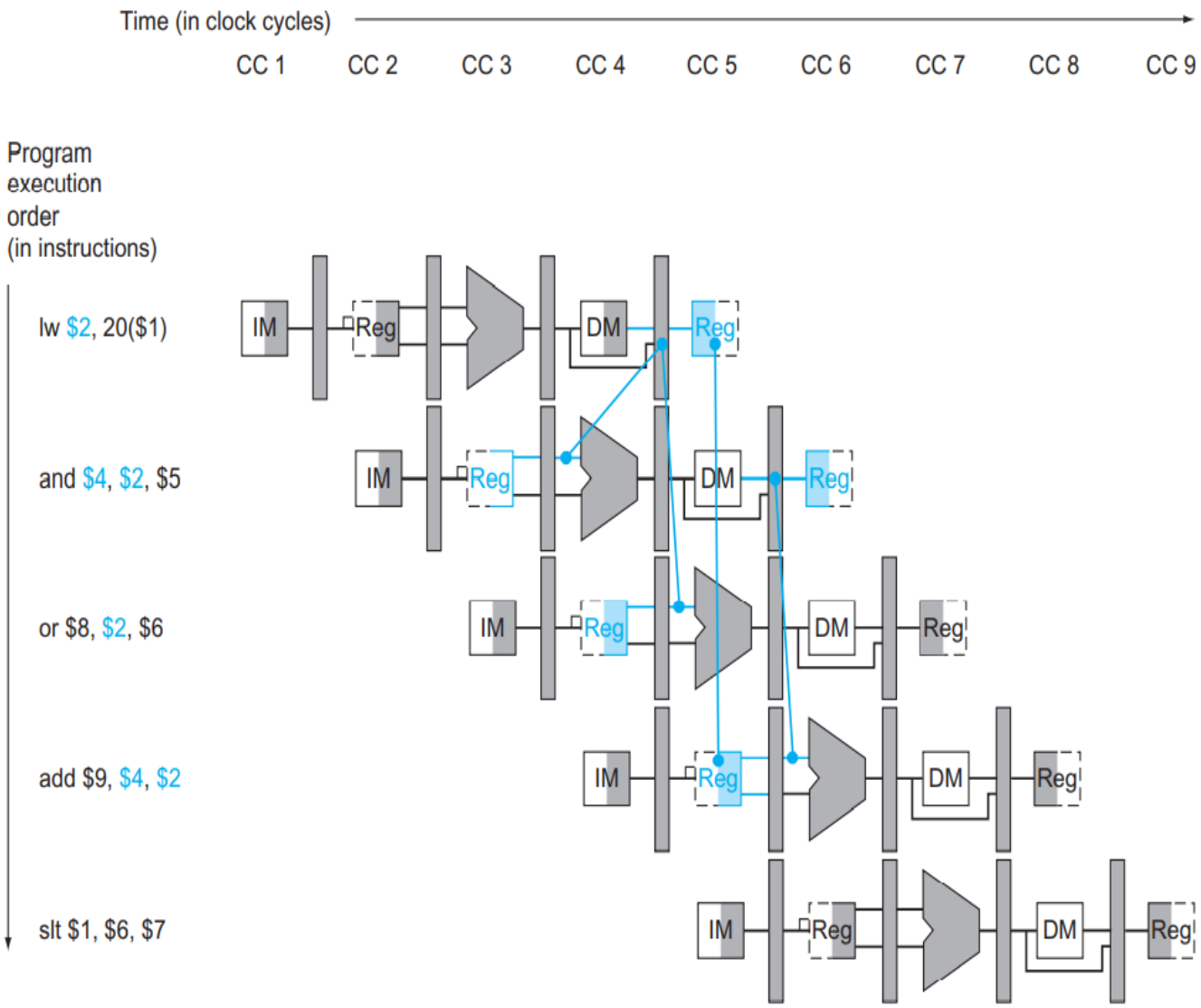


b. With forwarding

- We place multiplexers for two sources of ALU; the inputs of the multiplexers are
  - (1) from register file (ID/EX) - no data hazard
  - (2) from data memory or from an earlier ALU result (MEM/WB)
  - (3) from the prior ALU result (EX/MEM)
- The forwarding unit determines one of the three inputs using 2-bit signal
  - ForwardA for the first source of the ALU
  - ForwardB for the second source of the ALU

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.

Data Hazard - Another Example Scenario



- 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 is ready in MEM stage (MEM/WB register) at CC5
- **and $4, $2, $5**
  - The data is needed in EX stage at least by CC4
- Data forwarding forwards the data from *MEM/WB register* to *Rs port of the ALU*; but, it is too late
  - How can we handle this problem?