CMSC 411
Computer Architecture

Lecture 14

Pipelined Datapath and Control

Lecture's Overview

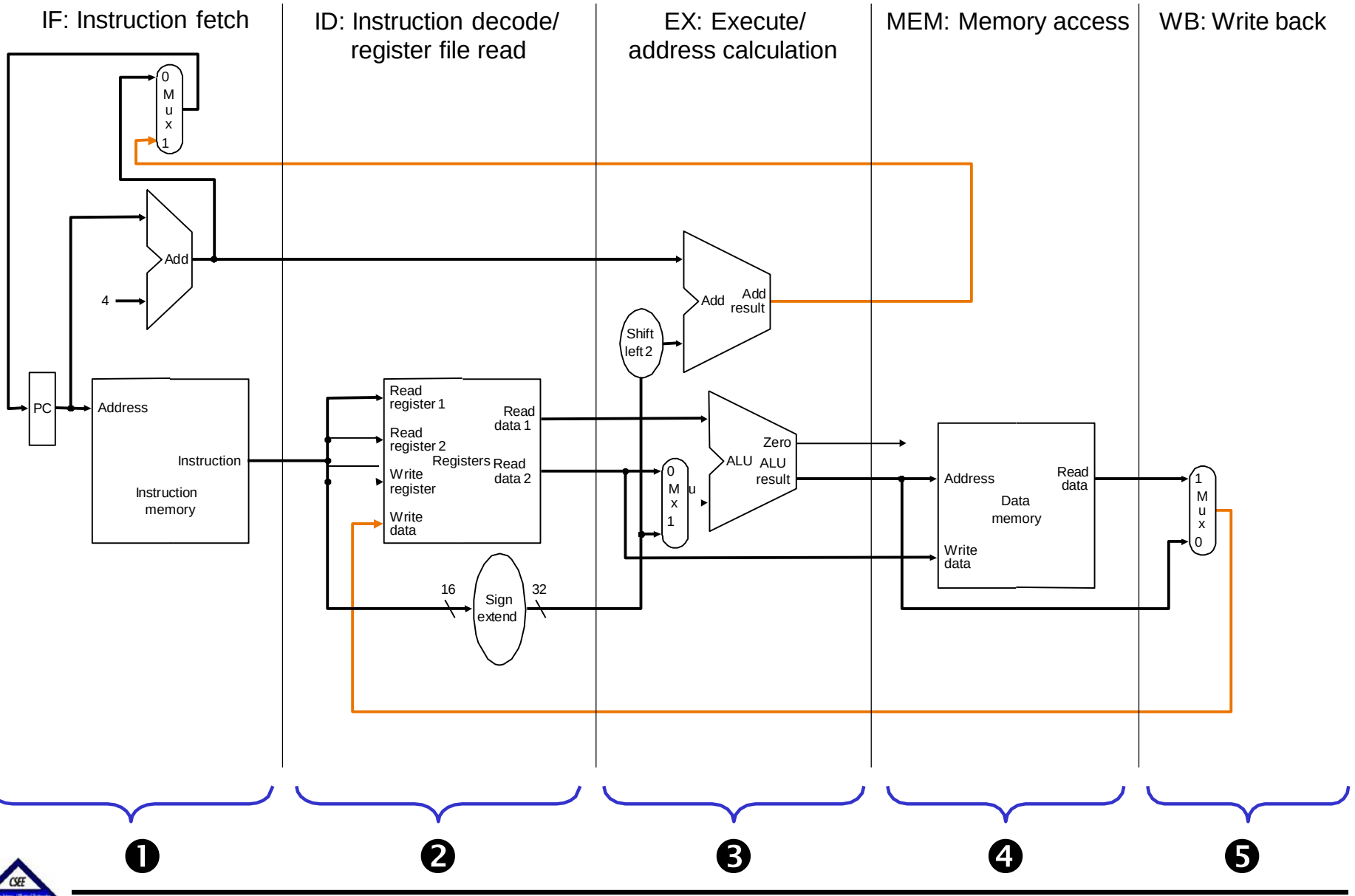
❑ *Previous Lecture:*

- An overview of pipelining
  - ➔ Pipelining concept is natural
  - ➔ Start handling of next instruction while current one is in progress
- Pipeline performance
  - ➔ Performance improvement by increasing instruction throughput
  - ➔ Ideal and upper bound for speedup is number of stages in pipeline
- Pipelined hazards
  - ➔ Structural, data and control hazards
  - ➔ Hazard resolution techniques

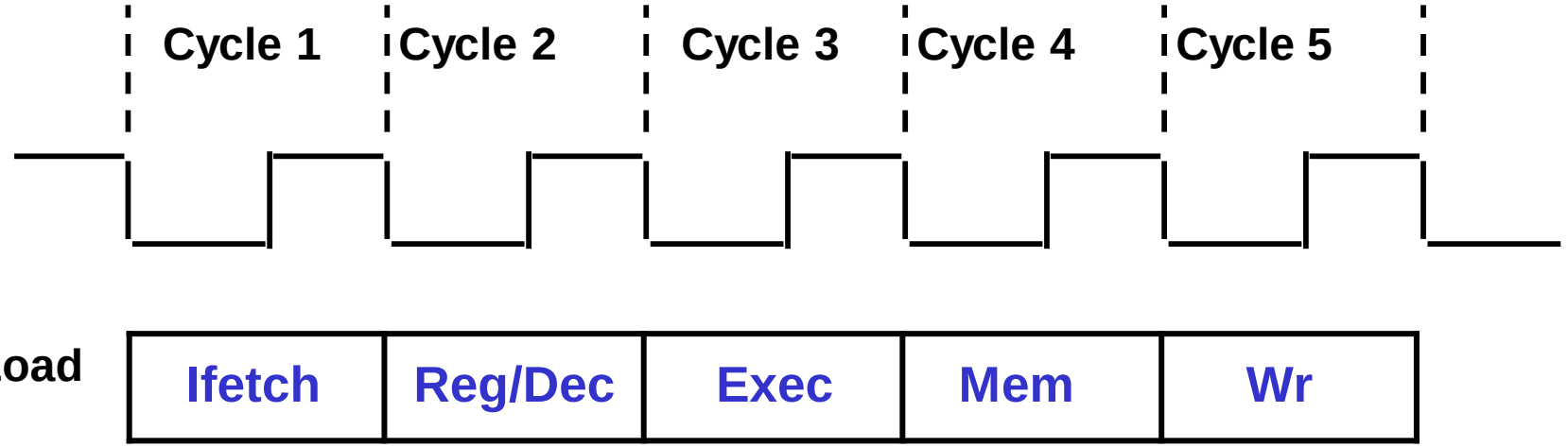
❑ *This Lecture:*

- Designing a pipelined datapath
- Controlling pipeline operations

Multi-cycle Instruction Execution



Stages of Instruction Execution

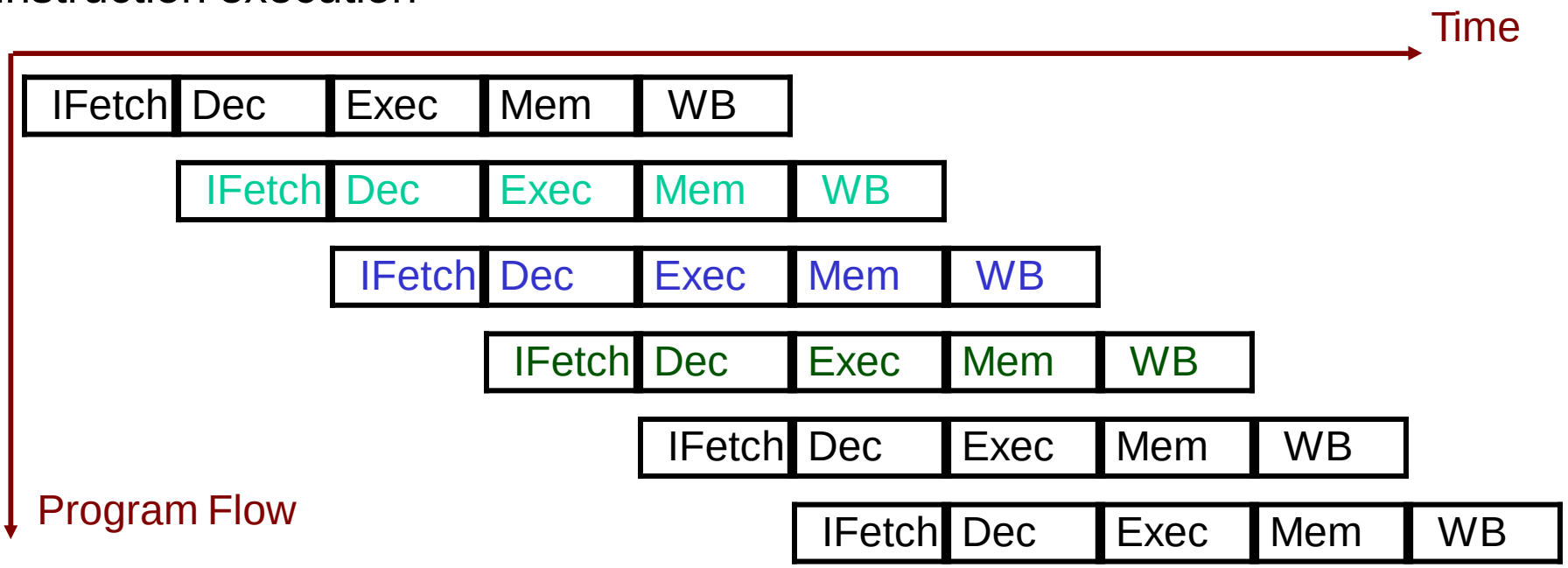


- The load instruction is the longest
- All instructions follows at most the following five steps:
  - ➔ Ifetch: Instruction Fetch
    - Fetch the instruction from the Instruction Memory
  - ➔ Reg/Dec: Registers Fetch and Instruction Decode
  - ➔ Exec: Calculate the memory address
  - ➔ Mem: Read the data from the Data Memory
  - ➔ Wr: Write the data back to the register file

* Slide is courtesy of Dave Patterson

Instruction Pipelining

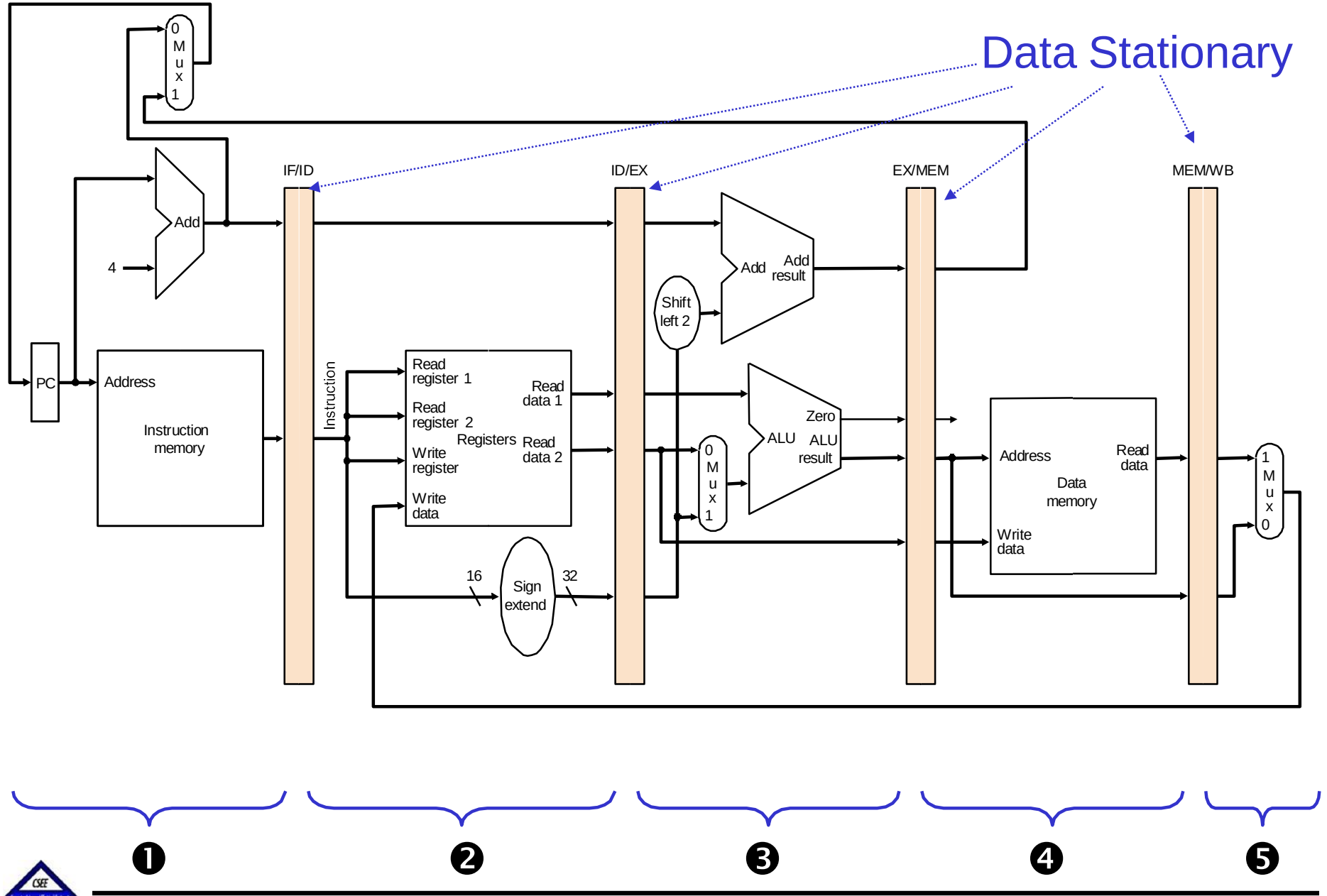
- Start handling of next instruction while the current instruction is in progress
- Pipelining is feasible when different devices are used at different stages of instruction execution



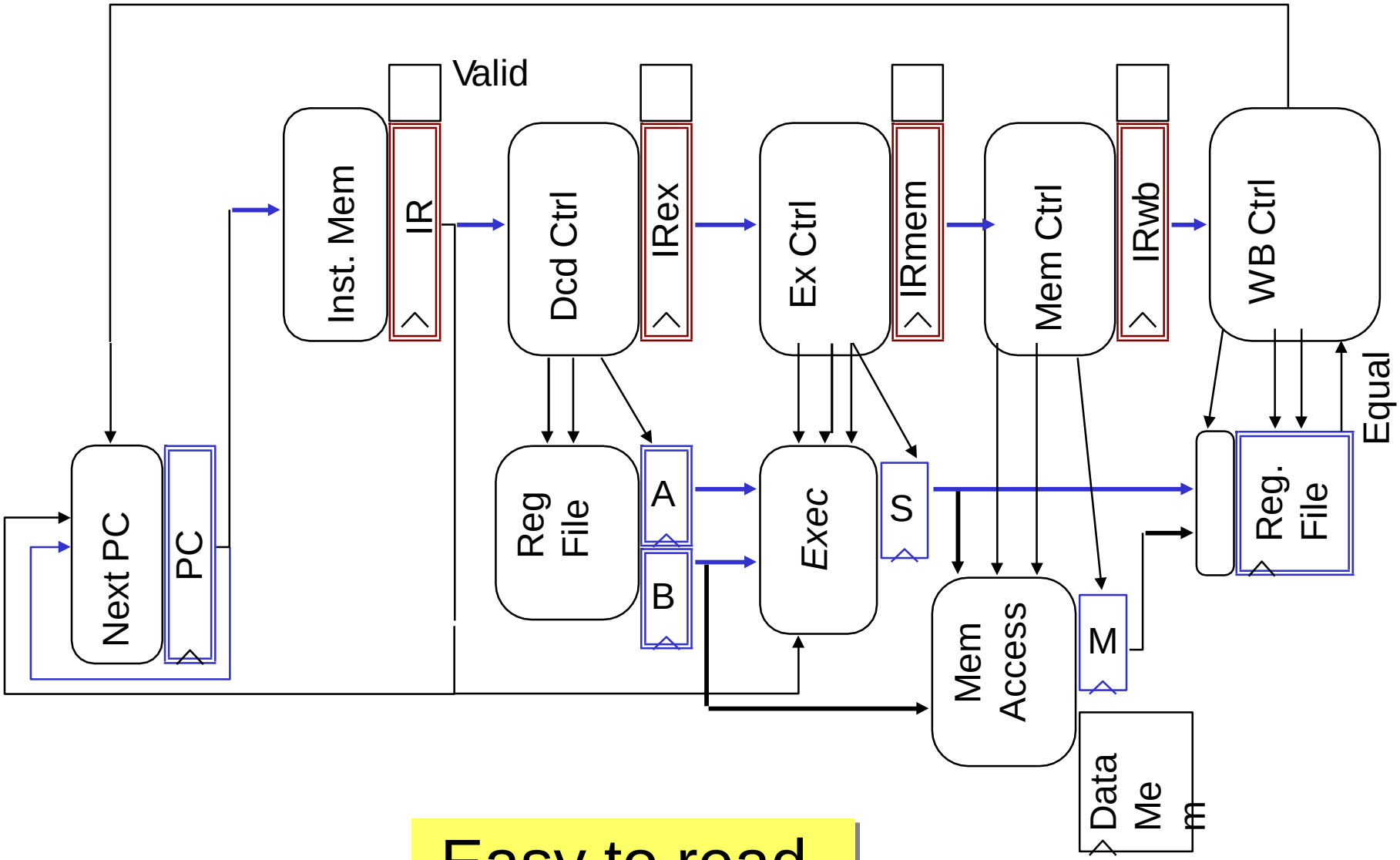
$$\text{Time between instructions}_{pipelined} = \frac{\text{Time between instructions}_{nonpipelined}}{\text{Number of pipe stages}}$$

Pipelining improves performance by increasing instruction throughput

Pipelined Datapath



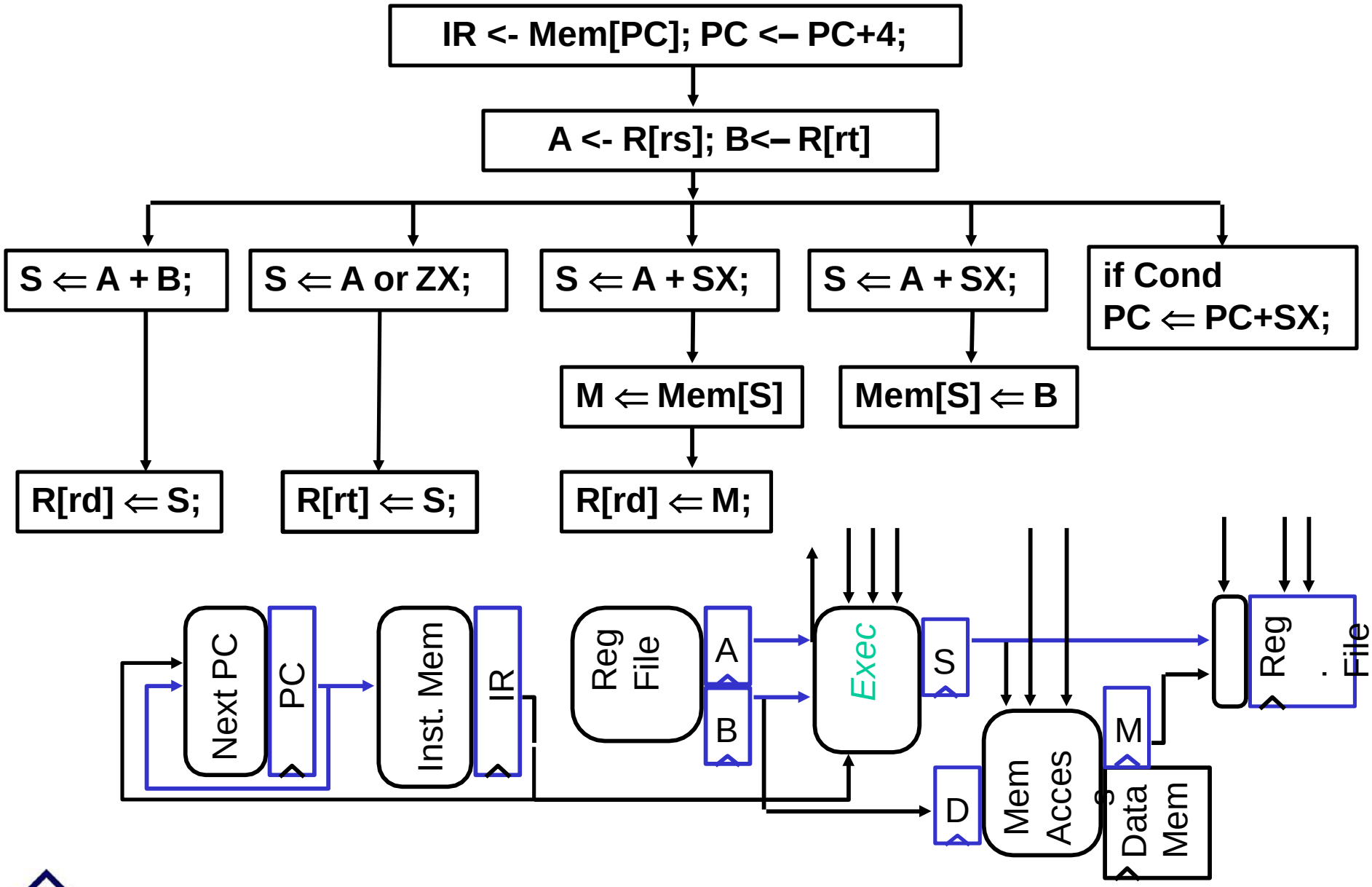
Datapath for Pipelined Processor



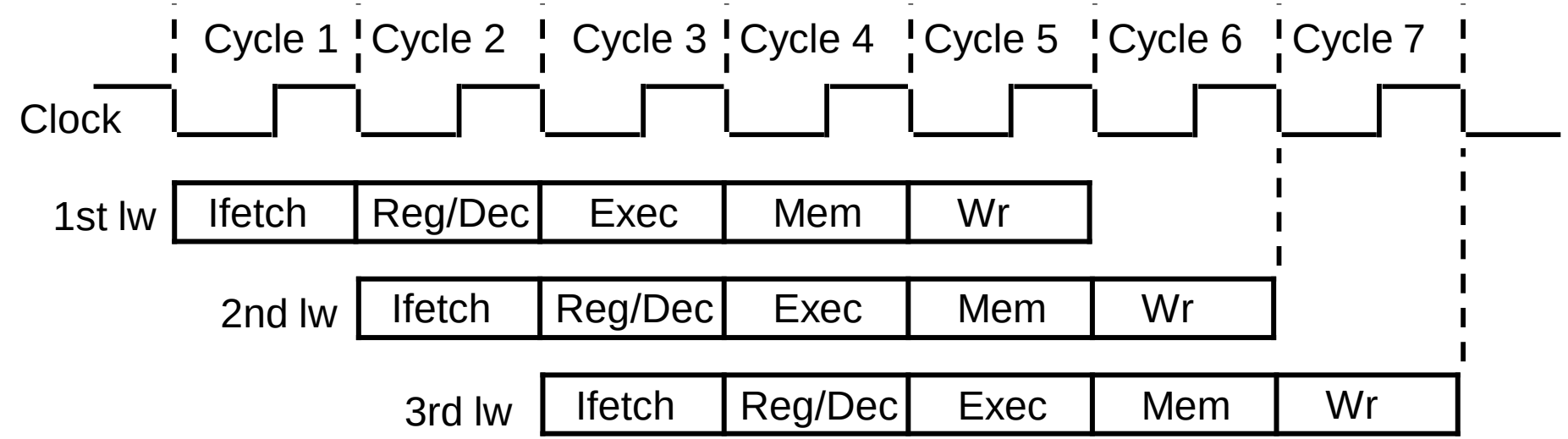
Easy to read

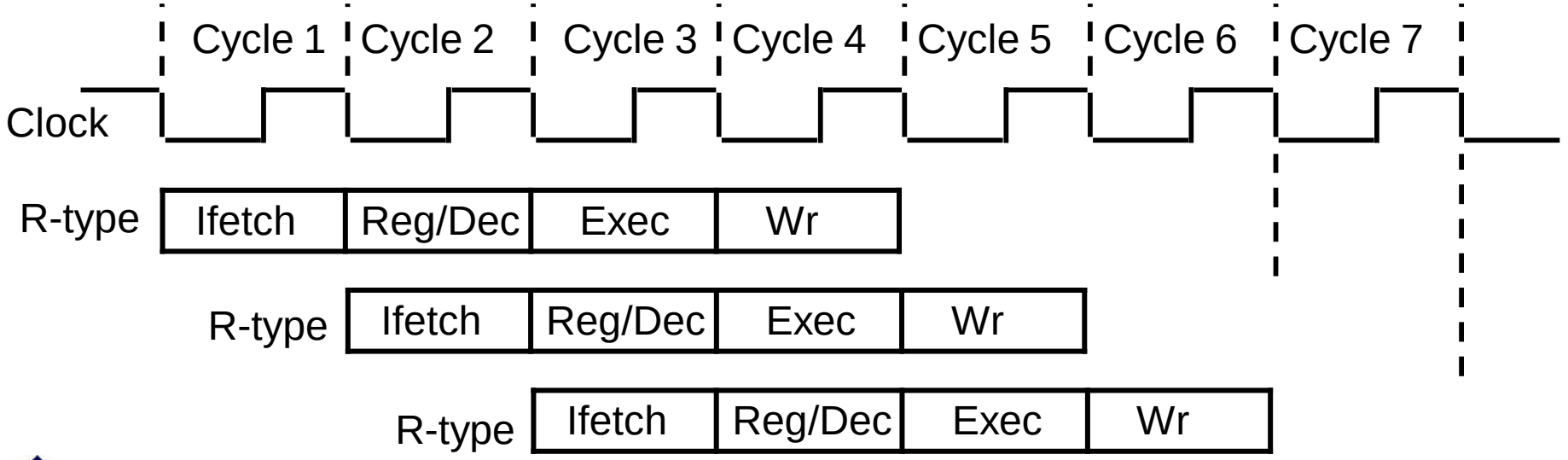
* Slide is courtesy of Dave Patterson

Control and Datapath

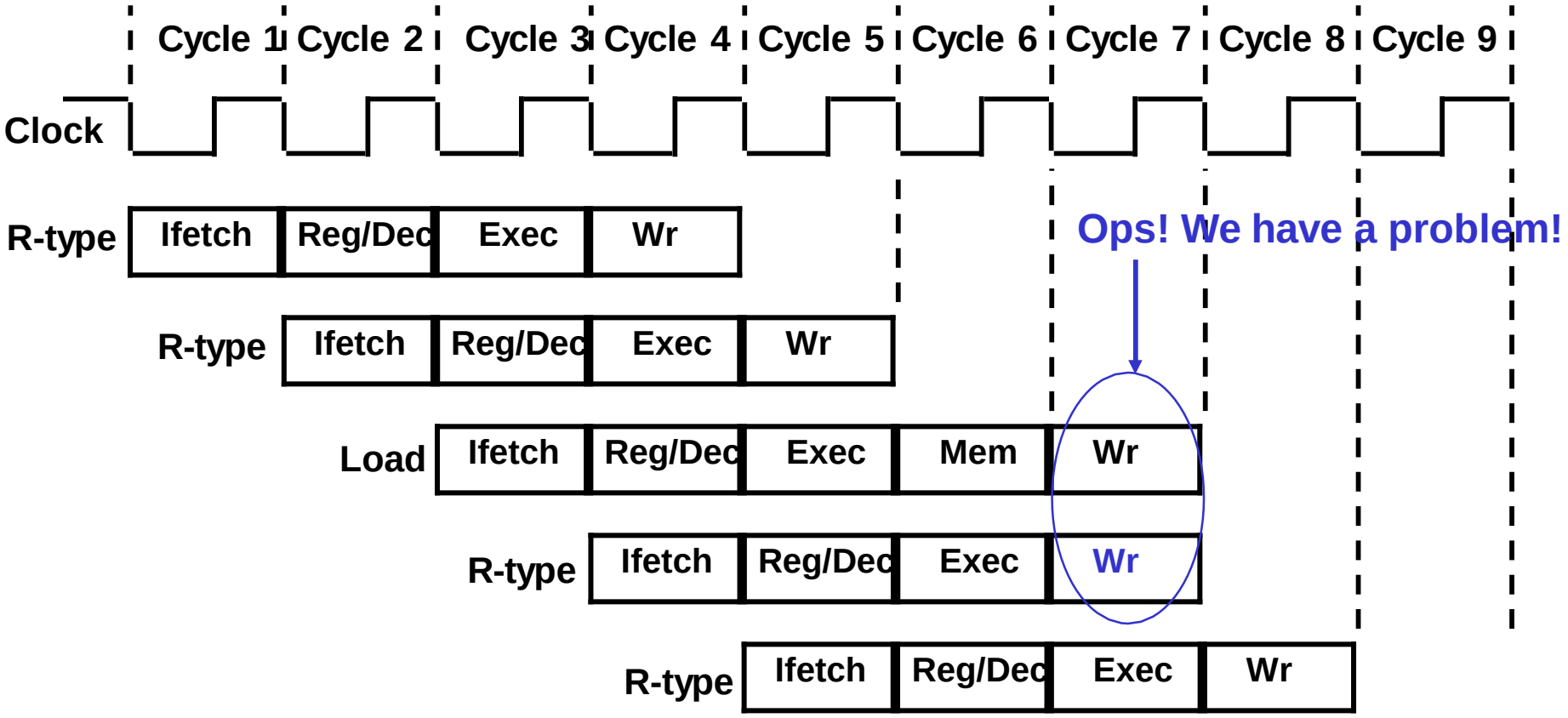


Pipelining the same Instruction





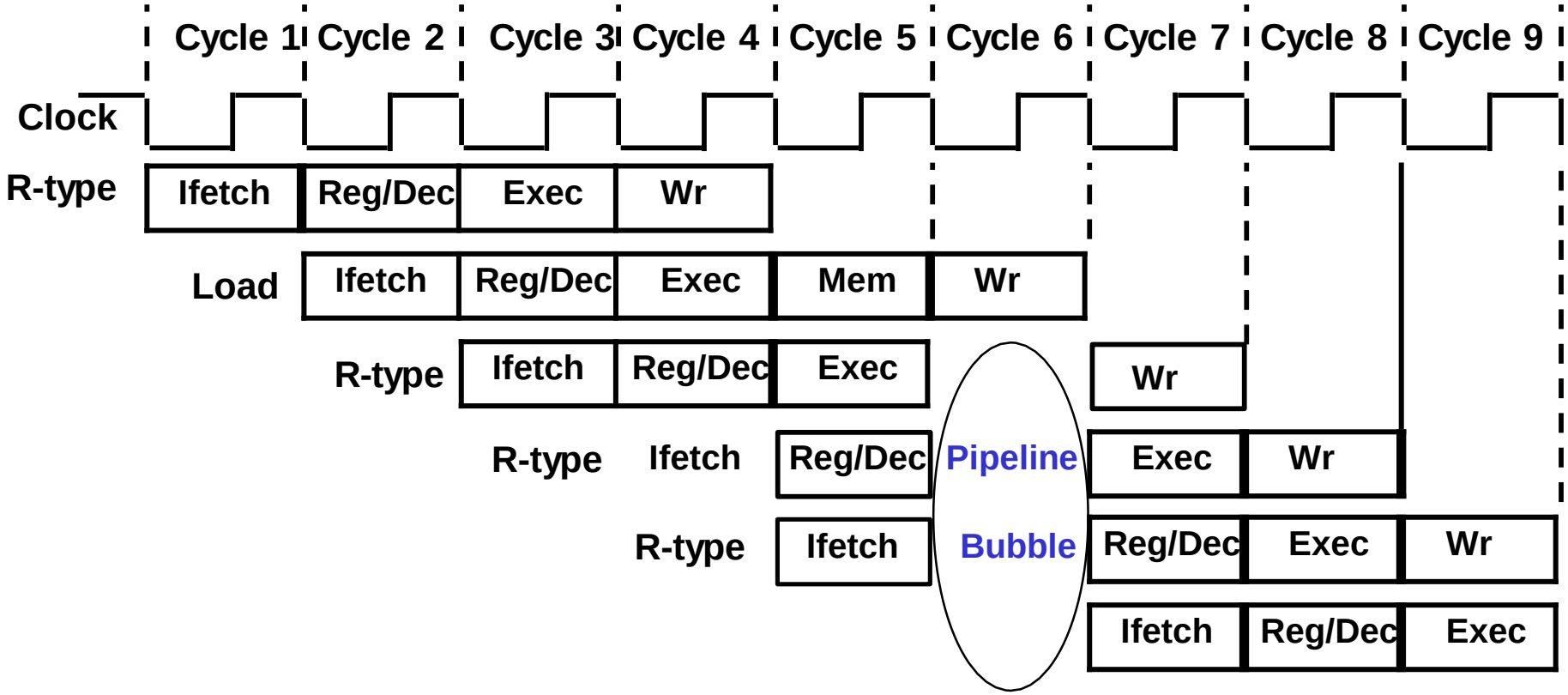
Pipelining R-type & Load Instructions



We have pipeline conflict or structural hazard:

- ➔ Two instructions try to write to register file at the same time!
- ➔ Only one write port

Solution 1: Insert "Bubble" into Pipeline



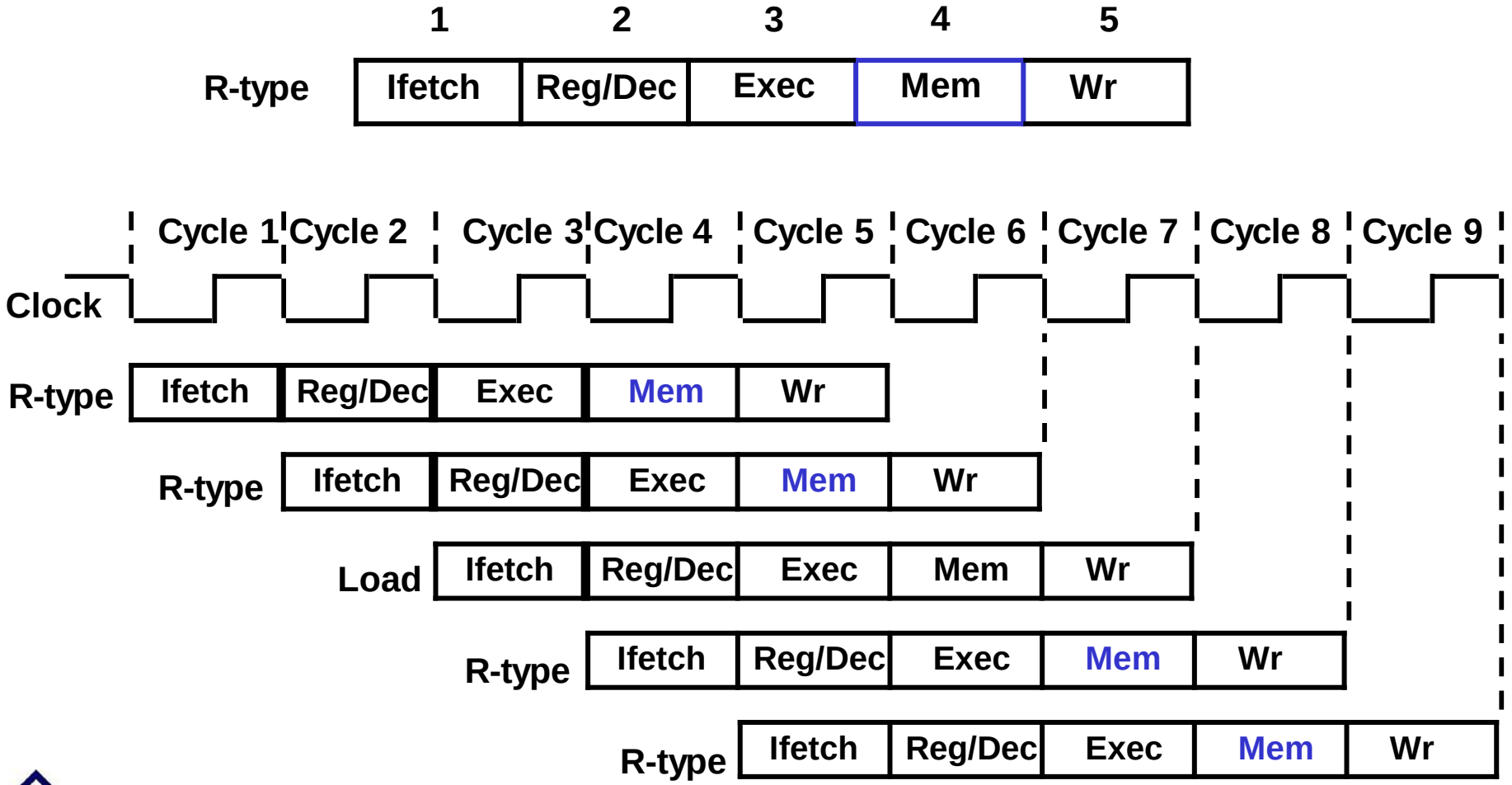
- Insert a "bubble" into pipeline to prevent 2 writes at same cycle
  - ➔The control logic can be complex.
  - ➔ Lost instruction fetch and speedup opportunity
- No instruction is started in Cycle 6!

Solution 2: Delay Write by One Cycle

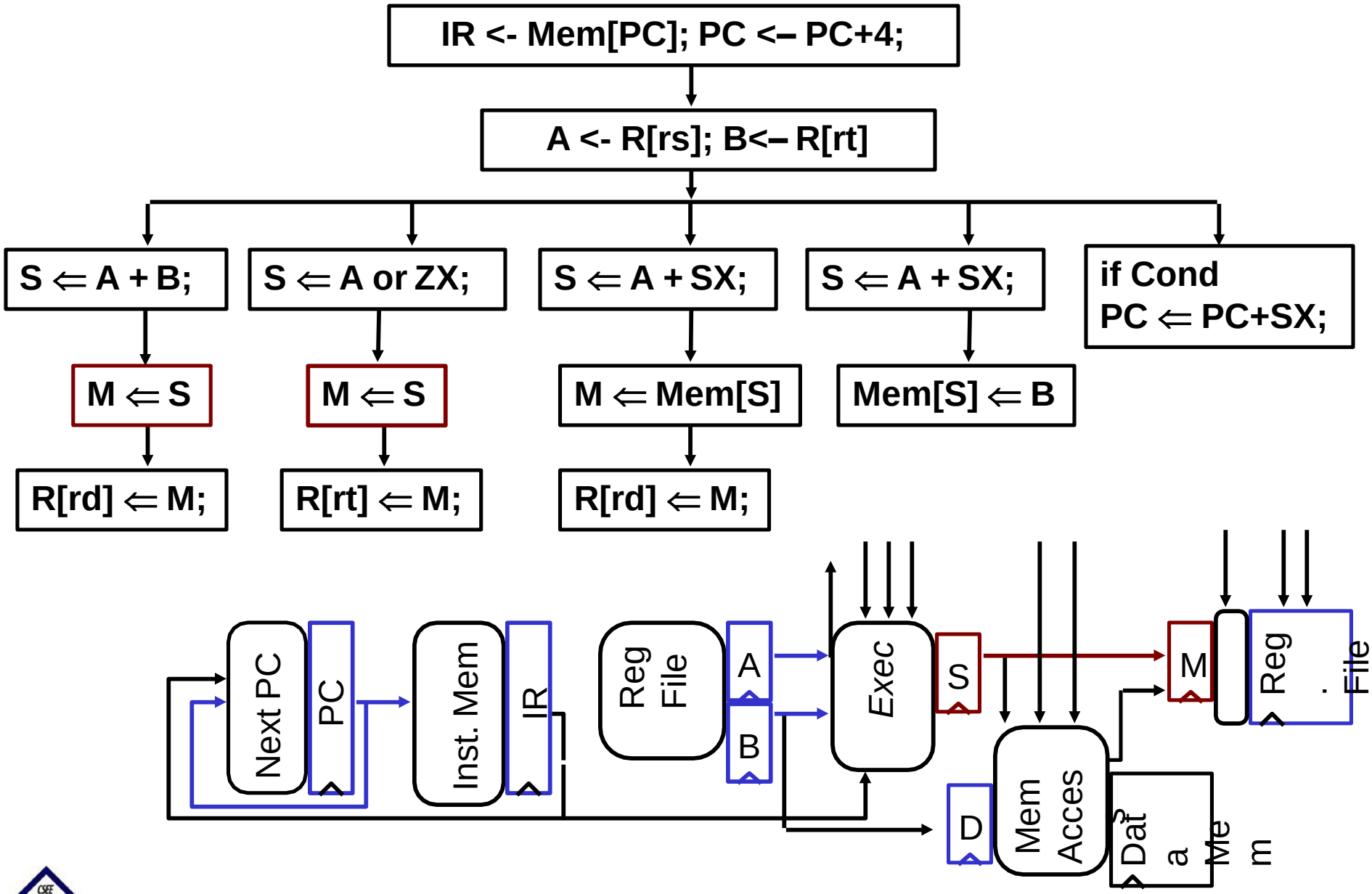
Delay R-type's register write by one cycle:

➔ Now R-type instructions also use Reg File's write port at Stage 5

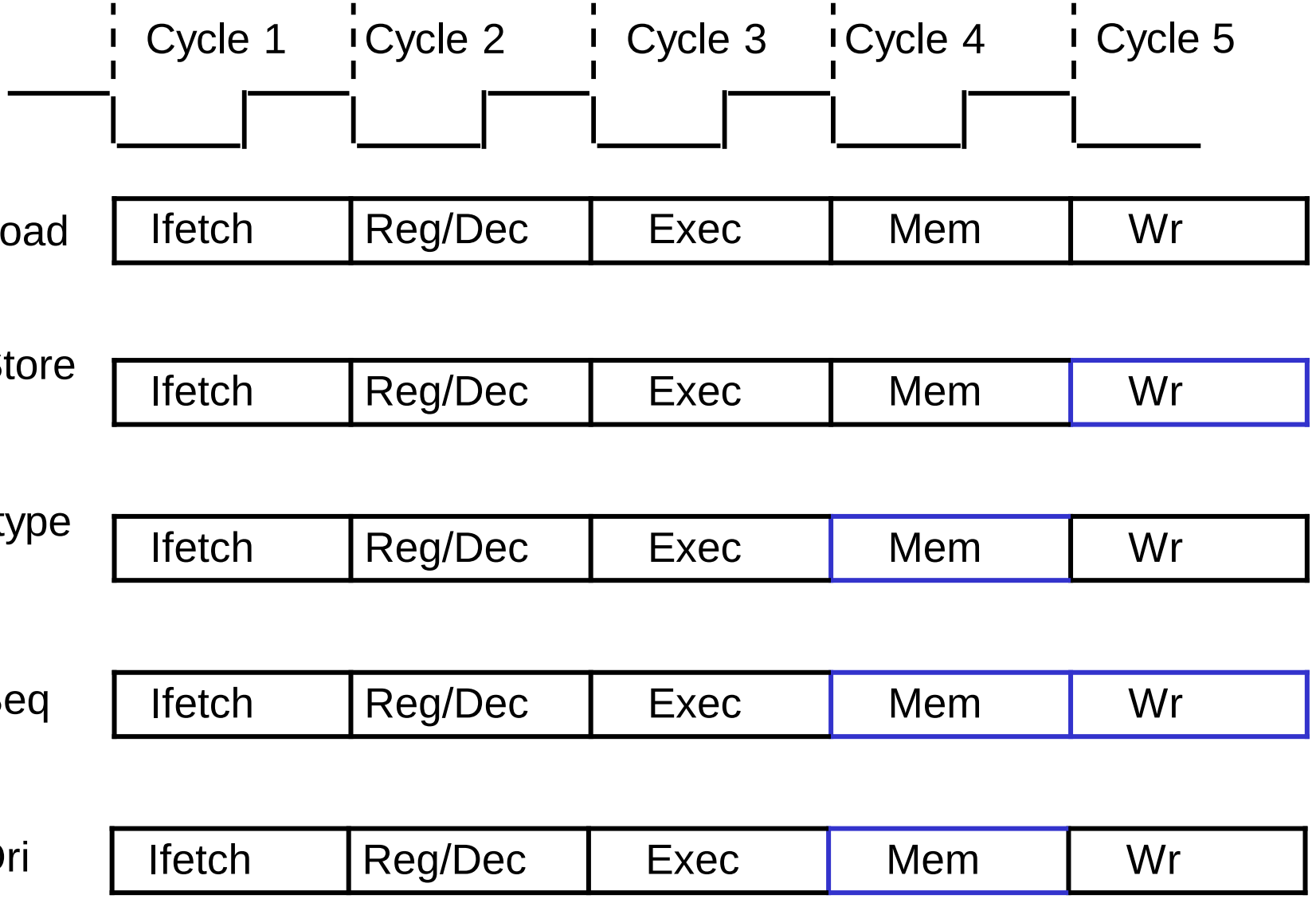
➔ Mem stage is a NOOP stage: nothing is being done.



Modified Control & Datapath



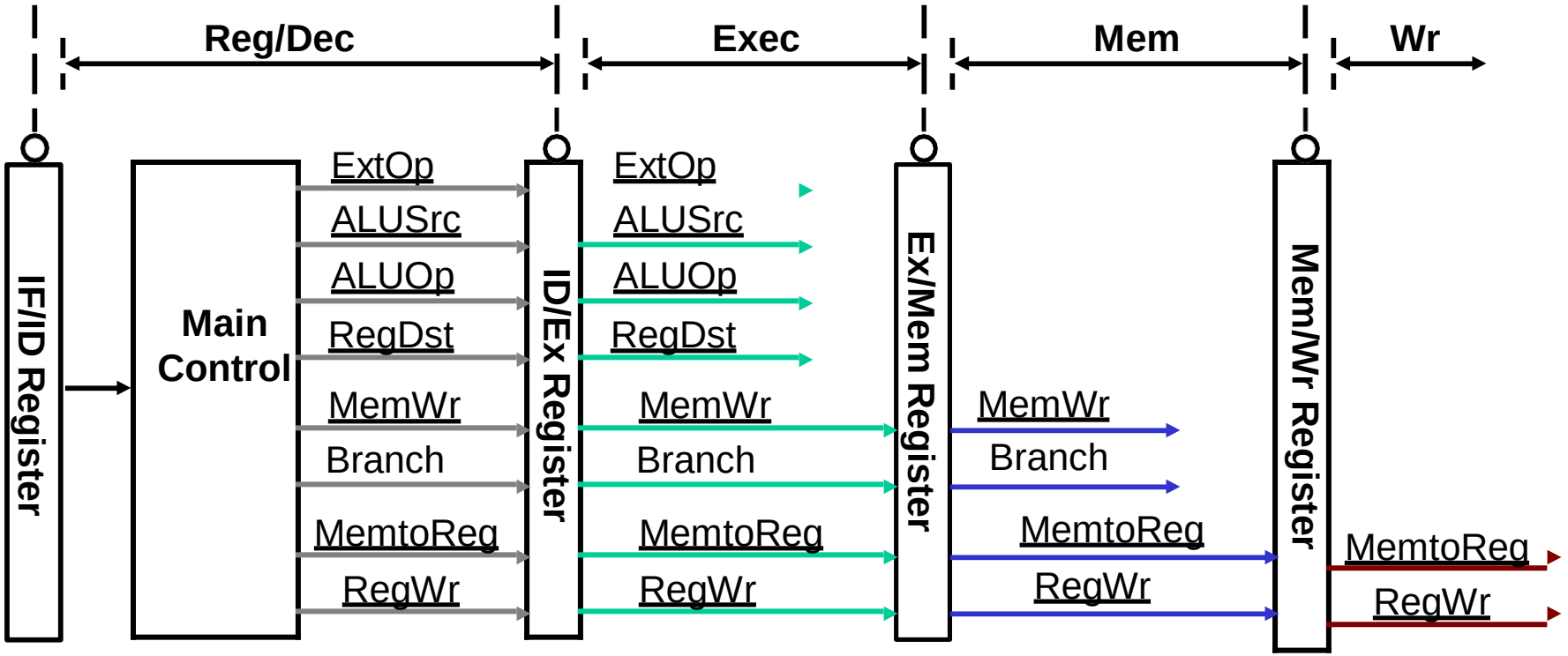
Standardized Five Stages Instructions



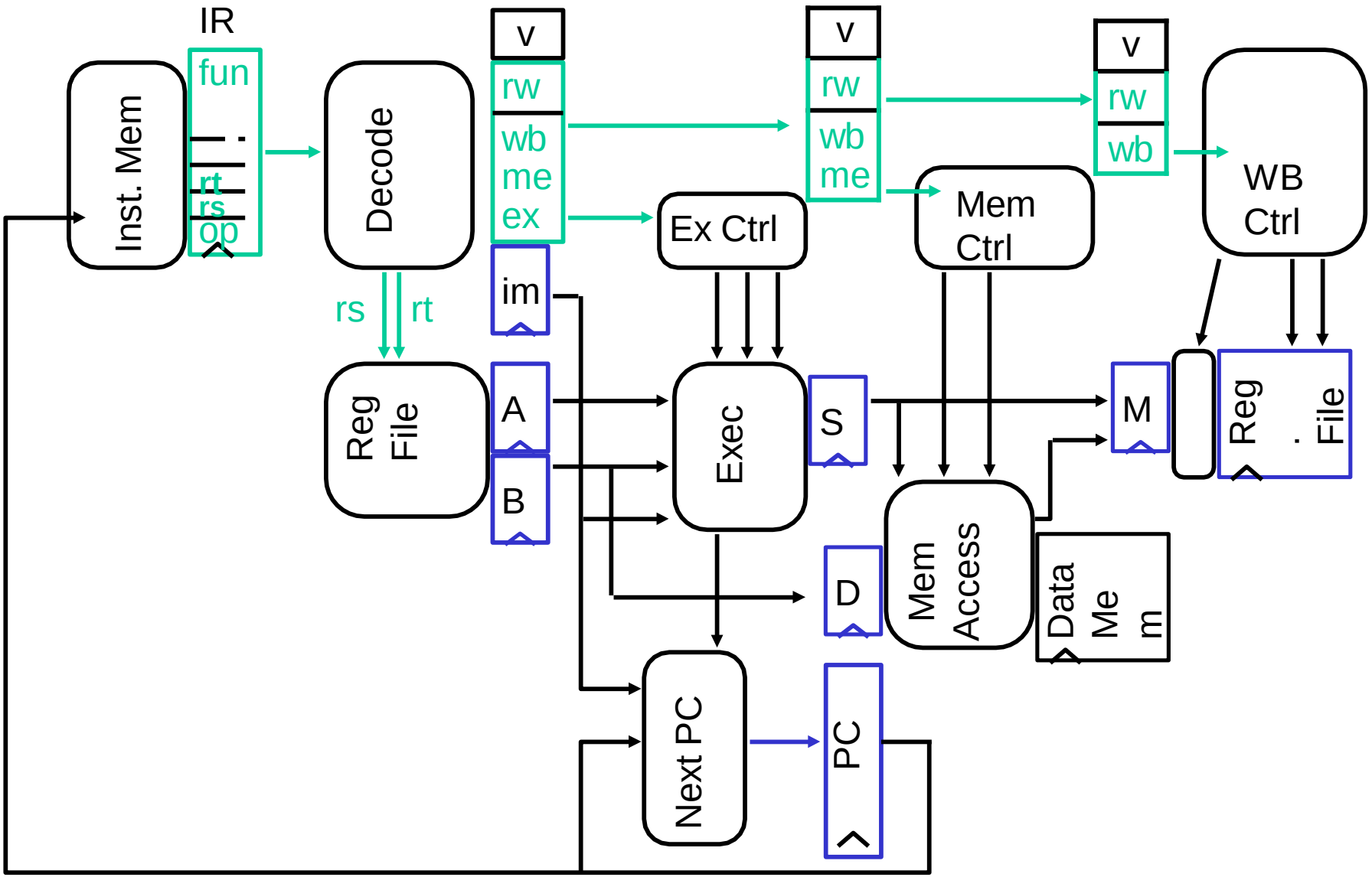
Data Stationary Control

The Main Control generates the control signals during Reg/Dec

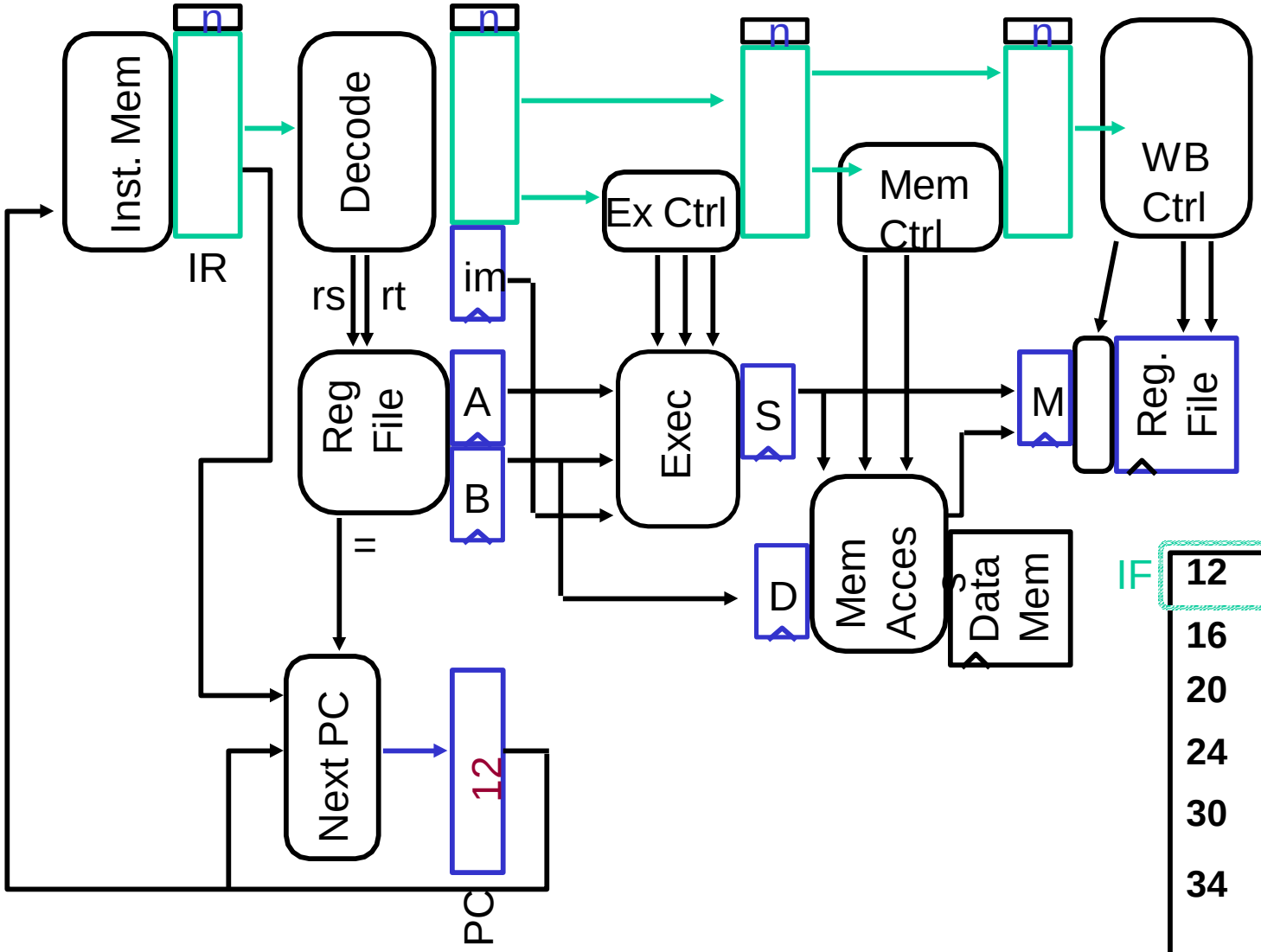
- ➔ Control signals for Exec (ExtOp, ALUSrc, ...) are used 1 cycle later
- ➔ Control signals for Mem (MemWr Branch) are used 2 cycles later
- ➔ Control signals for Wr (MemtoReg MemWr) are used 3 cycles later



Datapath + Data Stationary Control



An example

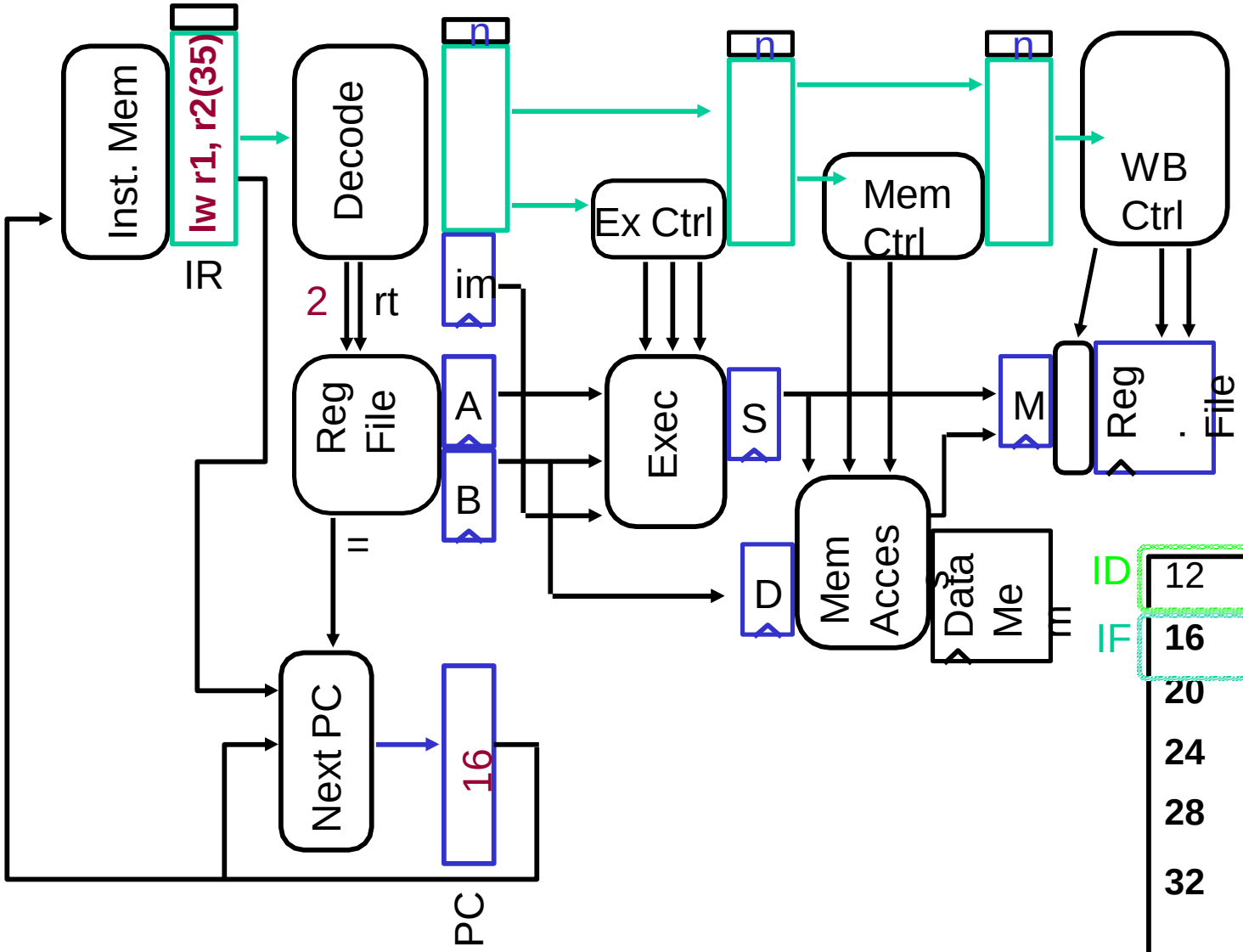


IF

12	lw	r1, r2(35)
16	addI	r2, r2, 3
20	sub	r3, r4, r5
24	beq	r6, r7, 100
30	ori	r8, r9, 17
34	add	r10, r11, r12
128	and	r13, r14, 15

Start: Fetch 12

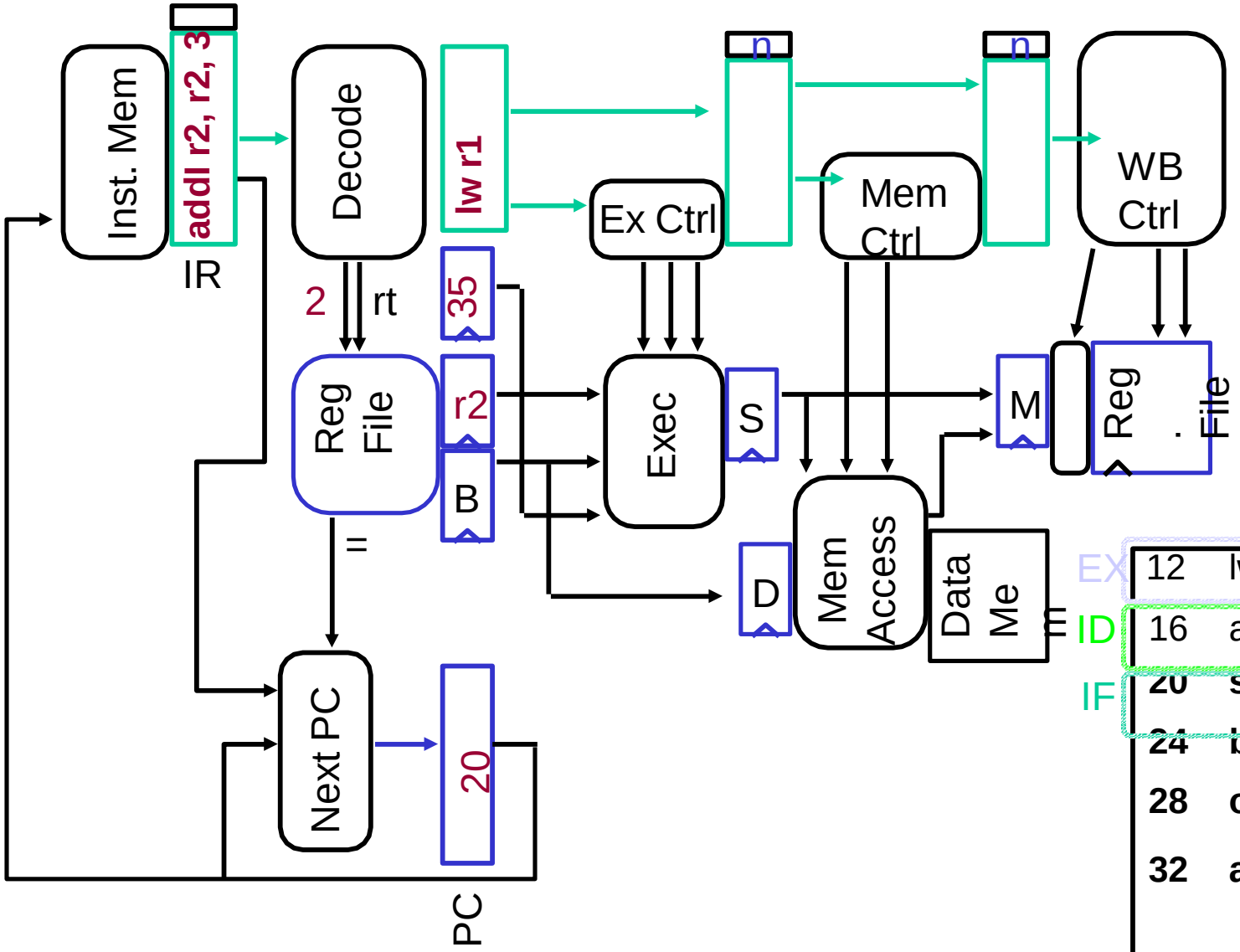
An example



ID	12	lw	r1, r2(35)
IF	16	addI	r2, r2, 3
	20	sub	r3, r4, r5
	24	beq	r6, r7, 100
	28	ori	r8, r9, 17
	32	add	r10, r11, r12
	128	and	r13, r14, 15

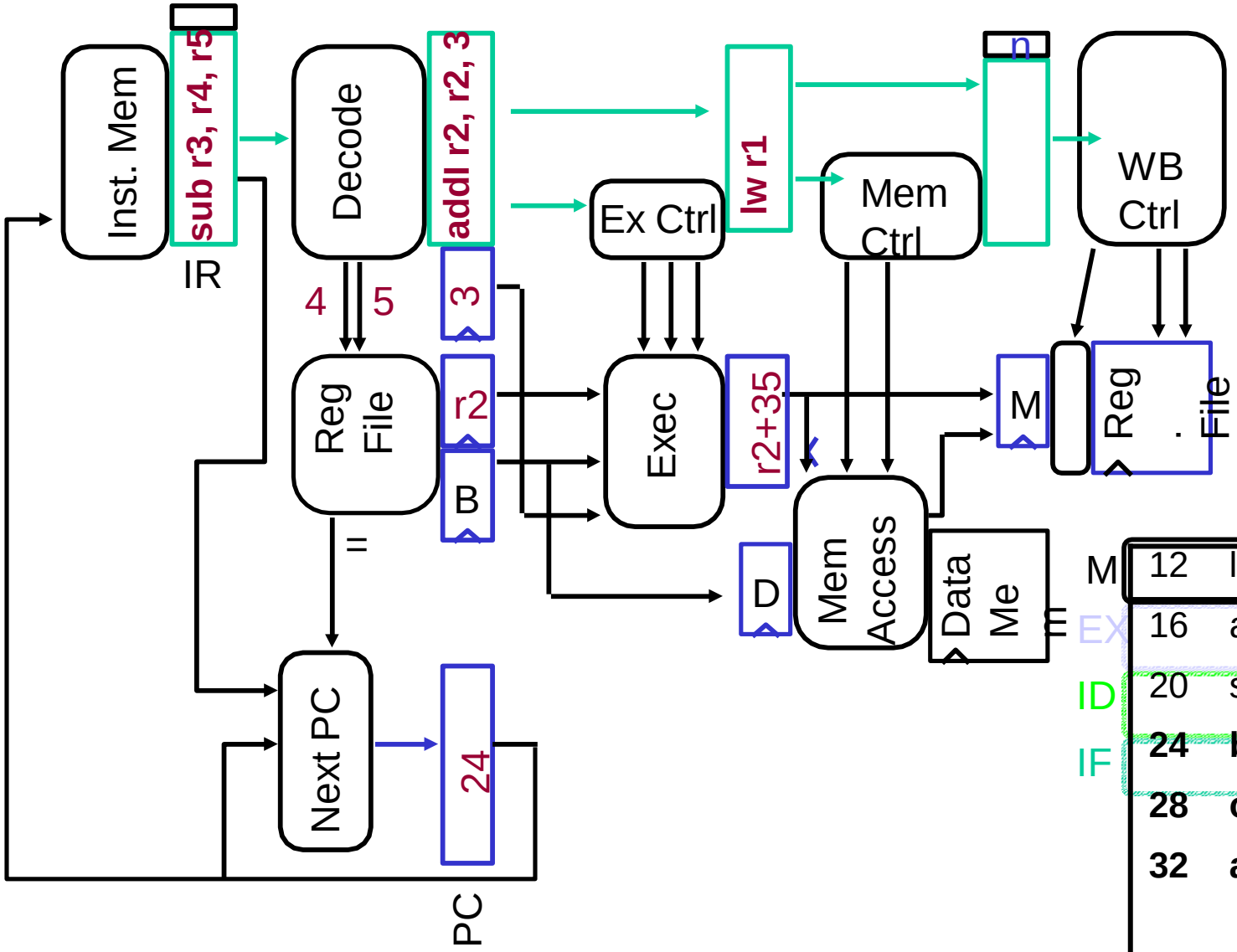
Fetch 16, Decode 12

An example



Fetch 20, Decode 16, Exec 12

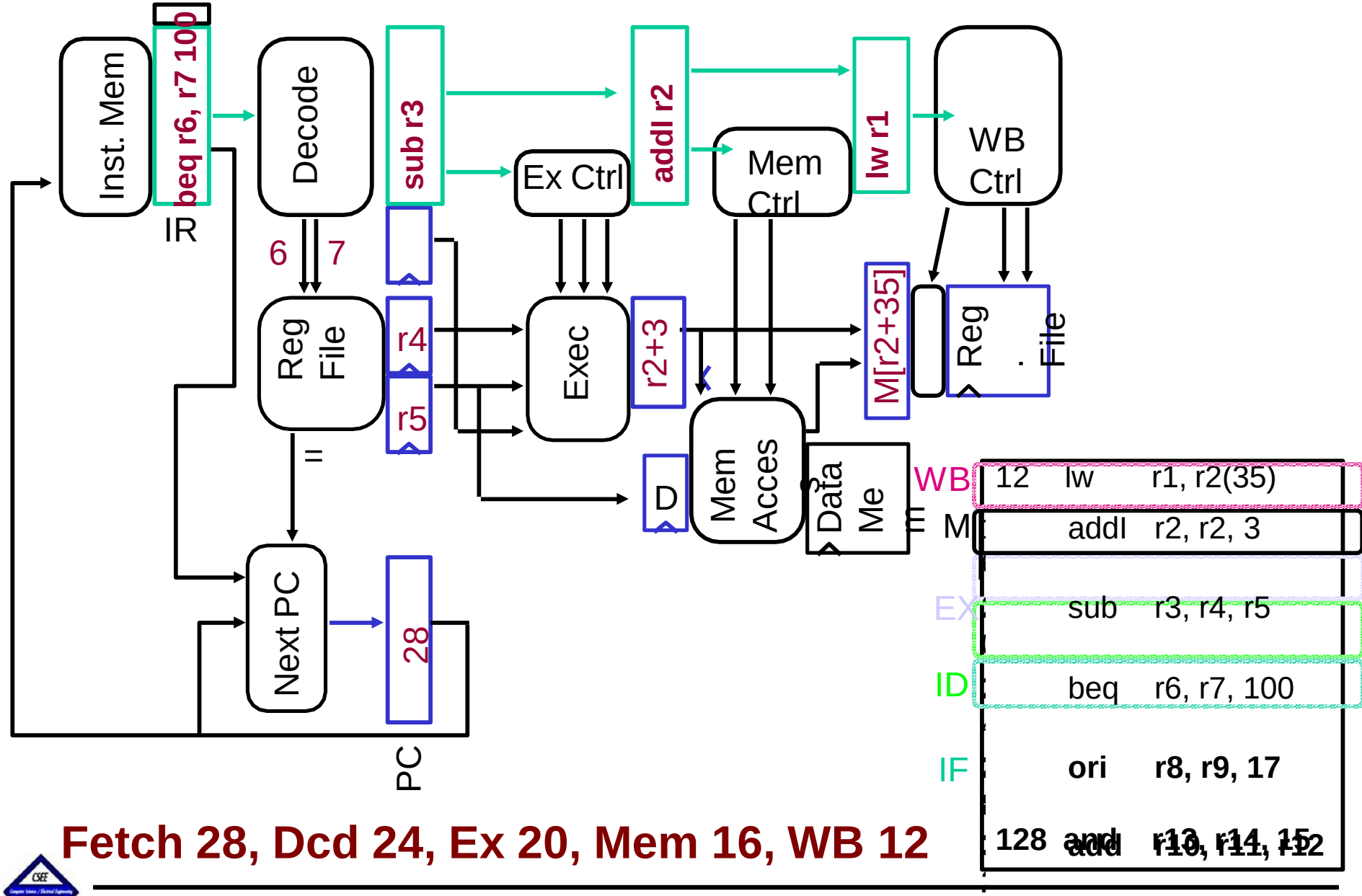
An example



	12	lw	r1, r2(35)
	16	addI	r2, r2, 3
	20	sub	r3, r4, r5
	24	**beq**	**r6, r7, 100**
	28	**ori**	**r8, r9, 17**
	32	**add**	**r10, r11, r12**
	128	**and**	**r13, r14, 15**

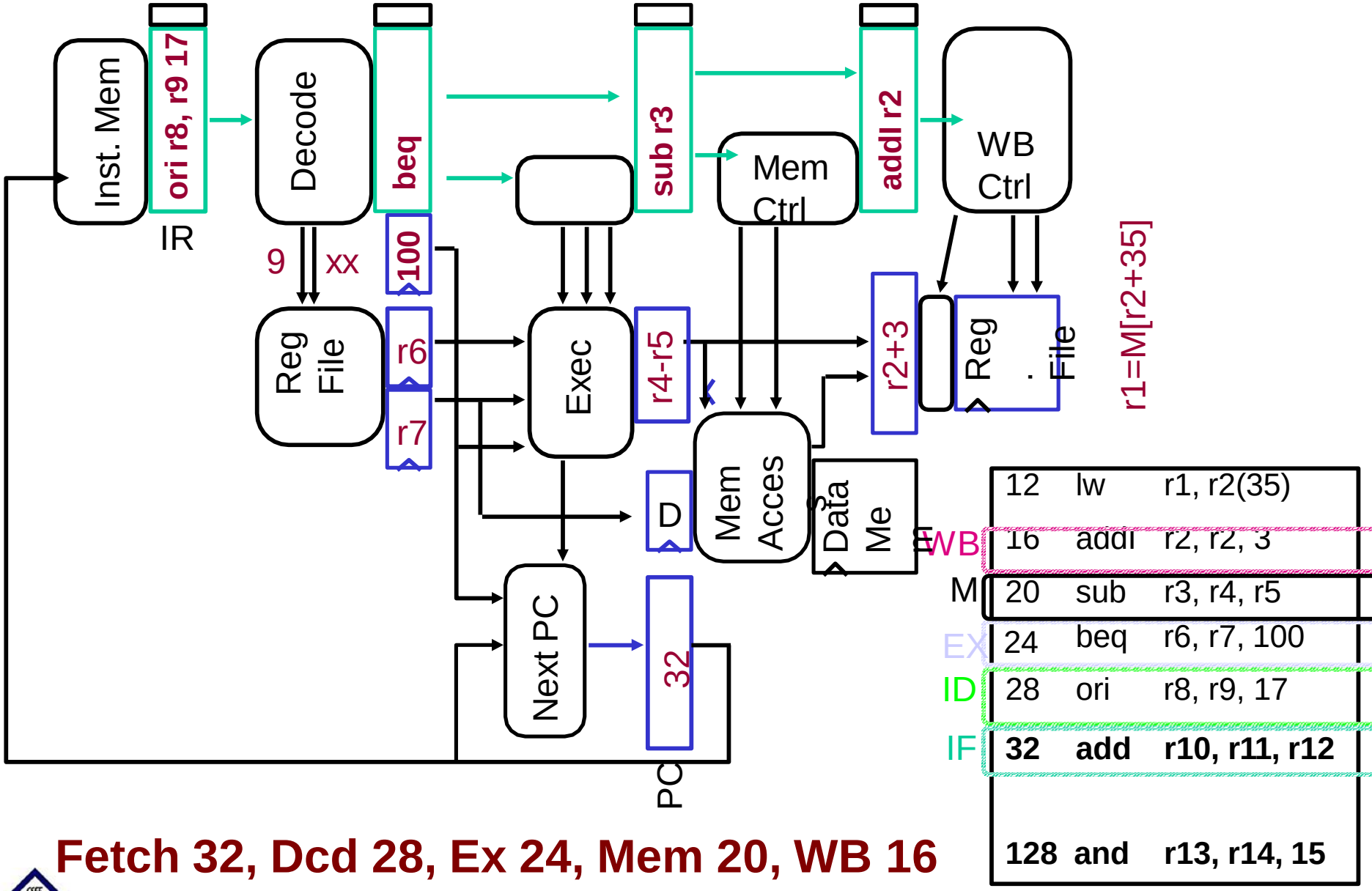
Fetch 24, Decode 20, Exec 16, Mem 12

An example



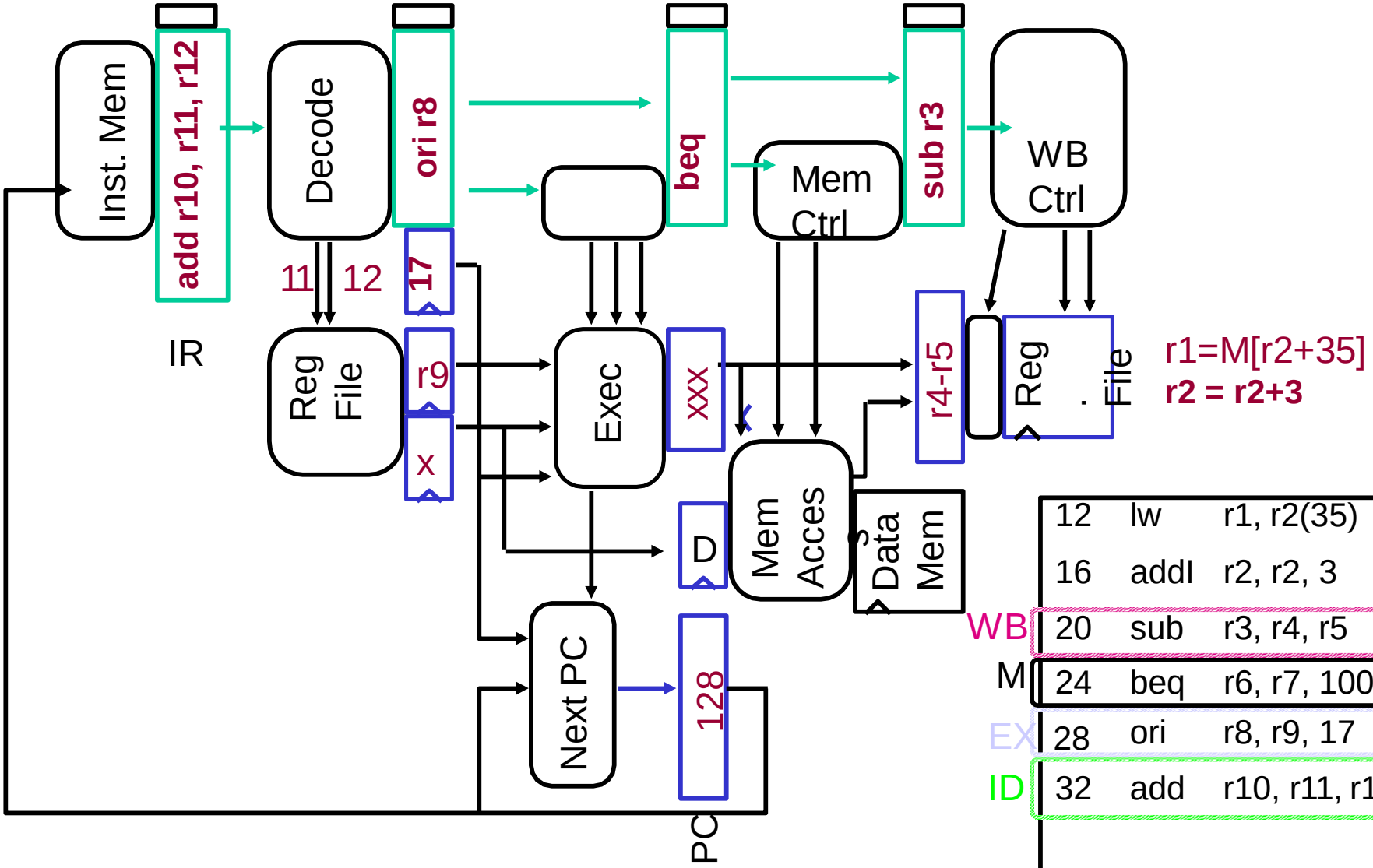
Fetch 28, Dcd 24, Ex 20, Mem 16, WB 12

An example



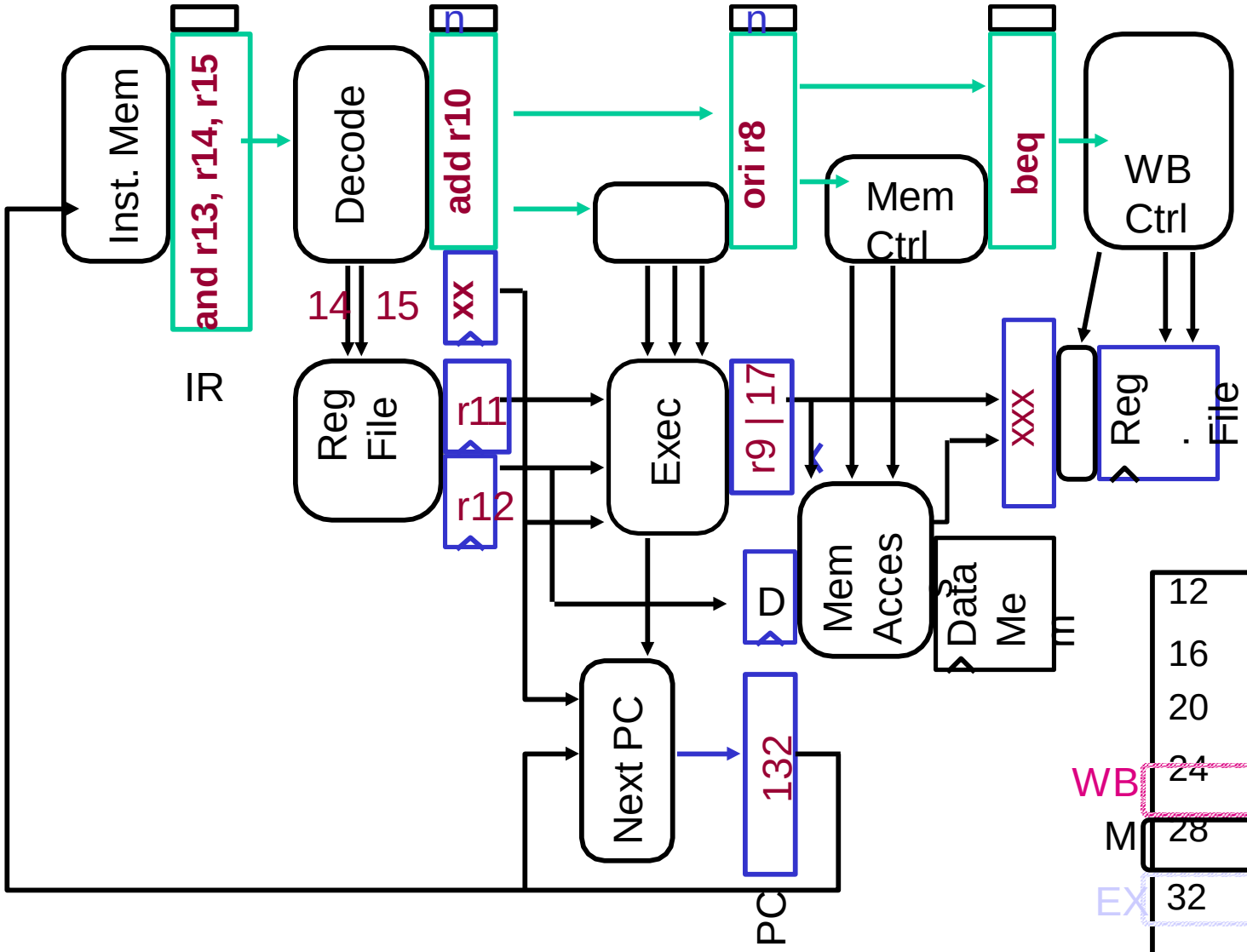
Fetch 32, Dcd 28, Ex 24, Mem 20, WB 16

An example



Fetch 128, Dcd 32, Ex 28, Mem 24, WB 20

An example



r1=M[r2+35]

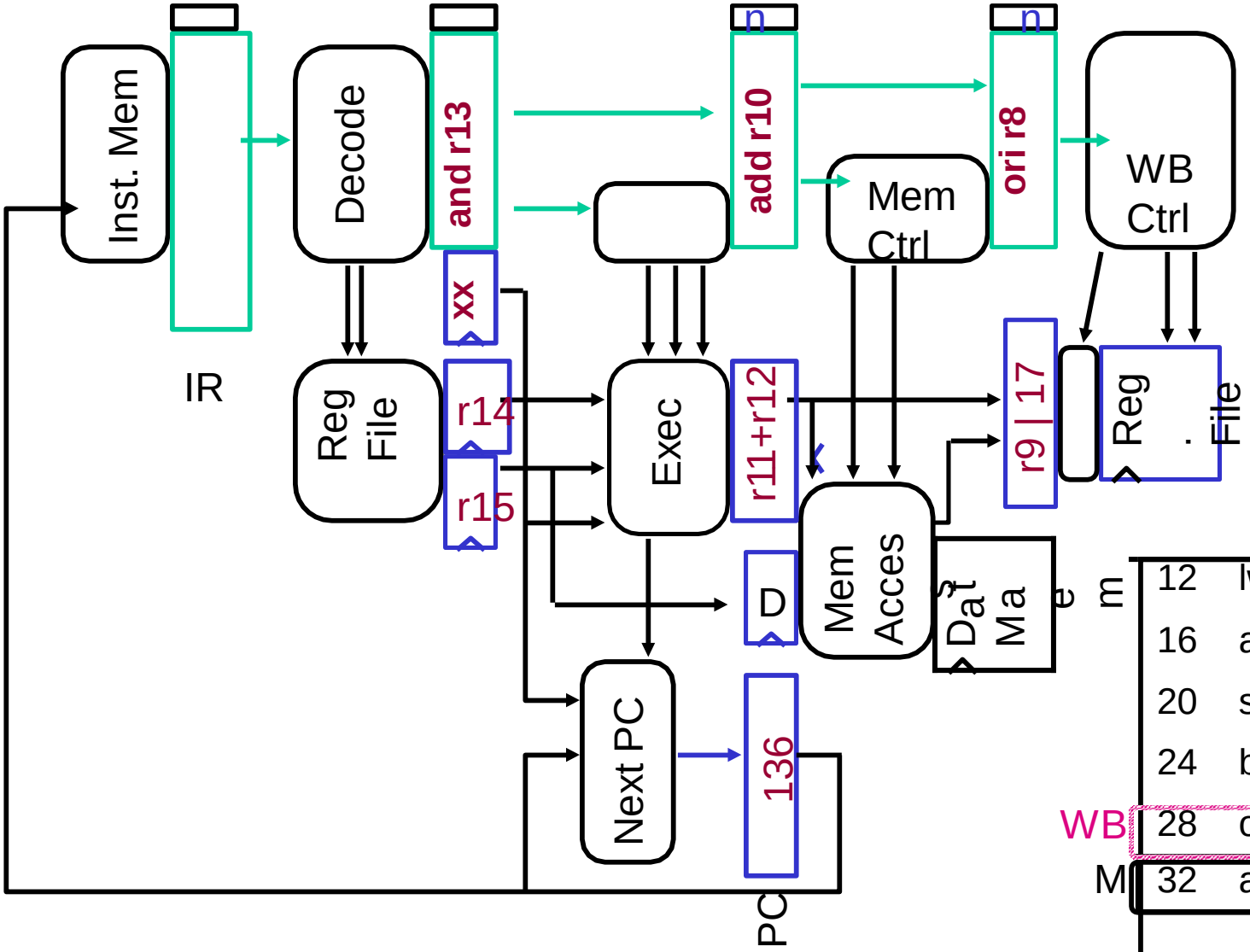
r2 = r2+3

r3 = r4-r5

	12	lw	r1, r2(35)
	16	addI	r2, r2, 3
	20	sub	r3, r4, r5
WB	24	beq	r6, r7, 100
M	28	ori	r8, r9, 17
EX	32	add	r10, r11, r12
ID	**128**	**and**	**r13, r14, 15**

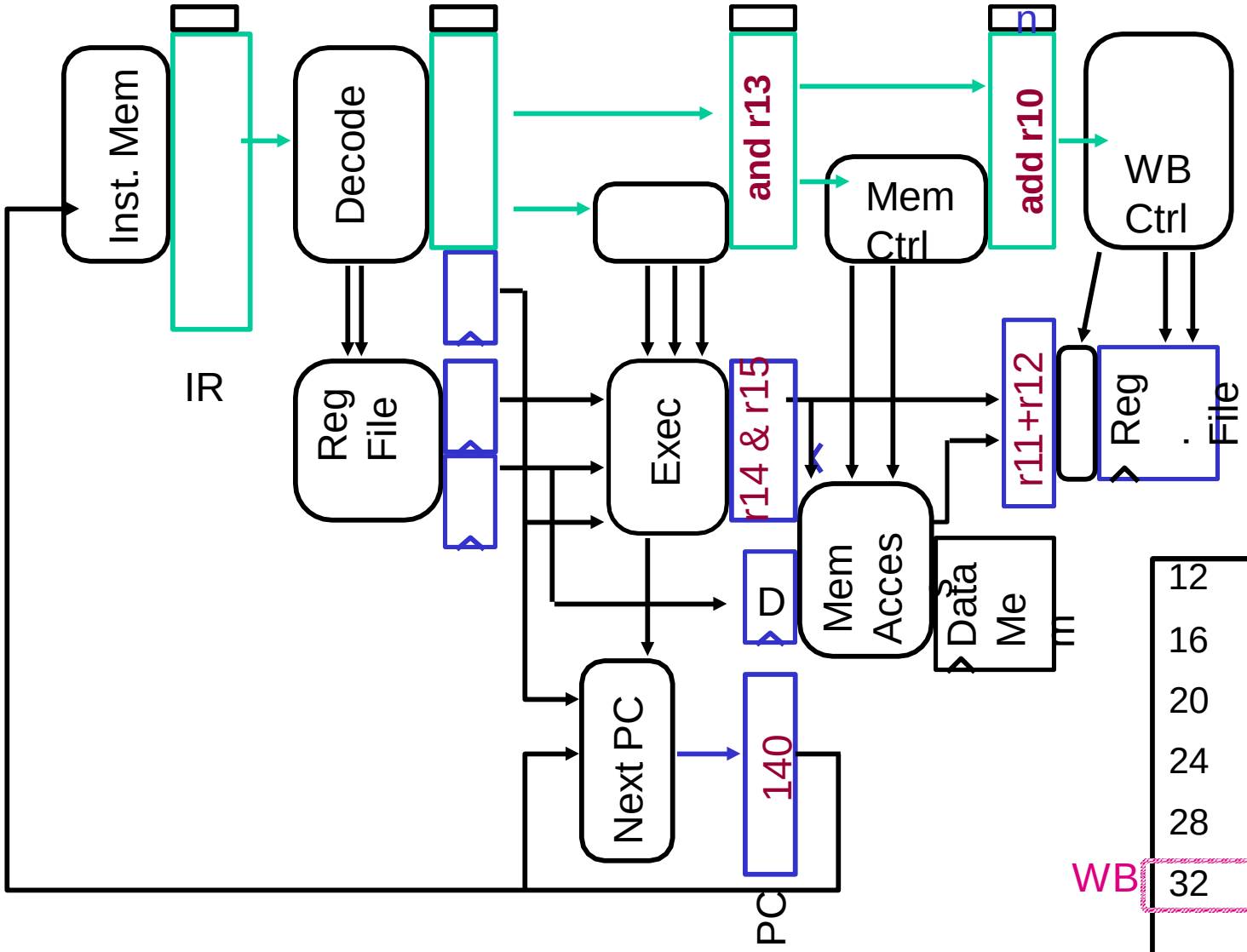
Fetch 132, Dcd 100, Ex 32, Mem 28, WB 24

An example



Fetch 136, Dcd 104, Ex 100, Mem 32, WB 28

An example



r1=M[r2+35]

r2 = r2+3

r3 = r4-r5

	12	lw	r1, r2(35)
	16	addI	r2, r2, 3
	20	sub	r3, r4, r5
	24	beq	r6, r7, 100
	28	ori	r8, r9, 17
WB	32	add	r10, r11, r12
M	128	and	r13, r14, 15

Fetch 140, Dcd 108, Ex 104, Mem 100, WB 32

Conclusion

- ***Summary***
  - ➔ Designing a pipelined datapath
    - Standardized multi-stage instruction execution
    - Unique resources per stage
  - ➔ Pipeline control
    - Simplified stage-based control
    - Detailed working example
- ***Next Lecture***
  - ➔ Pipeline hazard detection
  - ➔ Handling hazard in the pipeline design

Read sections 4.6 in the 5th Ed. or section 4.6 in the 4th Ed. of the textbook