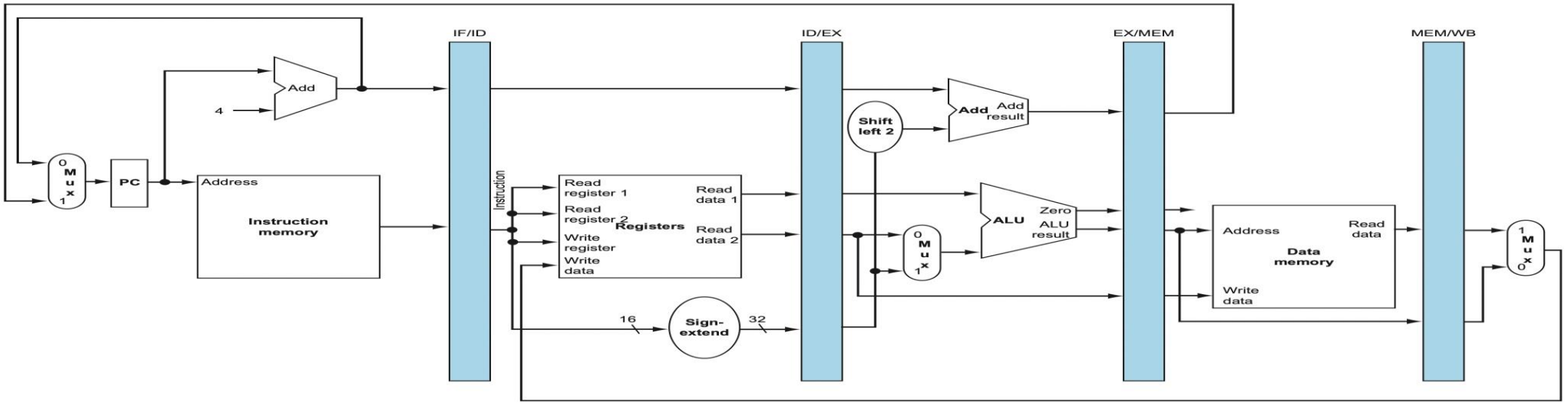
Computer Architecture
(ENE1004)

Lec – 16: The Processor (Chapter 4) - 7

Upcoming Schedule

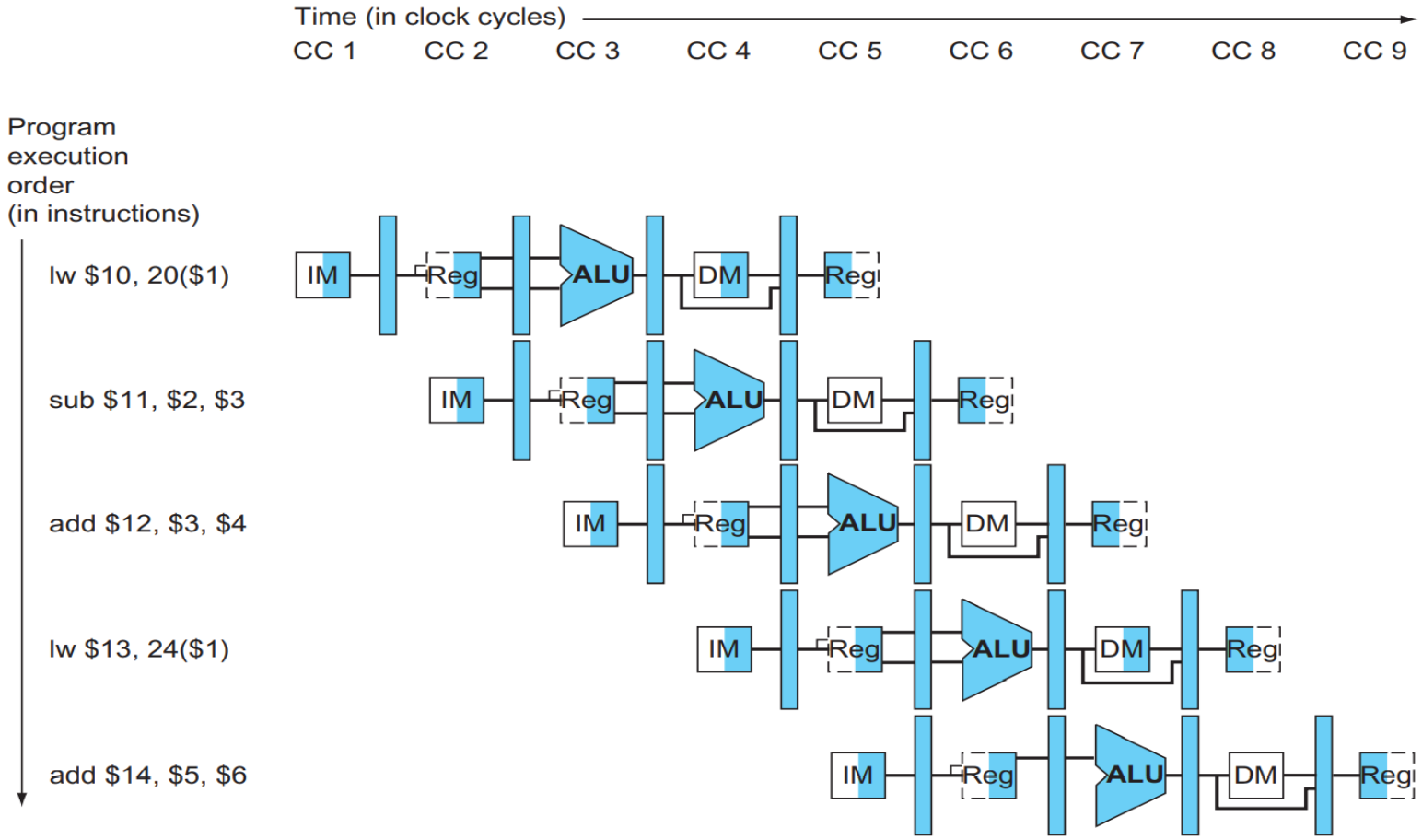
- May 15 (next Monday, university anniversary) – No class
  - A recorded video will be uploaded later for backup
- May 21 – Assignment #1 deadline
- May 29 – No class
  - A recorded video will be uploaded later for backup

Two Key Points in Pipelined Datapath



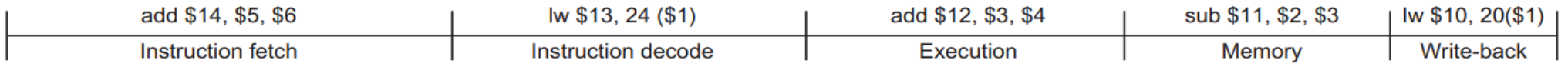
- (1) To pass something from a stage to another, it must be placed in a pipeline register
  - Otherwise, the data or the information is lost when the next instruction enters that stage
  - For **sw**, the register 2 value obtained from ID is passed to EX (through ID/EX register) and again is passed to MEM (through EX/MEM register)
- (2) Each hardware component - instruction memory, register read ports, ALU, data memory, register write port - can be used only within a single stage
  - Hardware cannot be shared by multiple instructions at a time
  - Register file has separate ports for read/write; so, it can be considered as different components

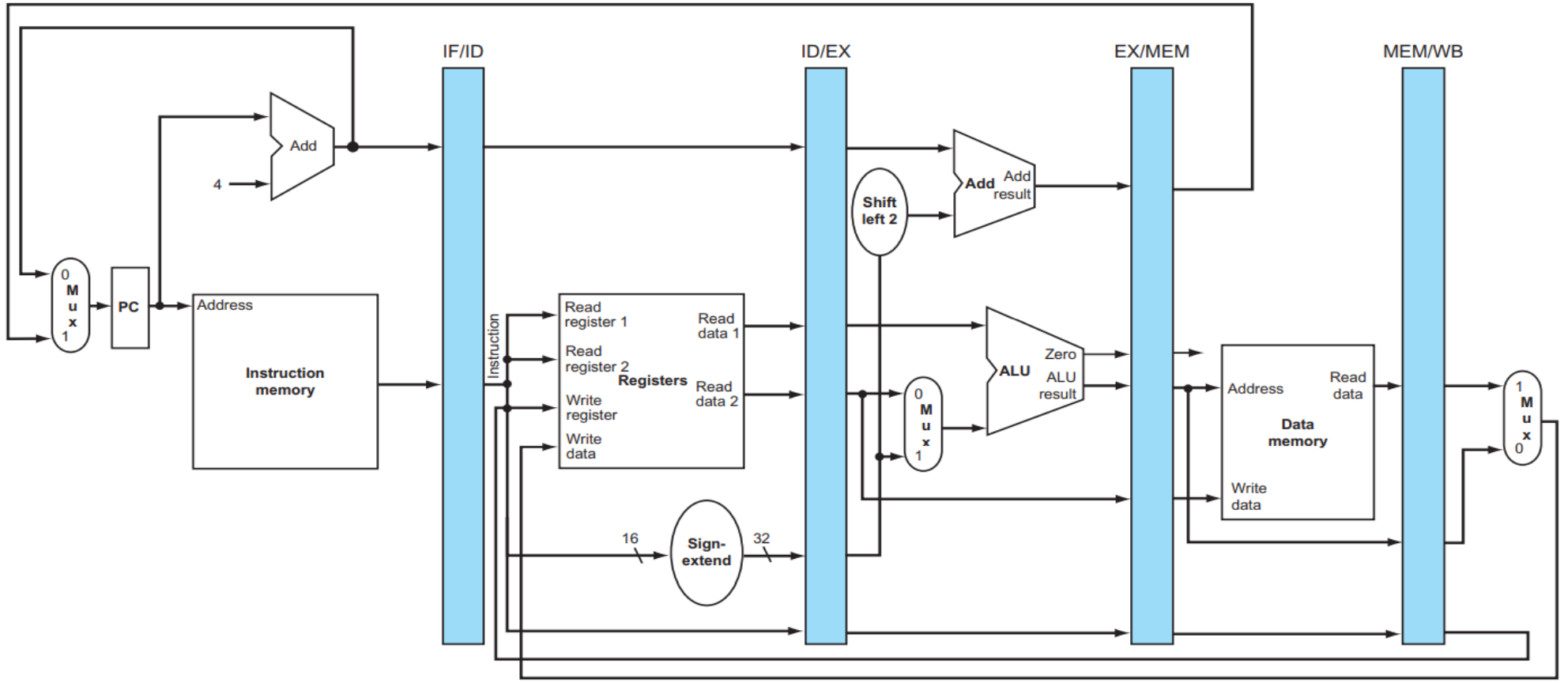
Example: Pipeline Diagram of Five Instructions



- Time advance from left to right; instructions advance in each clock cycle (CC)
- Five instructions enter to the pipeline one after another (from top to bottom)
- In each cycle, all instructions are in the different stages (use different components)
  - Register file can be read and written in the same cycle, thanks to its separate ports

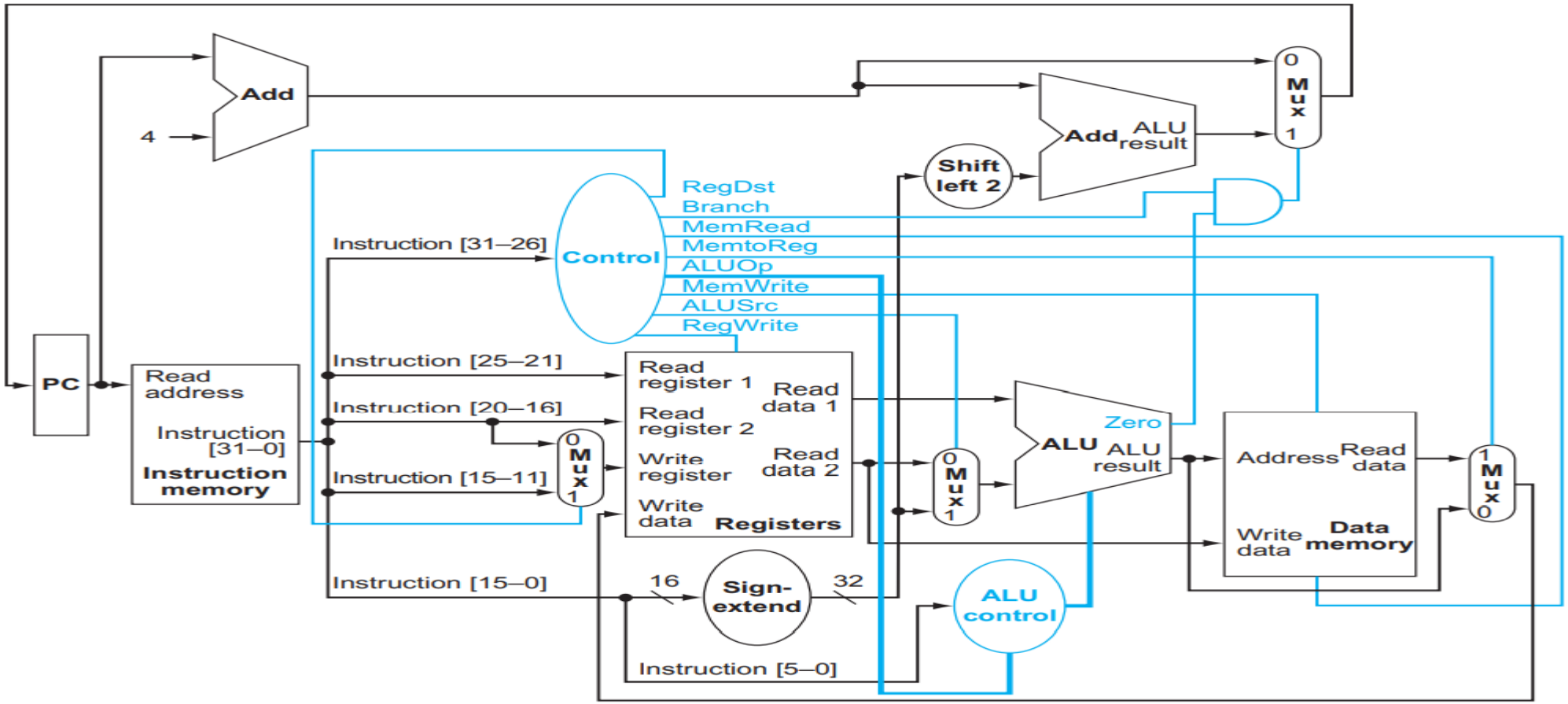
Example: Five Instructions in CC5





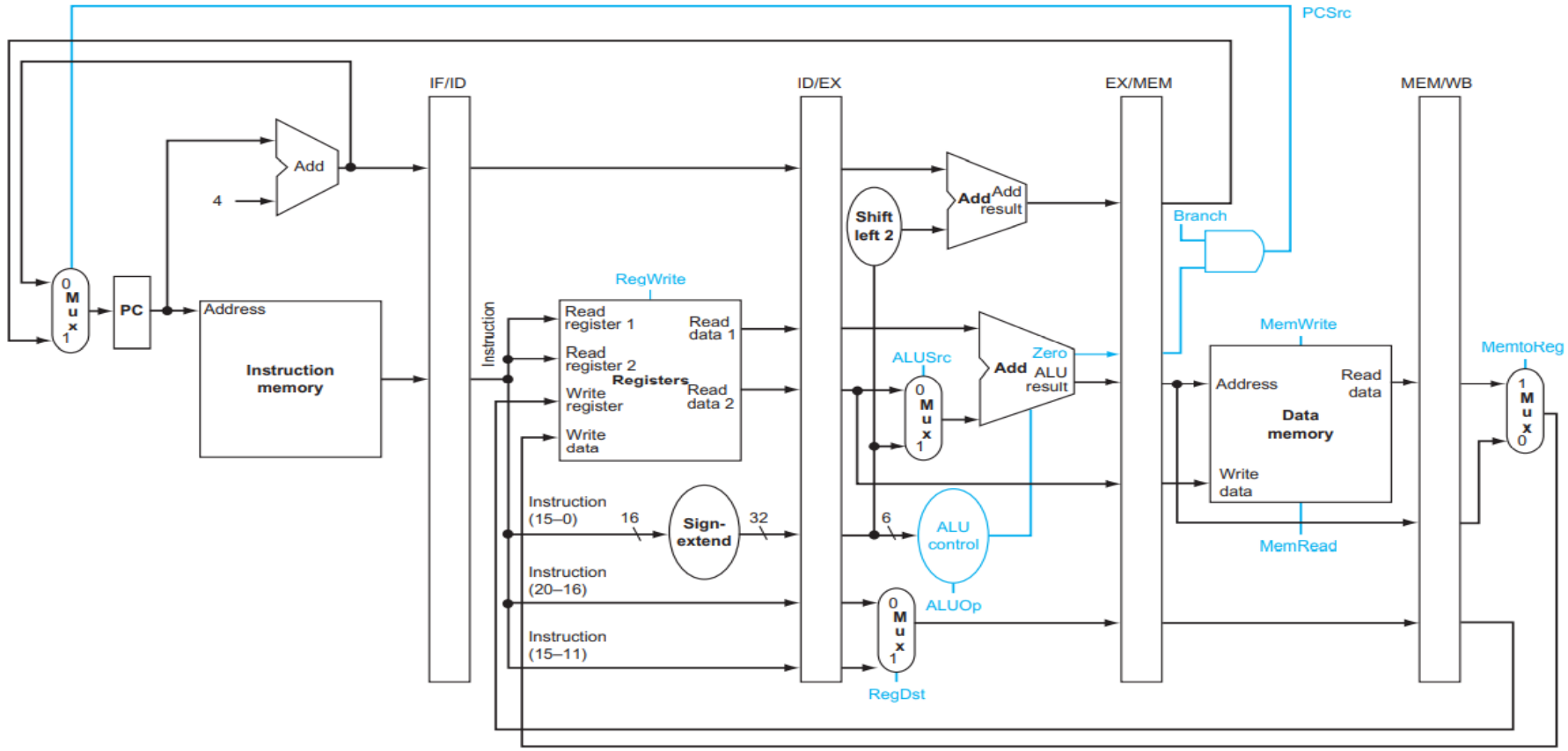
- There exists a (pipelined) datapath, which can process five instructions at a time
- In a given time (CC), all instructions are in the different stages (use different HWs)

Control Signals for Non-Pipelined Datapath (Review)



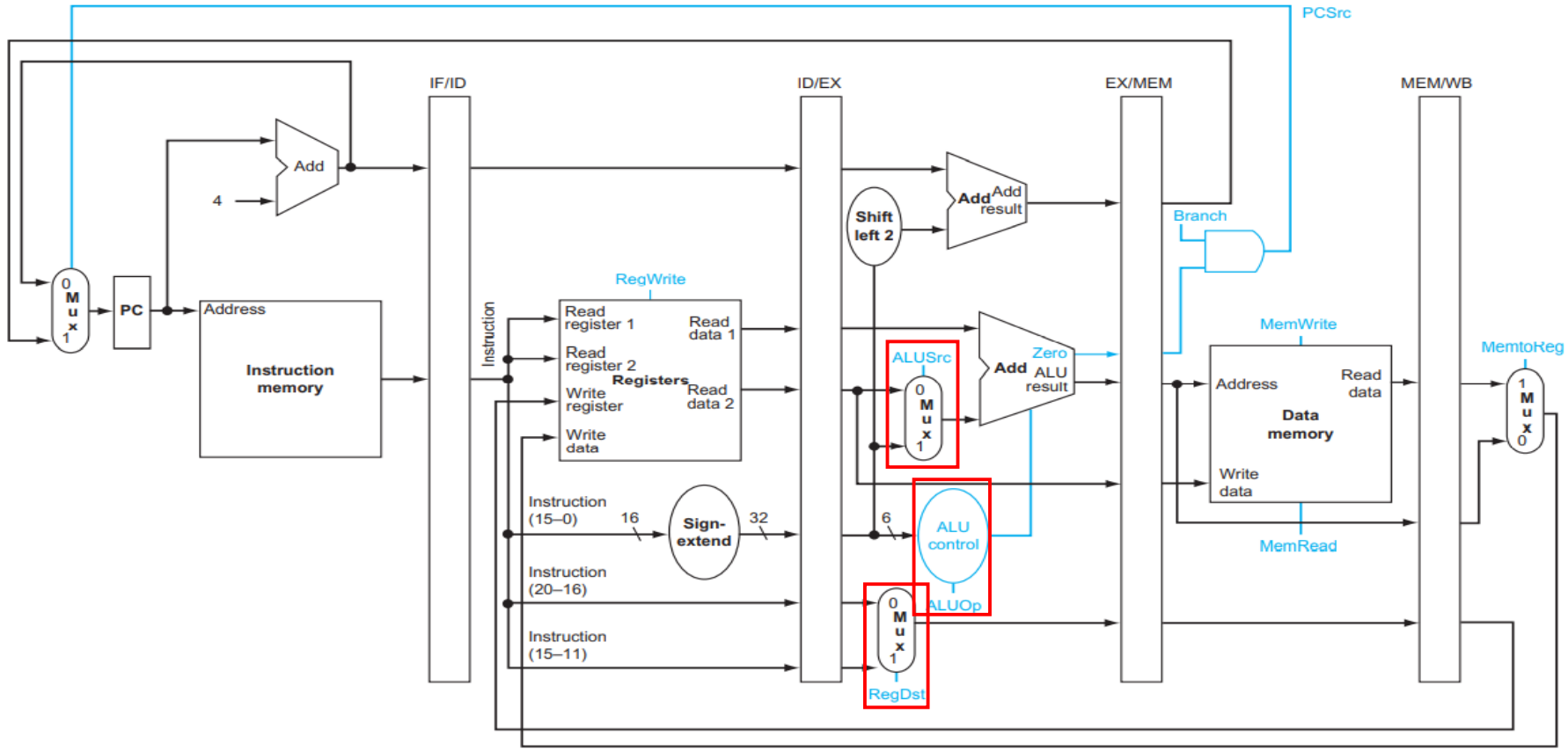
Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Control Signals for Pipelined Datapath (1)



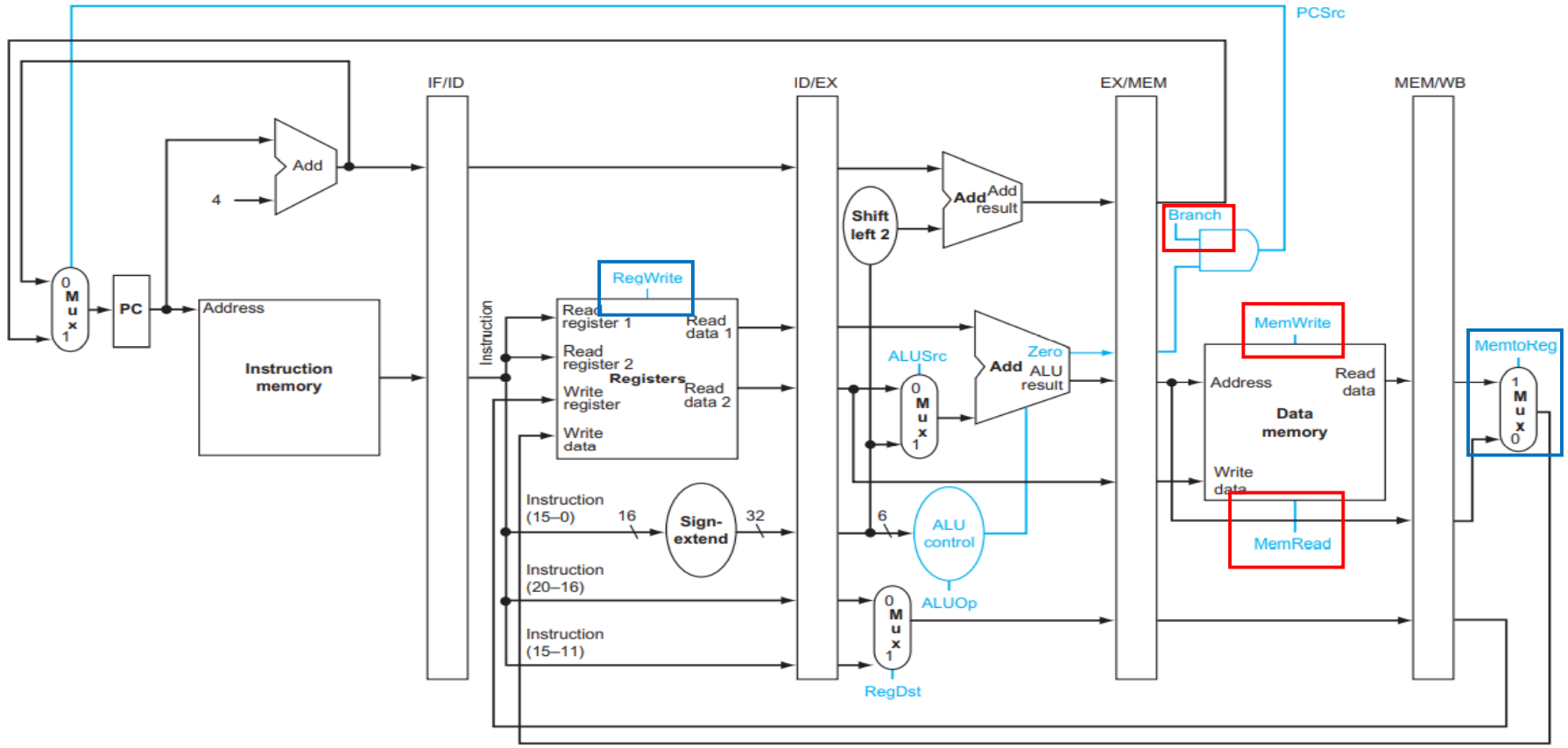
- We divide the control signals into five groups according to the stages
  - Each control line is associated with an active component in a single stage
- IF stage: There is nothing special to control (as is in non-pipelined datapath)
- ID stage: There is nothing special to control (as is in non-pipelined datapath)
  - Note that we do not need a control signal for reading registers; **RegWrite** is used in [WB] stage

Control Signals for Pipelined Datapath (2)



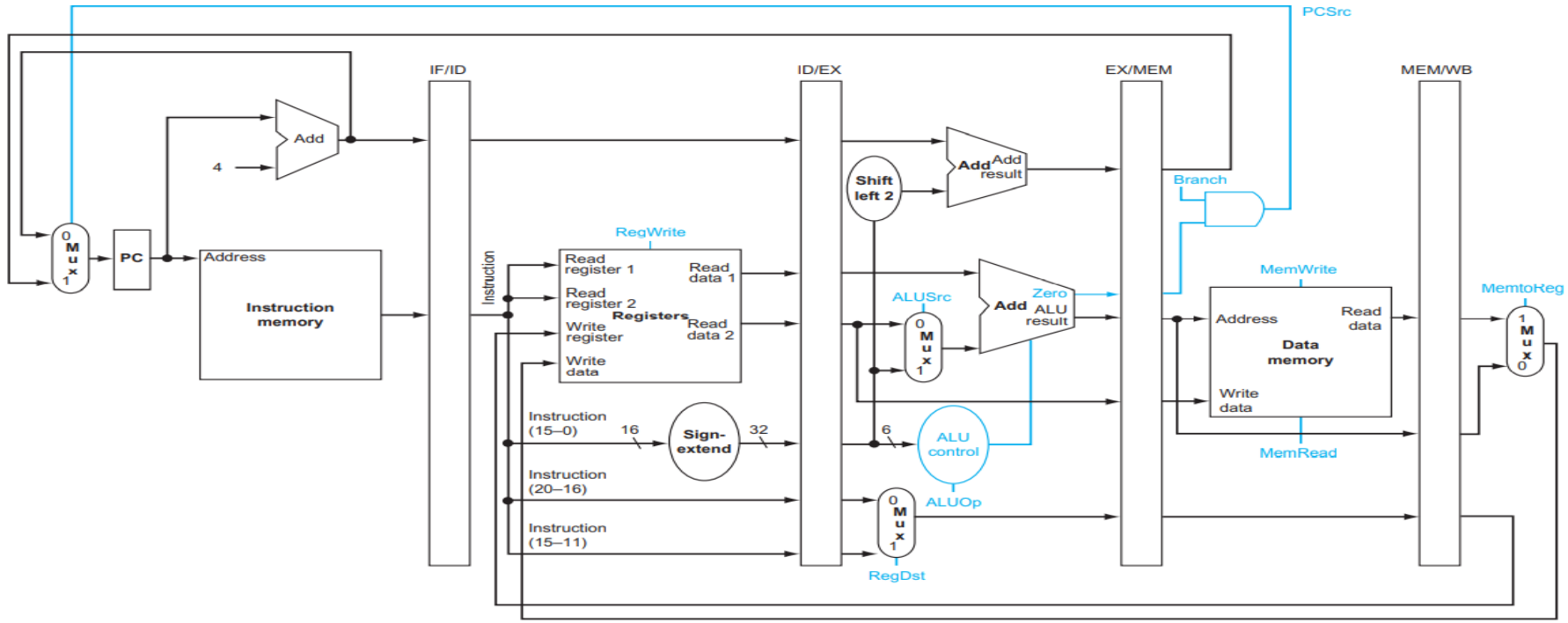
- EX stage: **ALUSrc**, **ALUOp**, and **RegDst** should be set
  - **ALUSrc** selects either Read data 2 or a sign-extended immediate for the 2nd input of the ALU
  - **ALUOp** selects the ALU operation in collaboration with 6-bit func field (the rightmost 6 bits)
  - **RegDst** selects the Write register (1 for R-type instructions; 0 for load instructions)

Control Signals for Pipelined Datapath (3)



- MEM stage: **Branch**, **MemRead**, and **MemWrite** should be set
  - **Branch, MemRead,** and **MemWrite** are set for branch equal, load, and store instructions
- WB stage: **MemtoReg** and **RegWrite** should be set
  - **MemtoReg** decides between sending the ALU result or the memory value to the register file
  - **RegWrite** writes the chosen value into the register

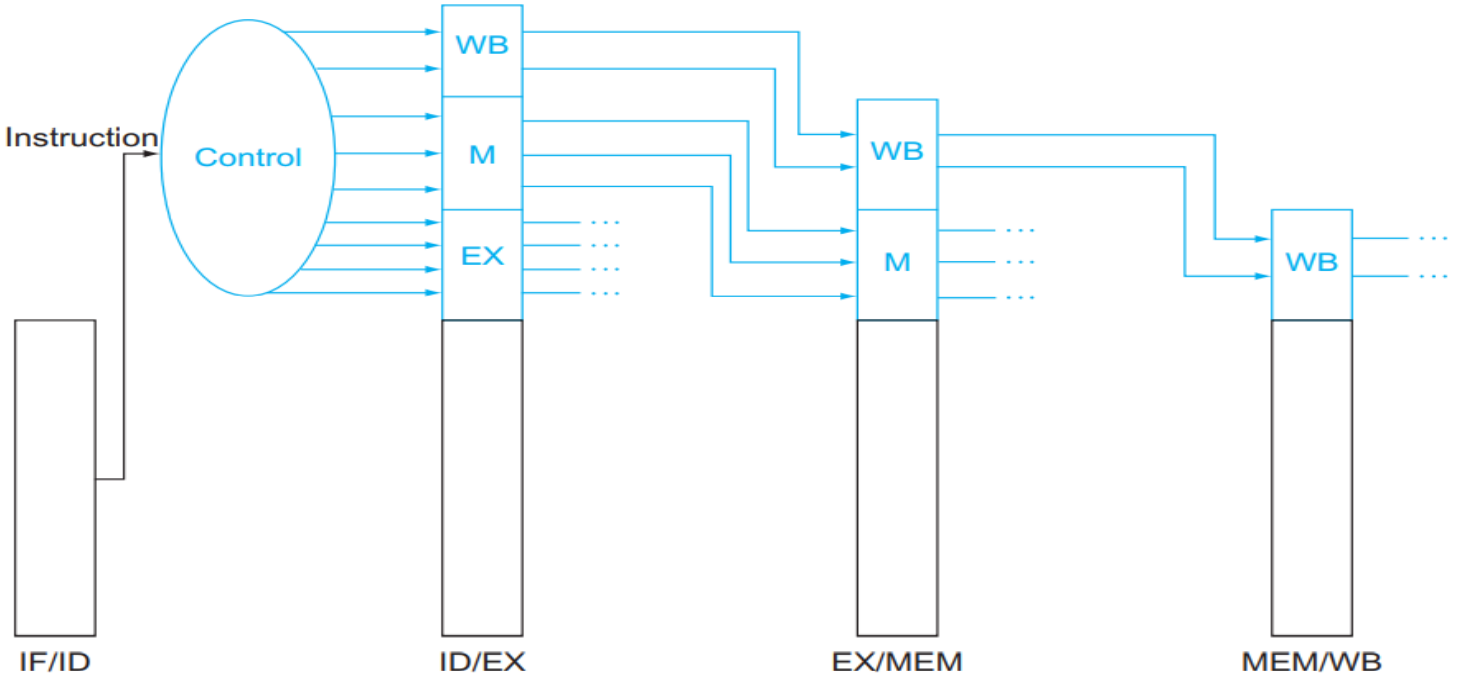
Control Signals for Pipelined Datapath (4)



- The meaning of the control signals remains unchanged; we can use the same values in the table
- But, the table is rearranged by grouping the signals by the pipeline stages

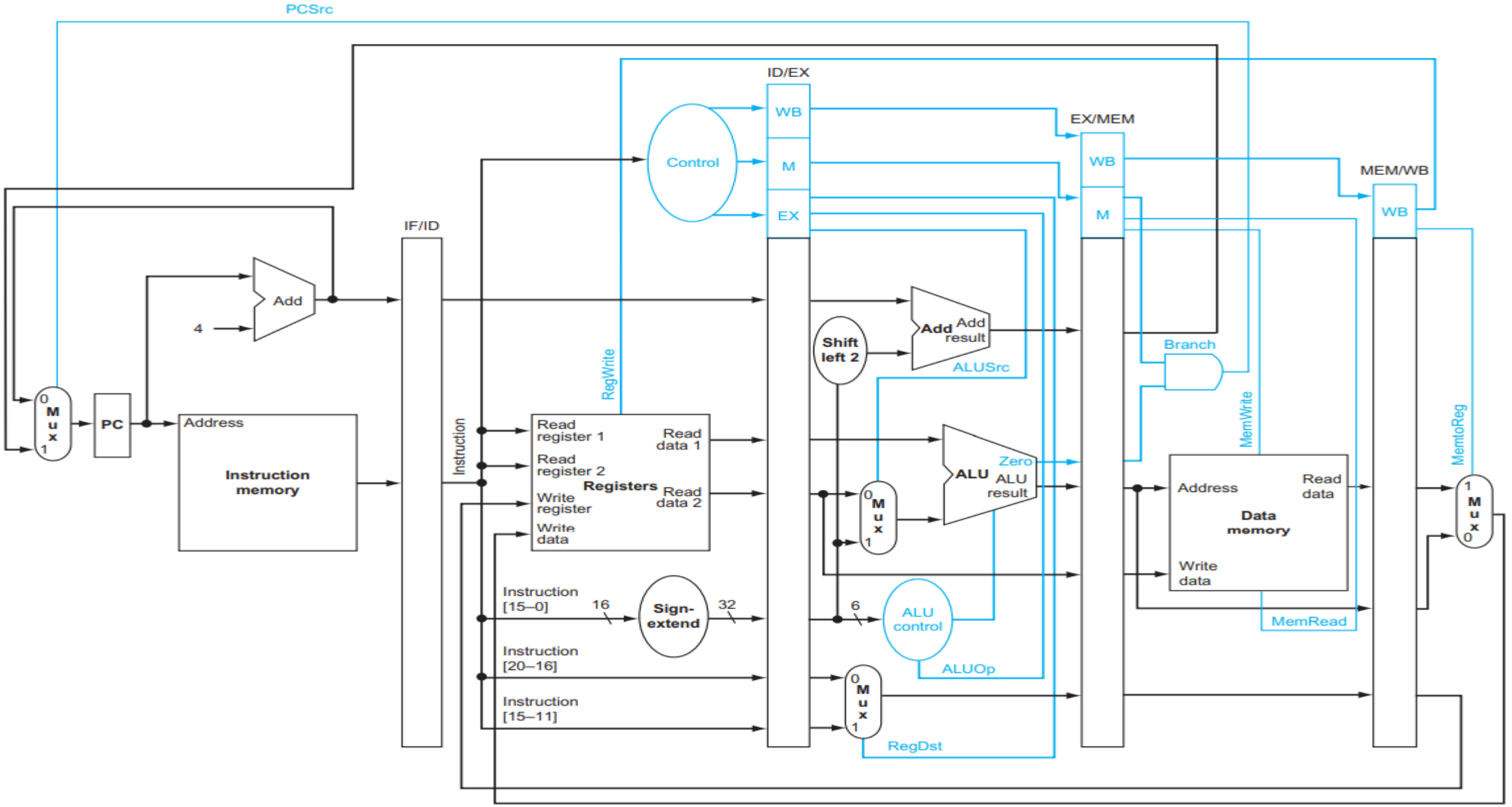
Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Implementing Control Lines for Pipelined Datapath (1)

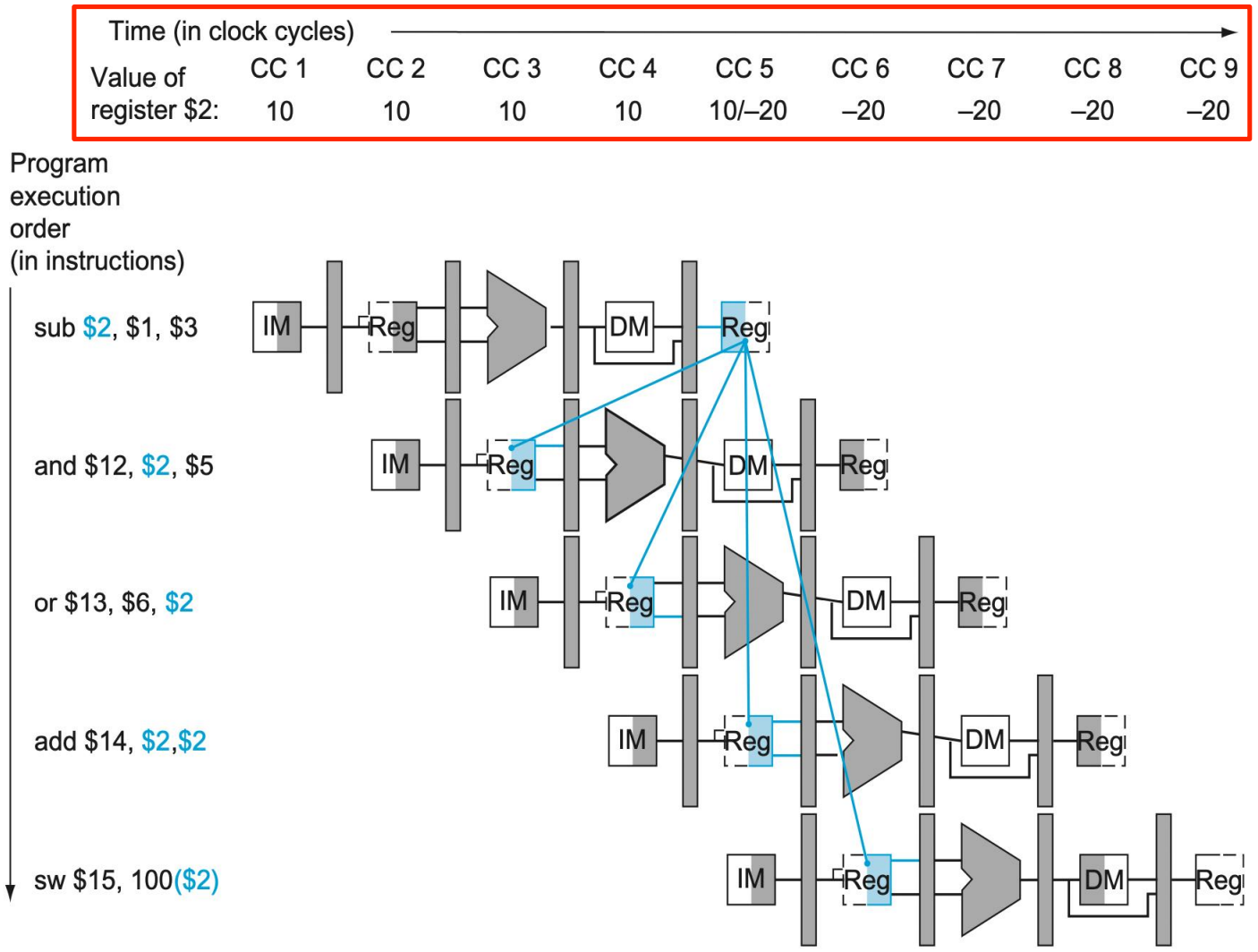


- For each instruction, the nine control signals should be set in each stage
  - (i) We can extend the pipeline registers to include control information
  - (ii) As an instruction advances, its control values can move from one pipeline register to the next
- Values for an instruction can be obtained in ID stage and advance through the stages
  - **RegDst**, **ALUOp1/0**, **ALUSrc** are used in EX stage; they are not copied to EX/MEM register
  - **Branch**, **MemRead**, **MemWrite** are used in MEM stage; they are not copied to MEM/WB register
  - **MemtoReg** and **RegWrite** are used in WB stage

Implementing Control Lines for Pipelined Datapath (2)

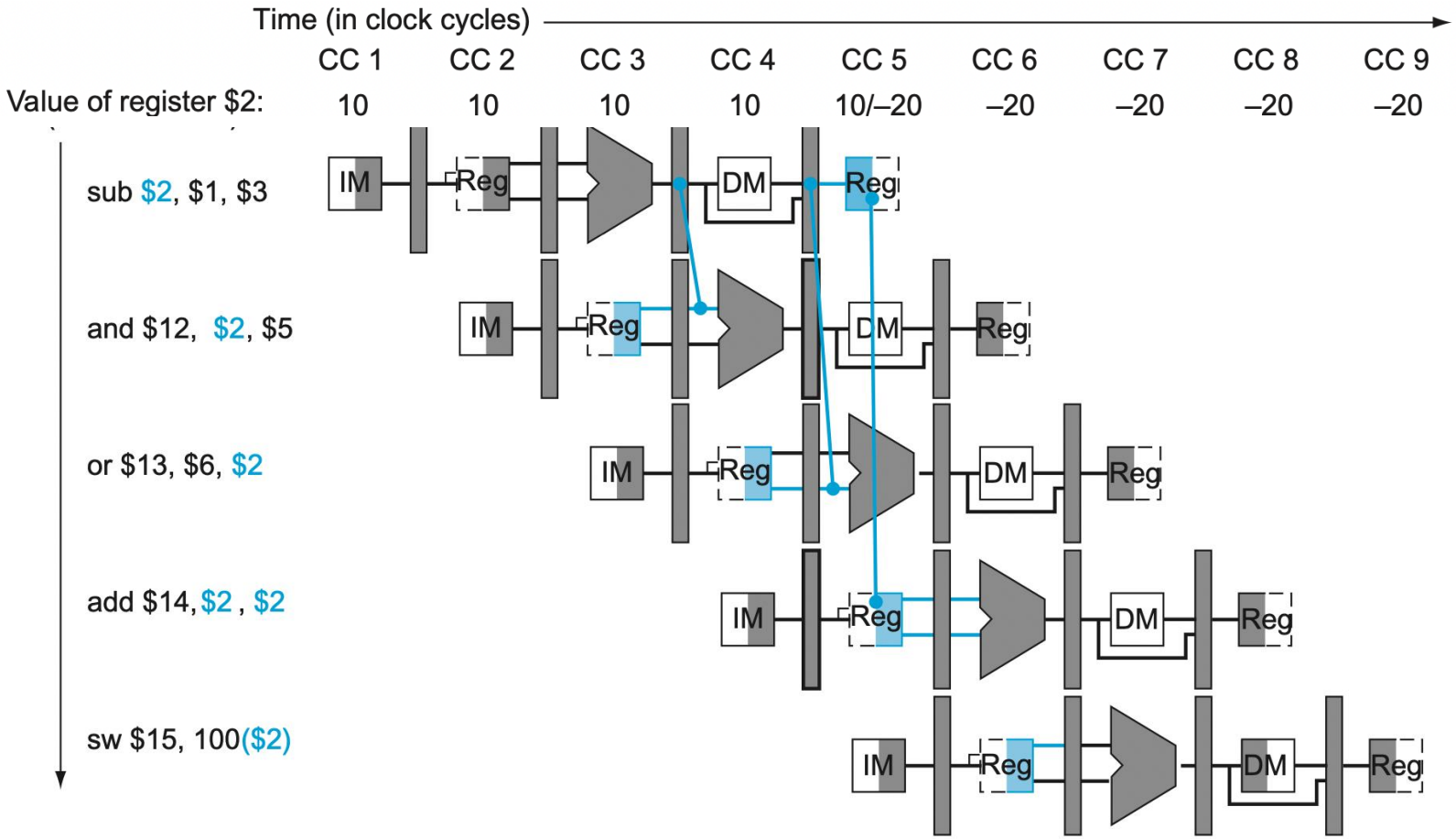


Data Hazard in RegWrite Scenarios



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

Data Forwarding (Solution) in RegWrite Scenarios



- The new value "-20" is generated by **sub** 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*** *port of the ALU* at CC4
  - For **or**, we can send the value from *MEM/WB register* to ***Rt*** *port of the ALU* at CC5