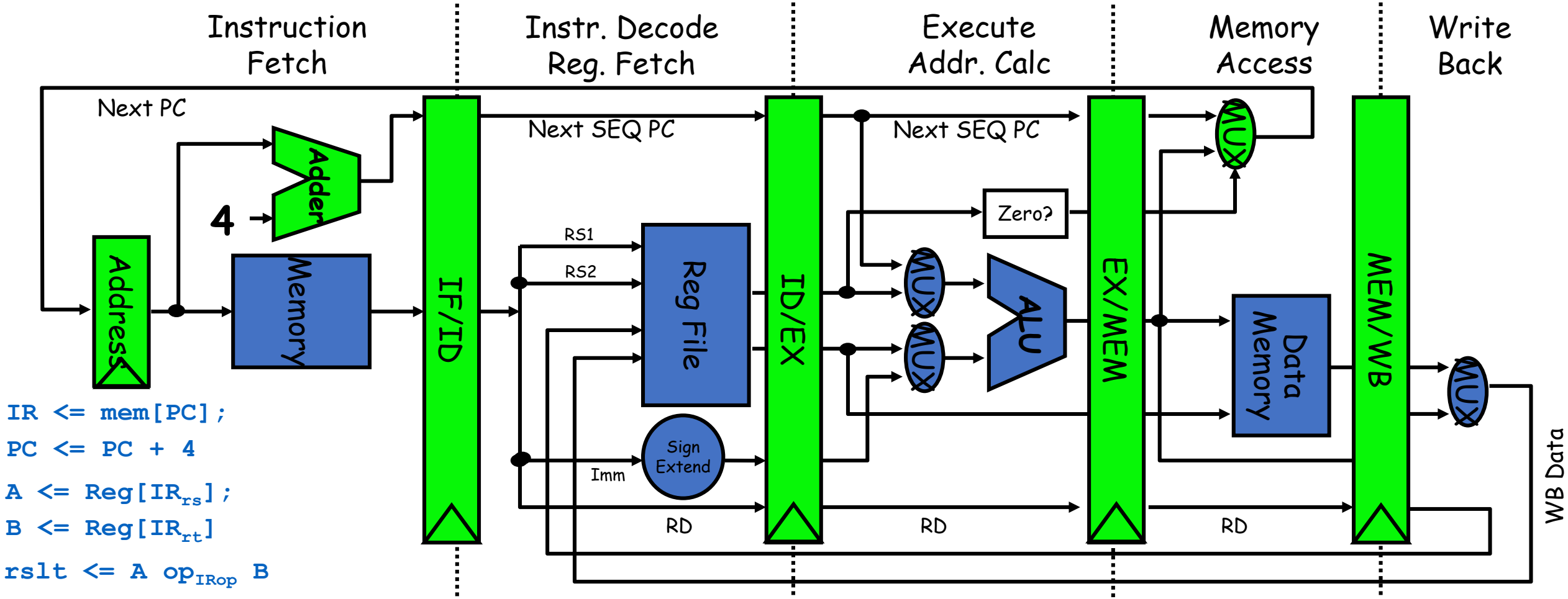
Brushing up the Processor
CS665-Fall 2019
Secure Memory Systems

Biswa@CSE-IITK

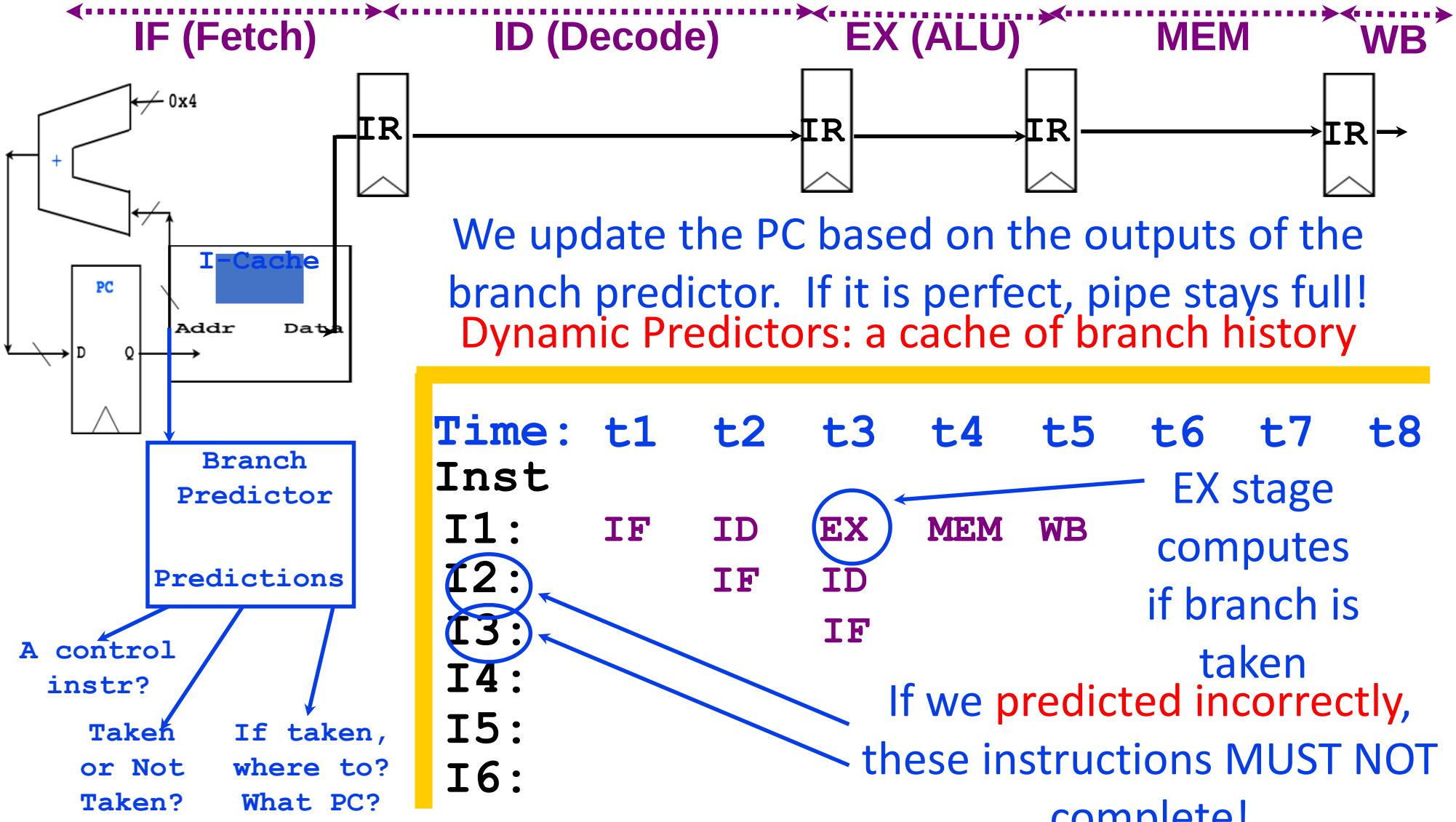
Vanilla Pipeline




```
IR <= mem[PC];
PC <= PC + 4
A <= Reg[IR_rs];
B <= Reg[IR_rt]
rslt <= A op_IRop B
WB <= rslt
Reg[IR_rd] <= WB
```

- Data stationary control
  - local decode for each instruction phase / pipeline stage

Branch Prediction and Speculative Execution



We update the PC based on the outputs of the branch predictor. If it is perfect, pipe stays full!

Dynamic Predictors: a cache of branch history

Time:	t1	t2	t3	t4	t5	t6	t7	t8
Inst								
I1:	IF	ID	EX	MEM	WB			
I2:		IF	ID					
I3:			IF					
I4:								
I5:								
I6:								

EX stage computes if branch is taken

If we predicted incorrectly, these instructions MUST NOT complete!

Branch Target Buffer

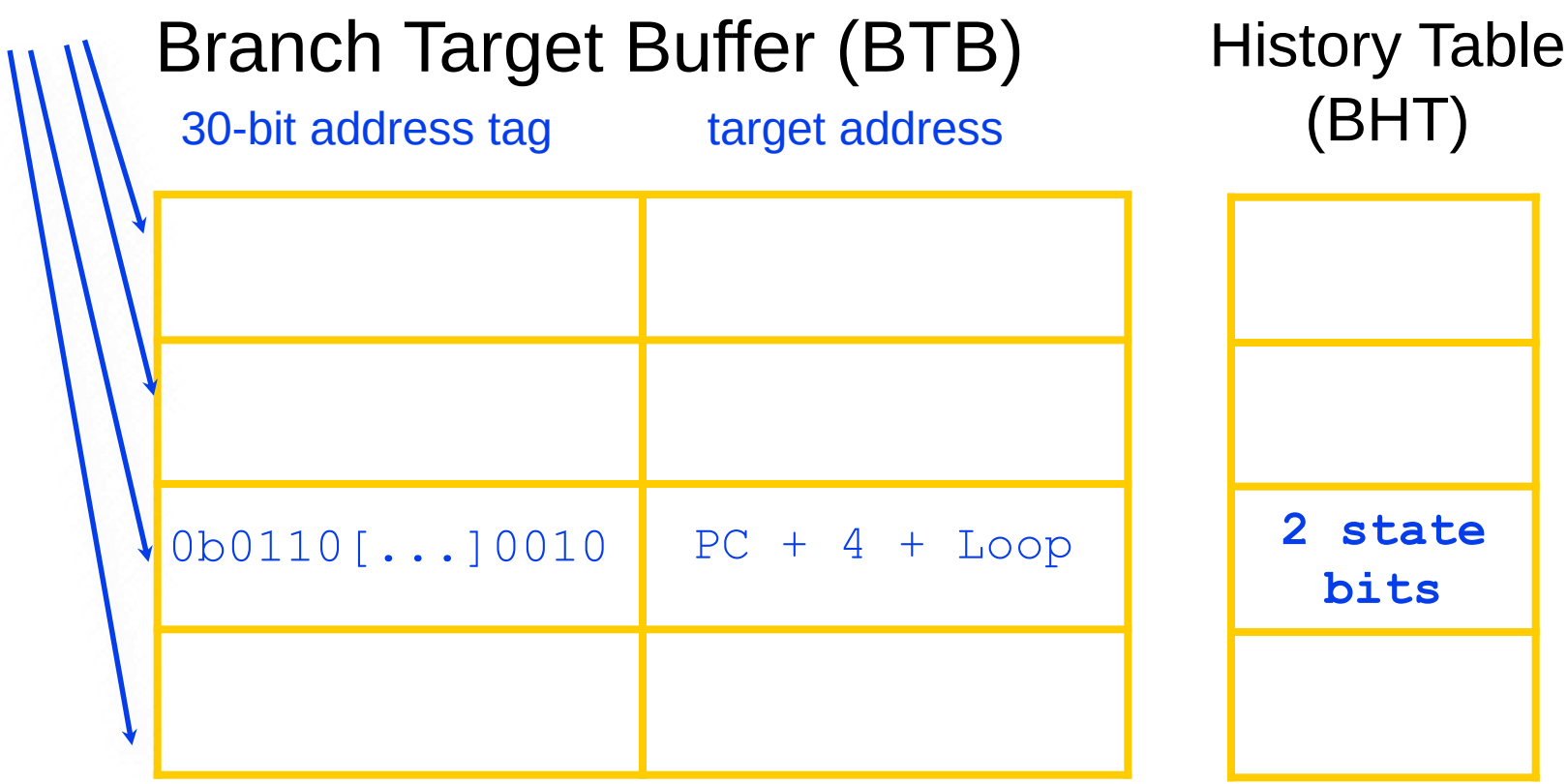
Address of branch instruction

`0b0110[...]01001000`

30 bits

Branch instruction

`BNEZ R1 Loop`

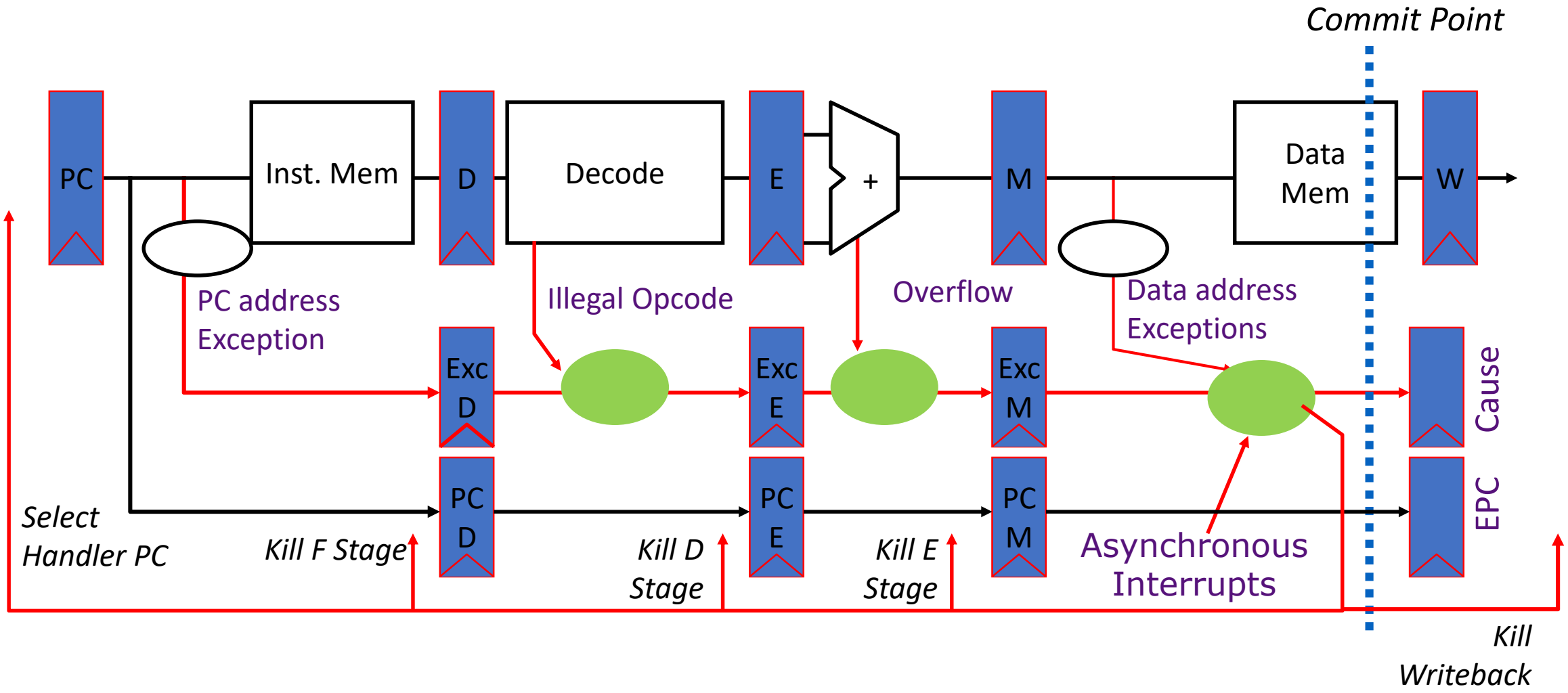


Drawn as fully associative to focus on the essentials.

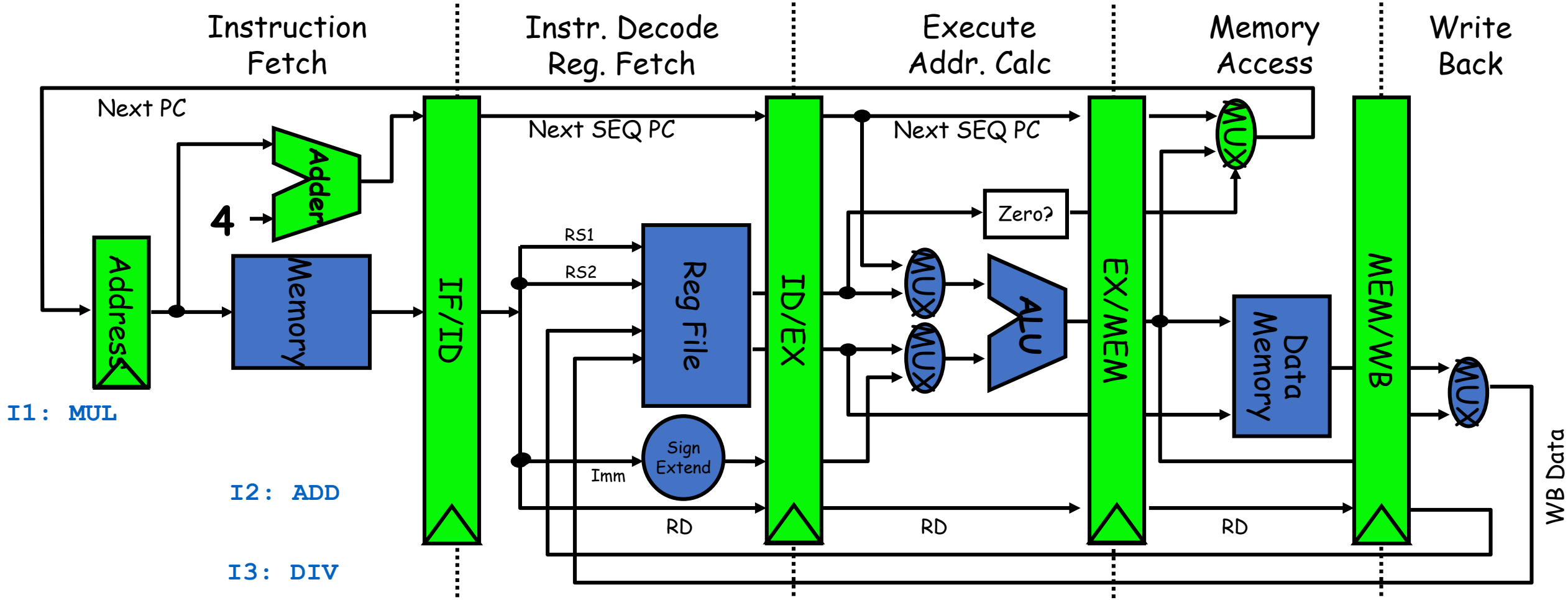
In real designs, always direct-mapped.

At EX stage, update BTB/BHT, kill instructions, if necessary,

Handling Exceptions

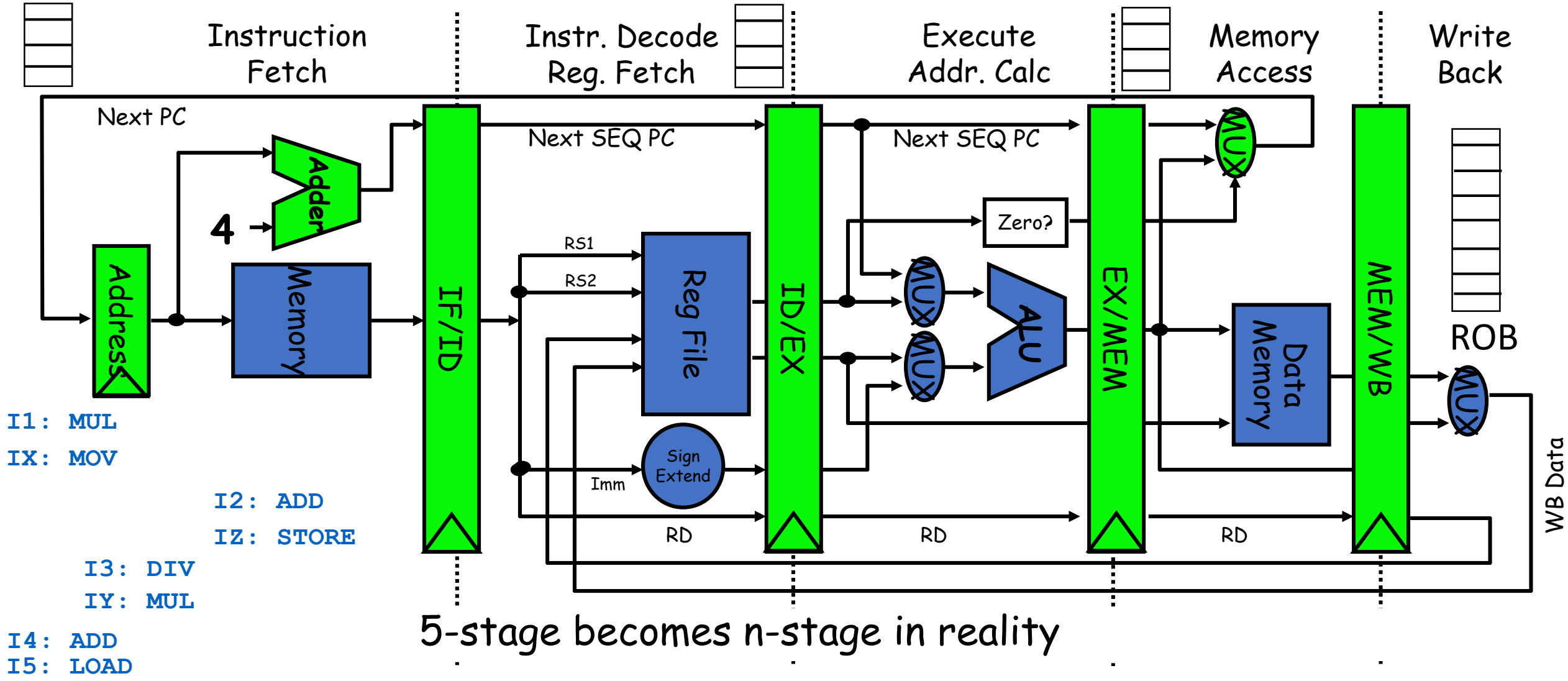


Some More: Out-of-order

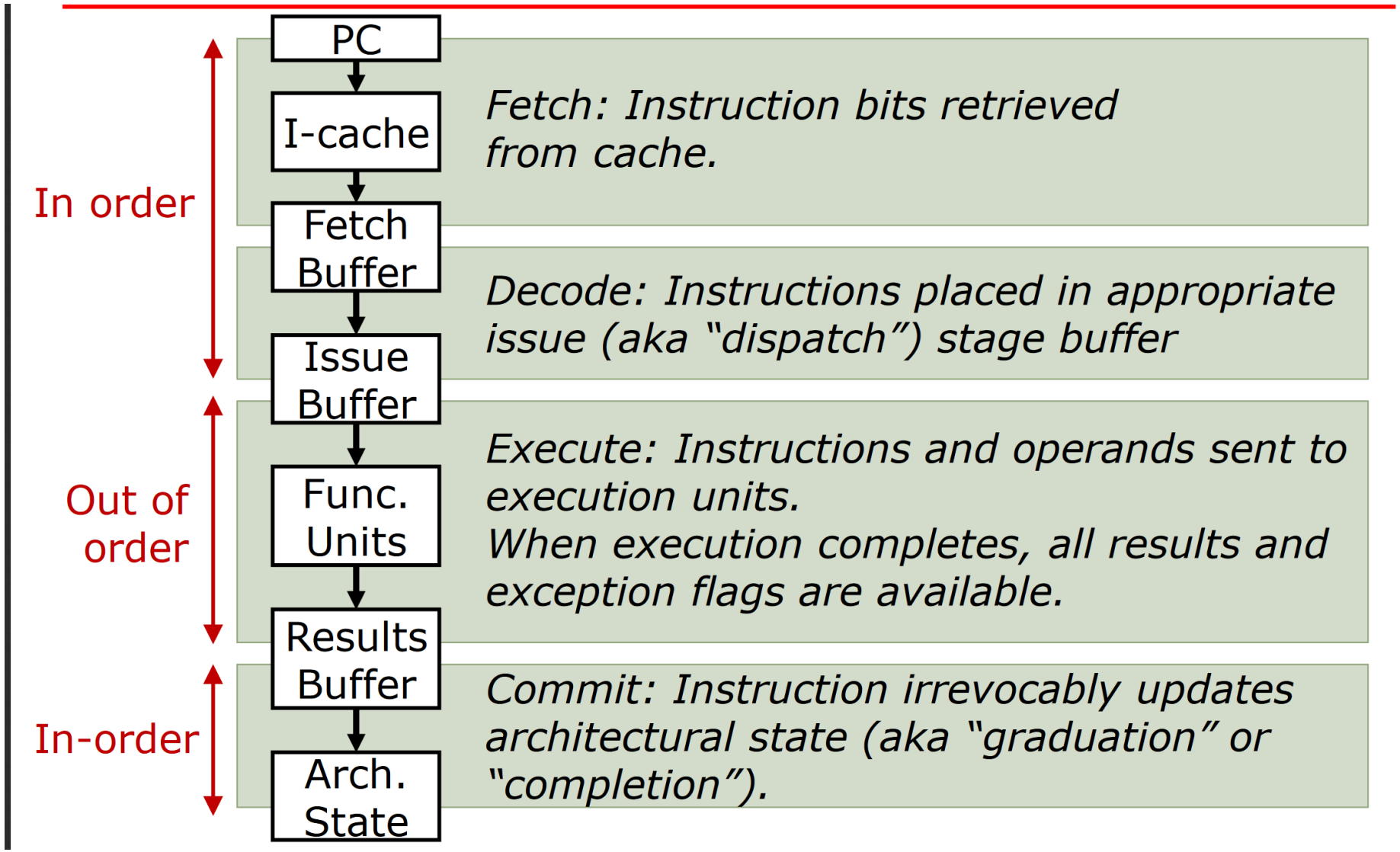


- Data stationary control
  - local decode for each instruction phase / pipeline stage

Some More: Out-of-order + Multi-Issue (a.k.a superscalar)



Life Cycle of Instruction in Execution



What is Speculative Execution?

1. Proceed ahead despite unresolved dependencies using a prediction for an architectural or micro-architectural value

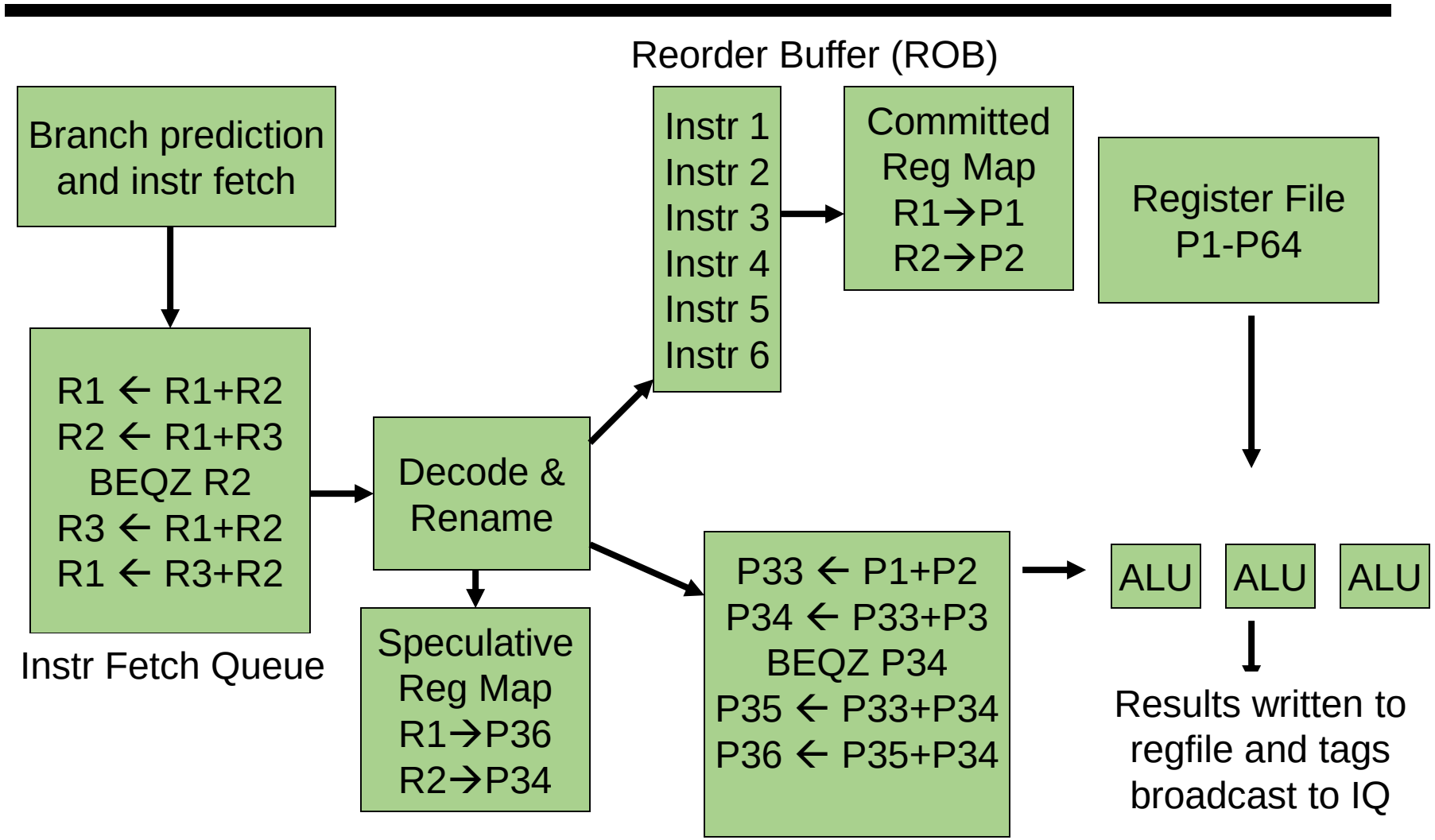
2. Maintain both old and new values on updates to architectural (and often micro-architectural) state

3. After sure that there was no mis-speculation and there will be no more uses of the old values, discard old values and just use new values

OR

3. In event of mis-speculation, dispose of all new values, restore old values, and re-execute from point before mis-speculation

Superscalar Processor + Speculative Execution



TLP: Multithreading