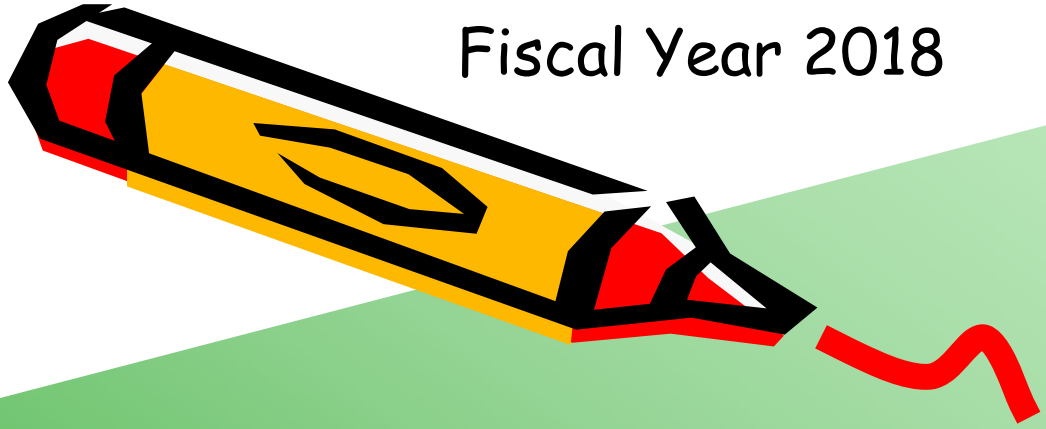


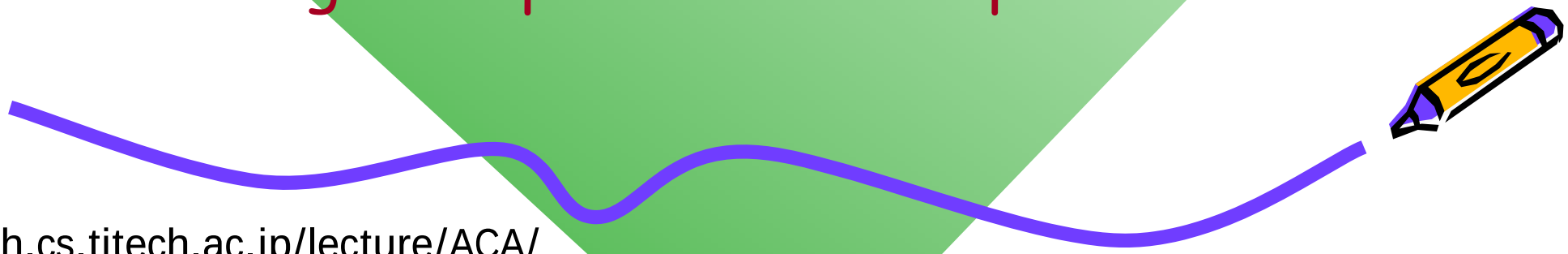
Fiscal Year 2018

Ver. 2019-01-10a

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

9. Instruction Level Parallelism: Exploiting ILP Using Multiple Issue and Speculation



www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 13:20-14:50, Thr 13:20-14:50

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Hardware register renaming

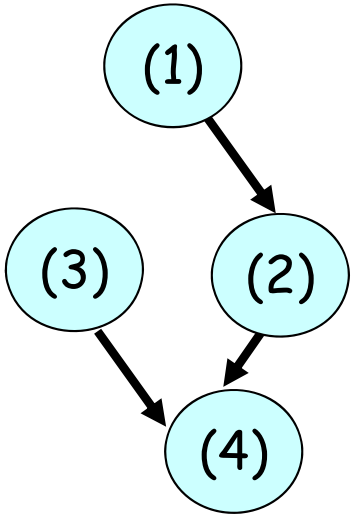
- Logical registers (architectural registers) which are ones defined by ISA
  - $0, $1, … $31
- Physical registers
  - Assuming plenty of registers are available, p0, p1, p2, …
- A processor renames (converts) each logical register to a unique physical register dynamically

Typical instruction pipeline of scalar processor

Typical instruction pipeline of high-performance superscalar processor

Out-of-order execution

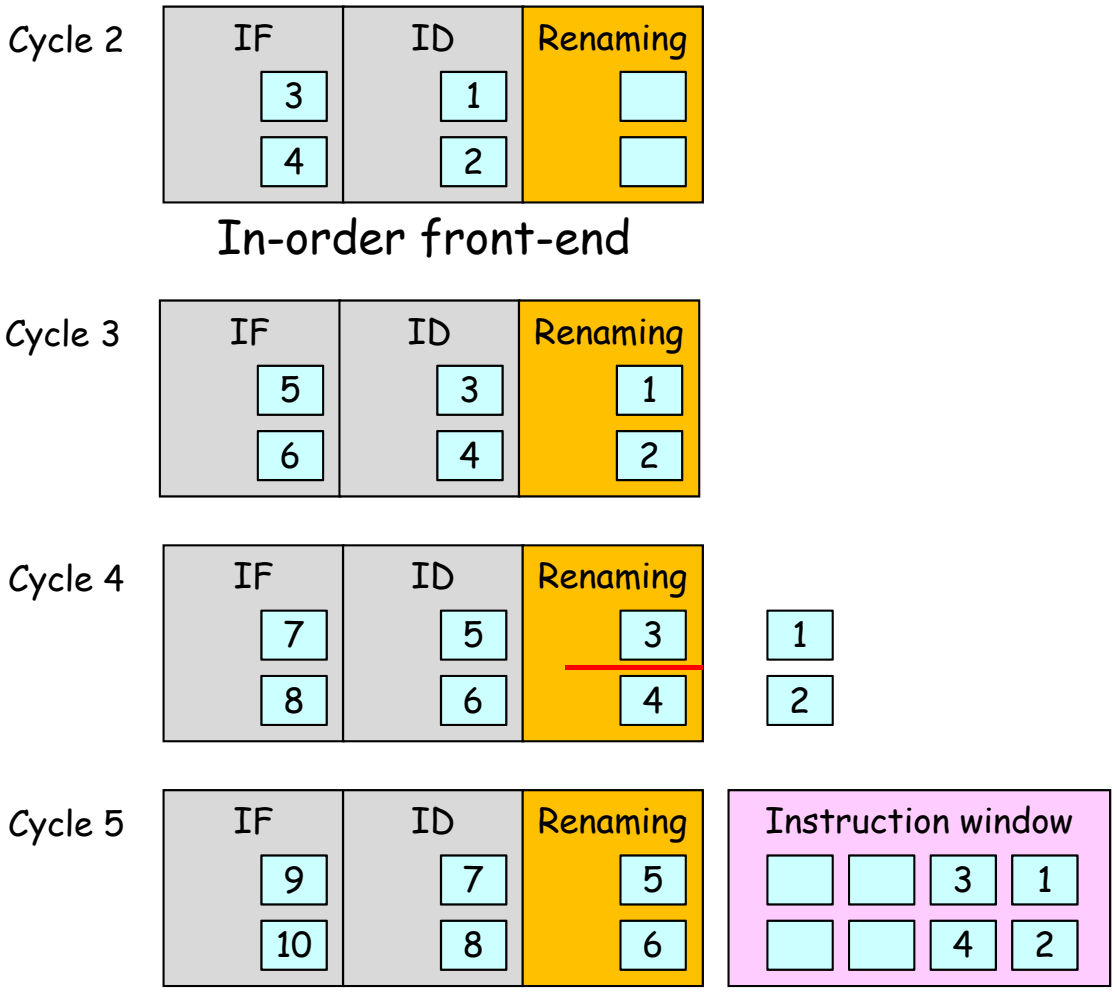
- In in-order execution model, all instructions executed in the order that they appear. This can lead to unnecessary stalls.
  - Instruction (3) stalls waiting for insn (2) to go first, even though it does not have a data dependence.
- Using register renaming to eliminate output dependence and antidependence, just having true data dependence
- With out-of-order execution, insn (3) is allowed to executed before the insn (2)
  - Scoreboarding (CDC6600 in 1964)
  - Tomasulo algorithm (IBM System/360 Model 91 in 1967)



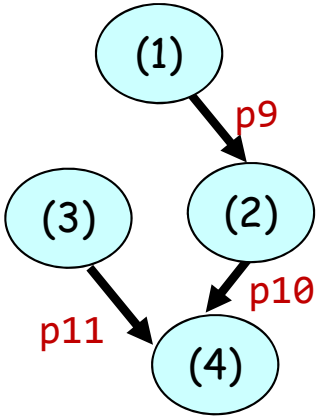
Data flow graph

The key idea for OoO execution (1/3)

- In-order front-end, OoO execution core, in-order retirement using instruction window and reorder buffer (ROB)




```
I1: sub p9,p1,p2
I2: add p10,p9,p3
I3: or  p11,p4,p5
I4: and p12,p10,p11
```



Data flow graph

The key idea for OoO execution (2/3)

- In-order front-end, OoO execution core, in-order retirement using instruction window and reorder buffer (ROB)

Cycle 5

IF	ID	Renaming	Instruction window			
9	7	5			3	1
10	8	6			4	2

Cycle 6

IF	ID	Renaming	Instruction window				Issue
11	9	7			6	5	1
12	10	8			4	2	3

We assume that I1 can be issued at cycle 6 by dependence.

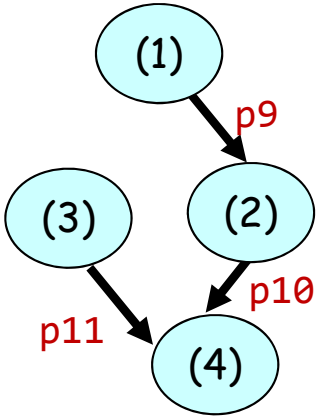
Cycle 7

IF	ID	Renaming	Instruction window				Issue	Execute
13	11	9		8	6	5	2	> 1
14	12	10			4	7		> 3

Cycle 8

IF	ID	Renaming	Instruction window				Issue	Execute	Commit
15	13	11		8	6	5	4	> 2	1
16	14	12		10	9	7		>	3

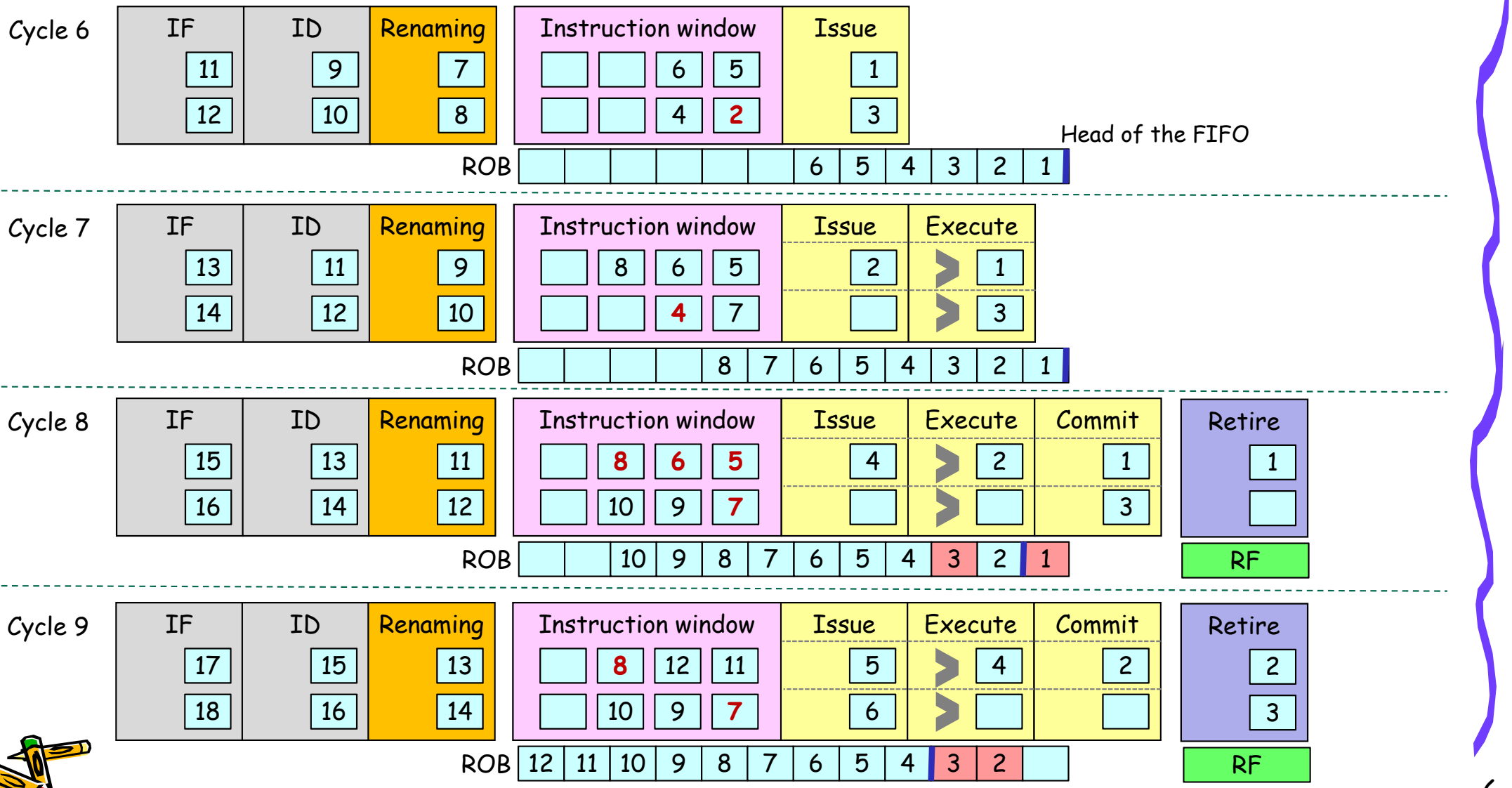
```
I1: sub p9,p1,p2
I2: add p10,p9,p3
I3: or  p11,p4,p5
I4: and p12,p10,p11
```



Data flow graph

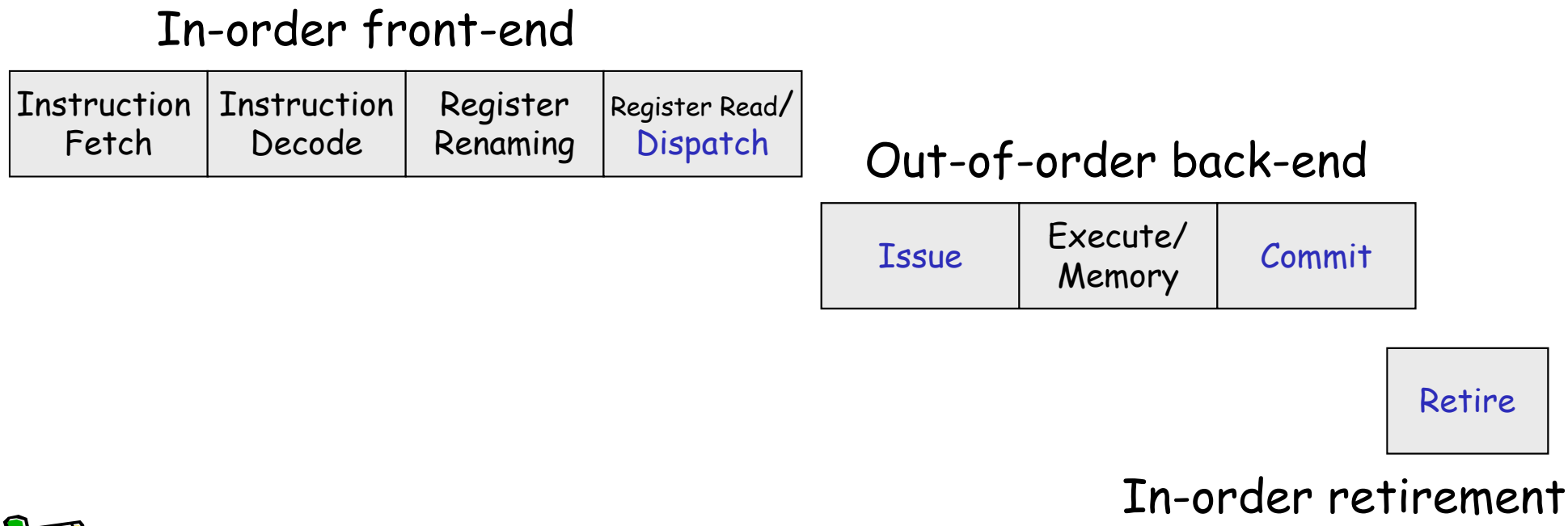
The key idea for OoO execution (3/3)

- In-order front-end, OoO execution core, in-order retirement using instruction window and reorder buffer (ROB)



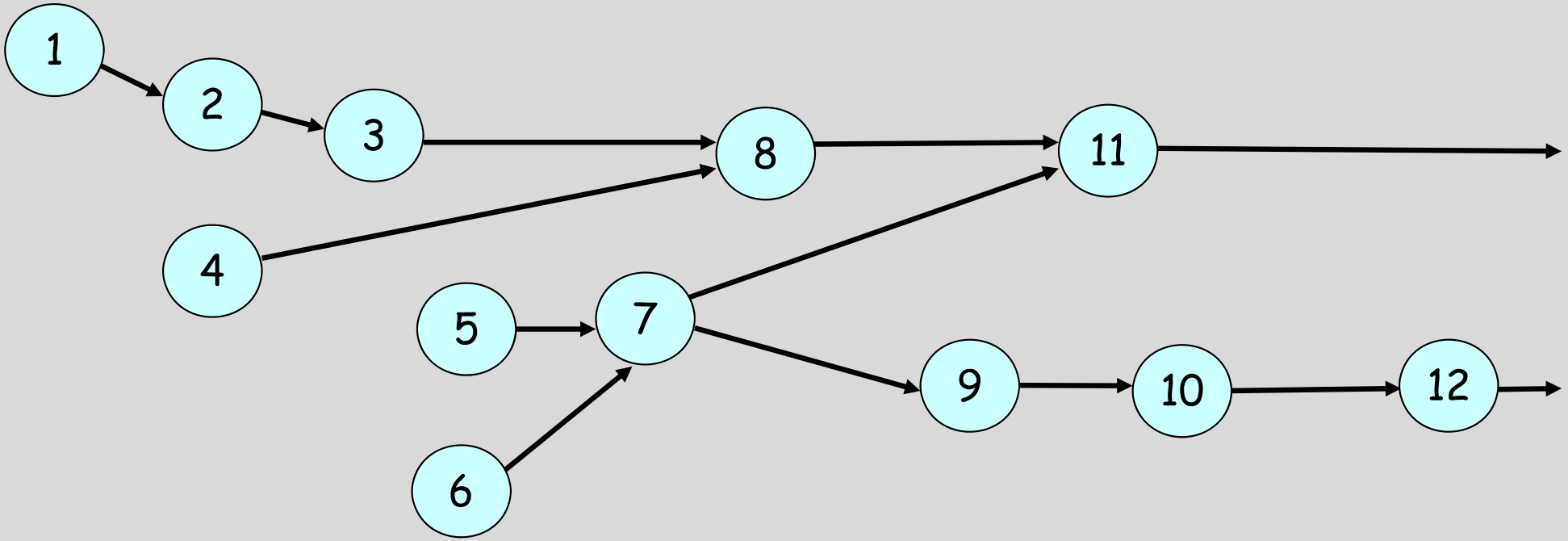
Instruction pipeline of OoO execution processor

- Allocating instructions to instruction window is called dispatch
- Issue or fire wakes up instructions and their executions begin
- In commit stage, the computed values are written back to ROB
- The last stage is called retire or graduate. The result is written back to register file (architectural register file) using a logical register number.

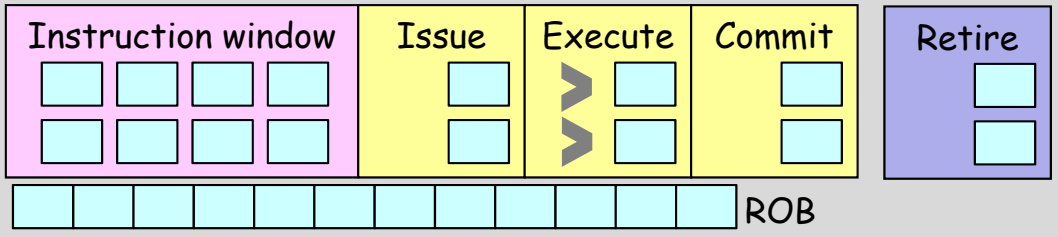


Exercise: OoO execution

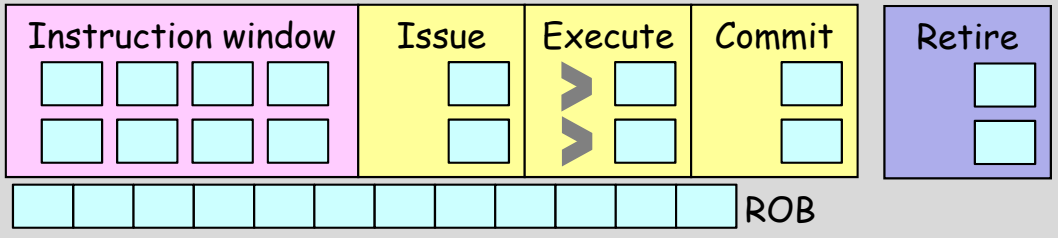
- Draw the cycle by cycle processing behavior of these 12 instructions
  - wakeup
  - select



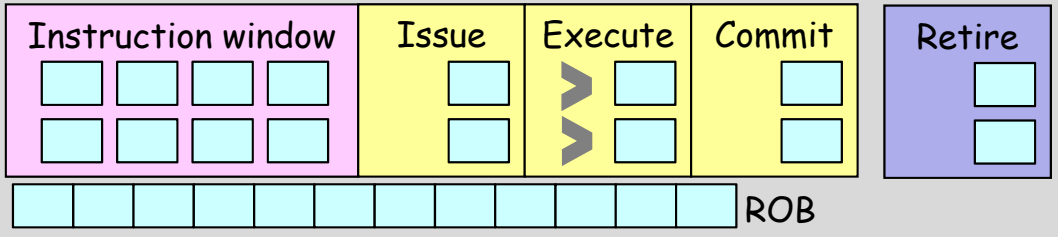
Cycle 1



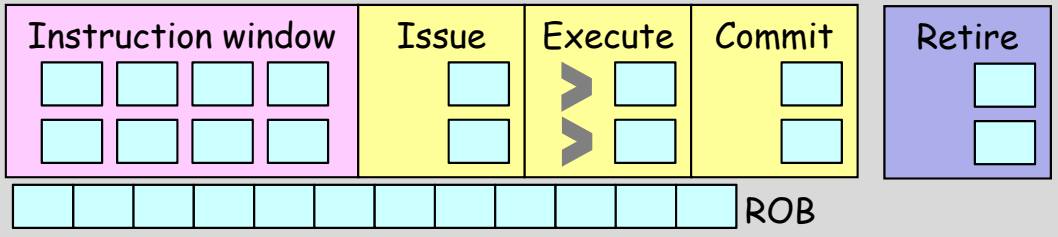
Cycle 2



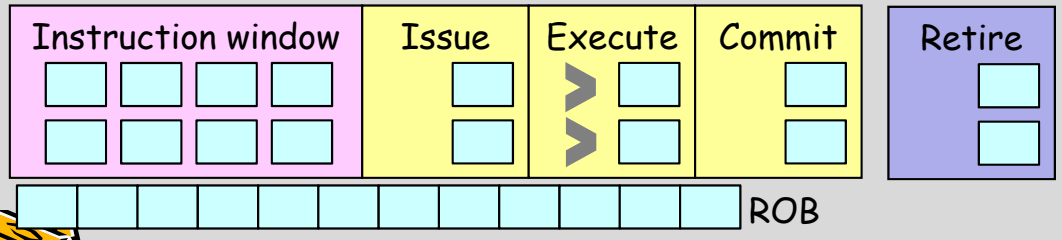
Cycle 3



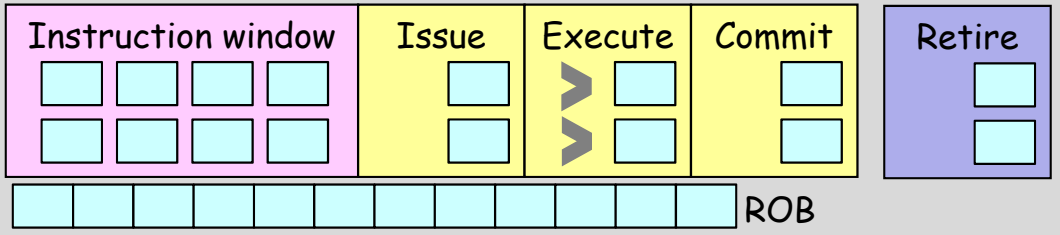
Cycle 4



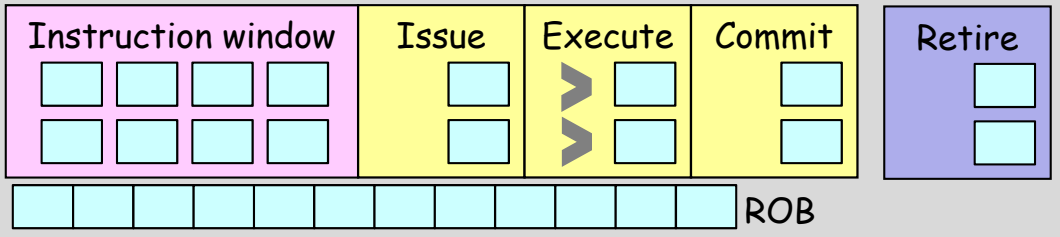
Cycle 5



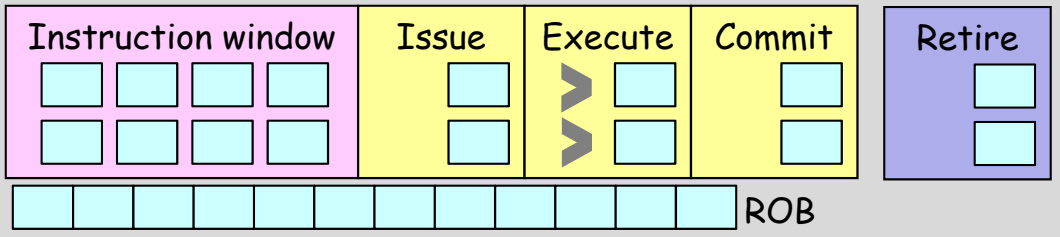
Cycle 6



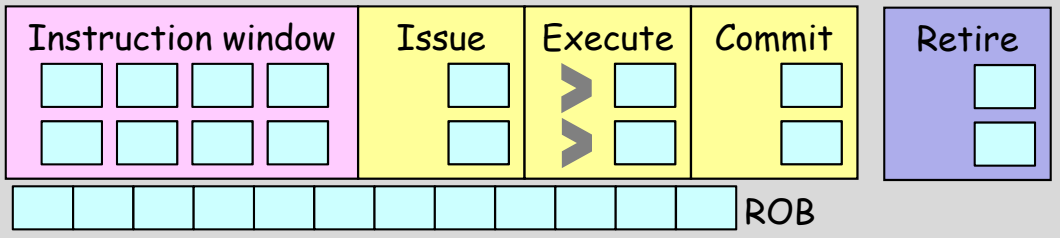
Cycle 7



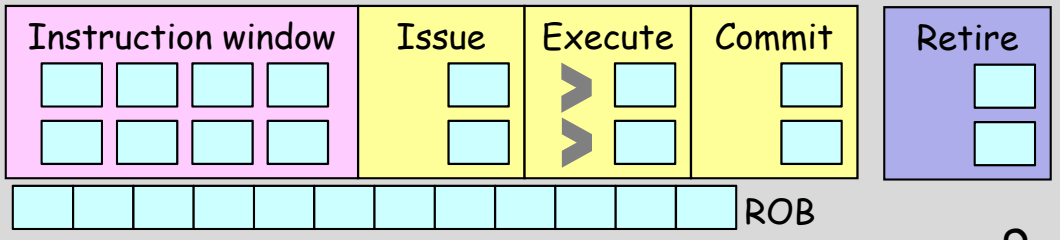
Cycle 8



Cycle 9

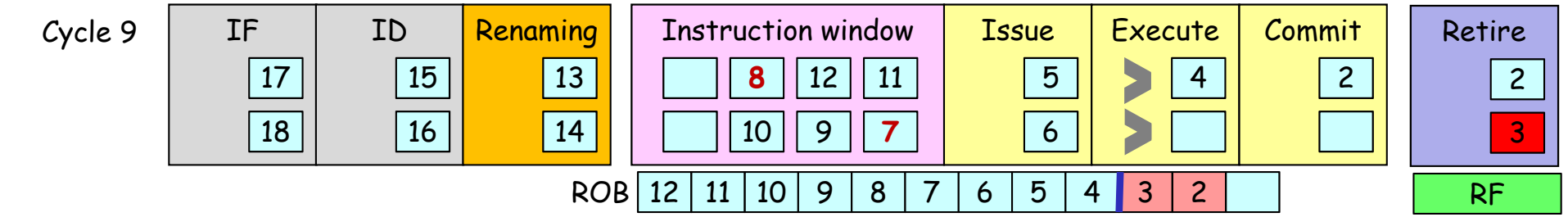


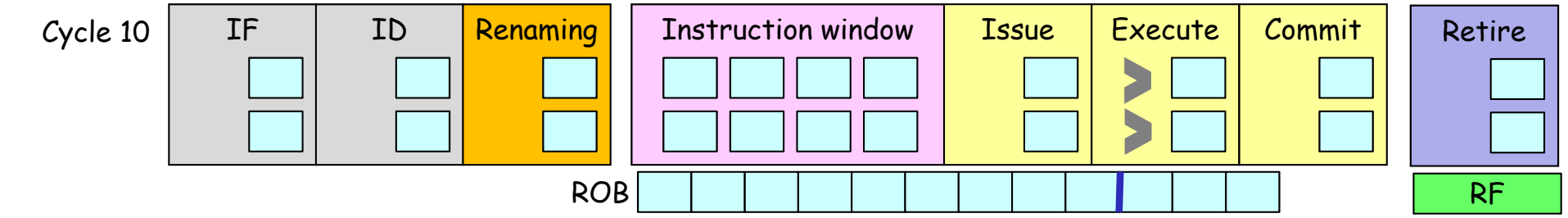
Cycle 10



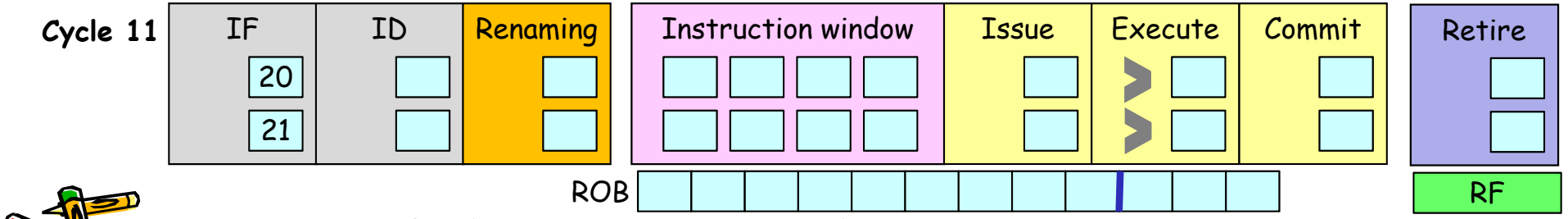
Prediction miss and recovery

- Assume that instruction 3 is a miss predicted branch and its target insn is 20
- Register file (and PC) has the architecture state after insn 3 is executed
- When insn 3 is retired, recover by flushing all instructions and restart





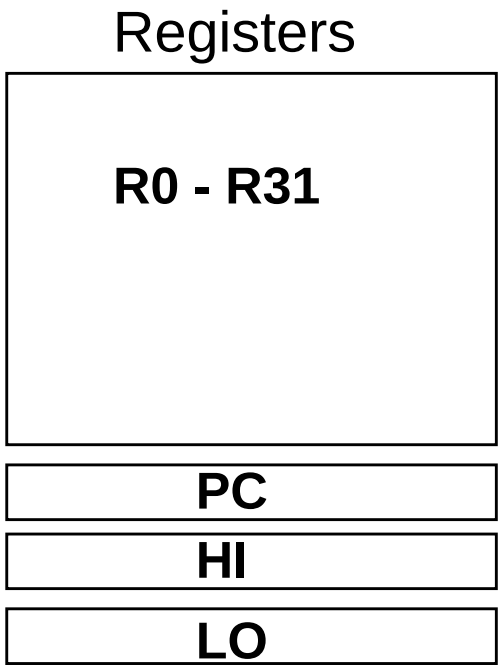
Recovery by flushing instructions on the wrong path (may takes several cycles)



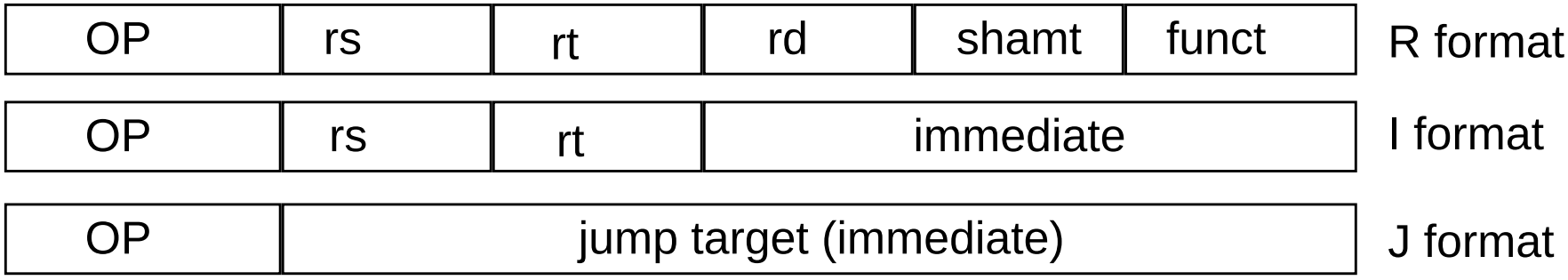
Restart by fetching instructions using the correct PC

MIPS R3000 Instruction Set Architecture (ISA)

- Instruction Categories
  - Computational
  - Load/Store
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

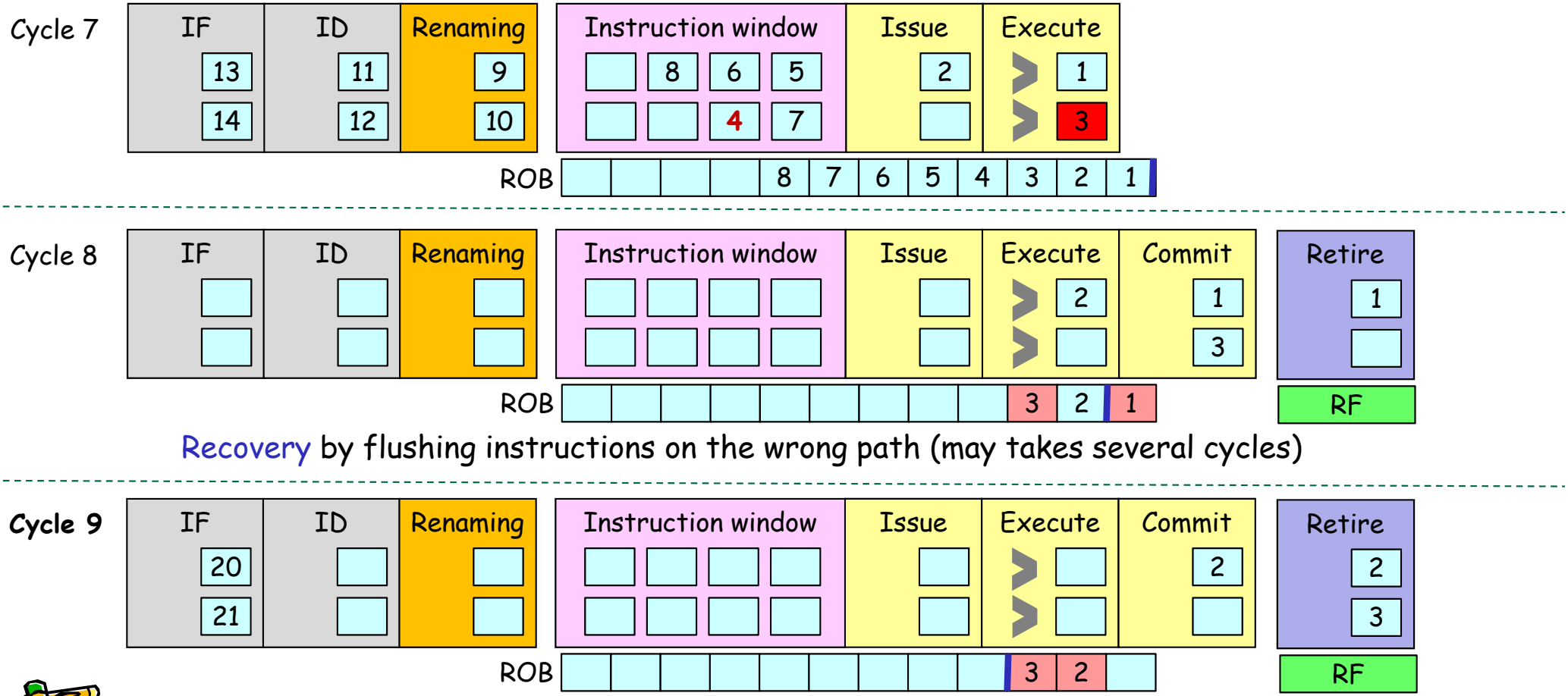


3 Instruction Formats: all 32 bits wide



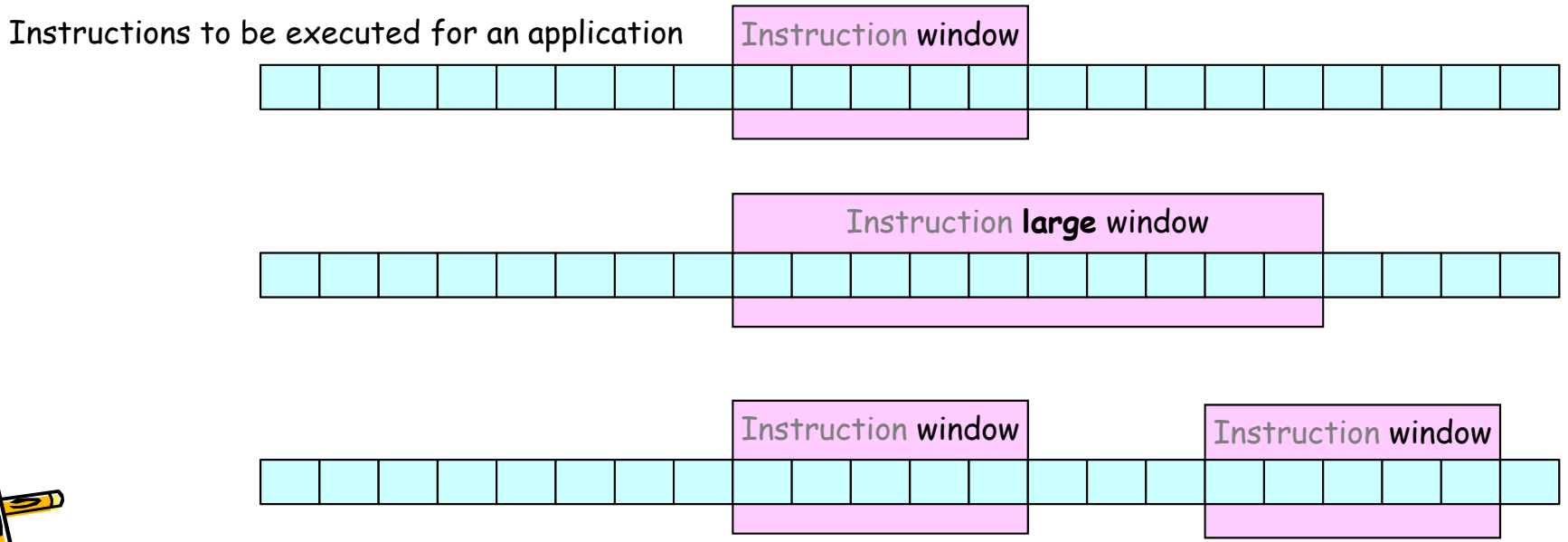
Branch prediction miss and aggressive recovery

- Instruction 3 is a miss predicted branch and its target insn is 20
- Register file (and PC) has the architecture state after insn 3 is executed
- When insn 3 is executed, recover by flushing instructions after insn 3 and restart

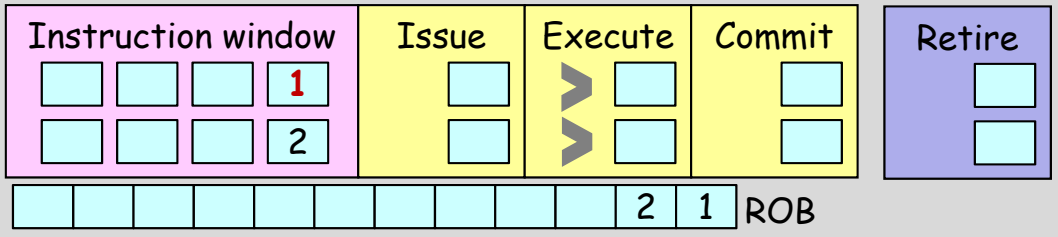


Aside: What is a window?

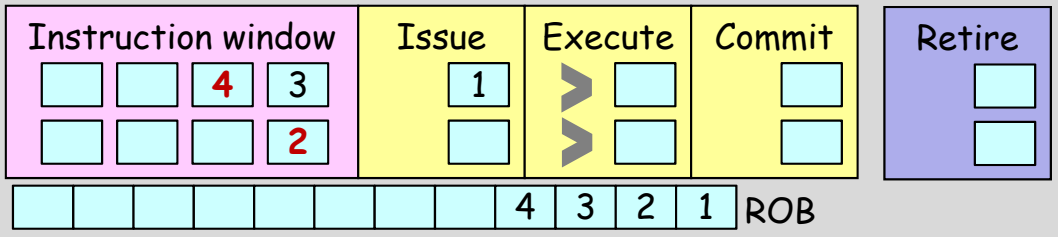
- A window is a space in the wall of a building or in the side of a vehicle, which has glass in it so that light can come in and you can see out. (Collins)



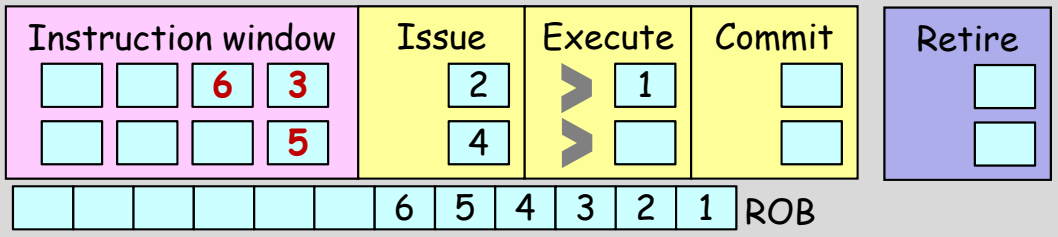
Cycle 1



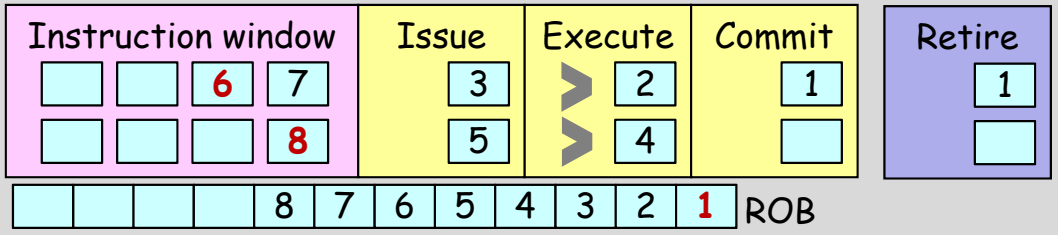
Cycle 2



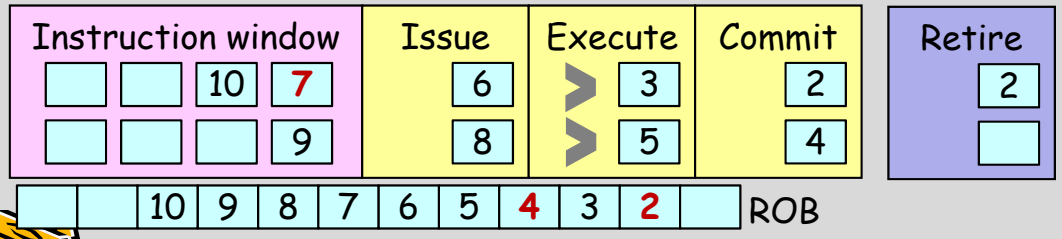
Cycle 3



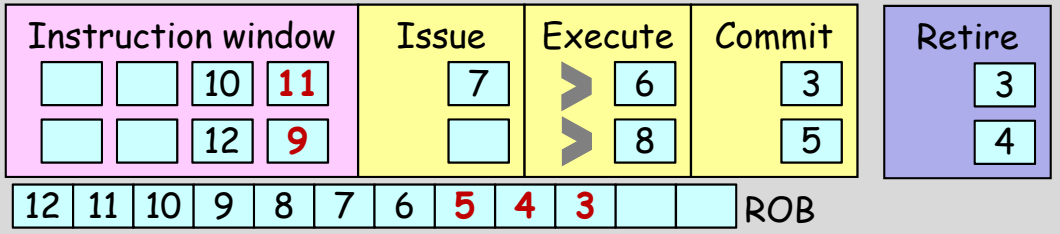
Cycle 4



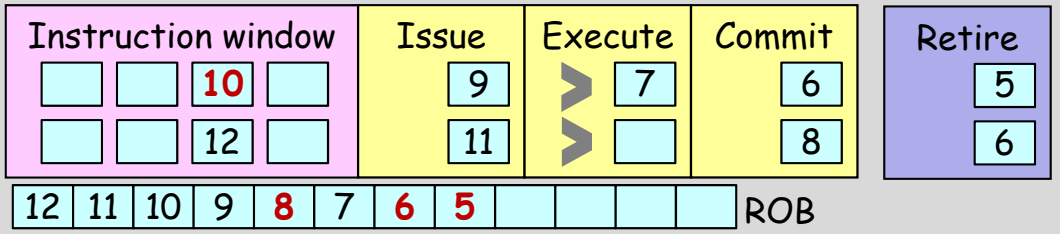
Cycle 5



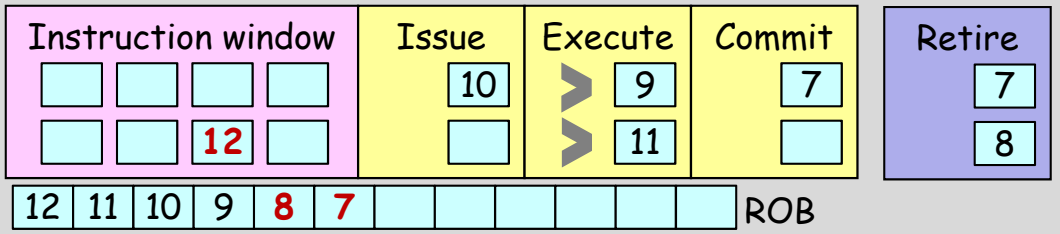
Cycle 6



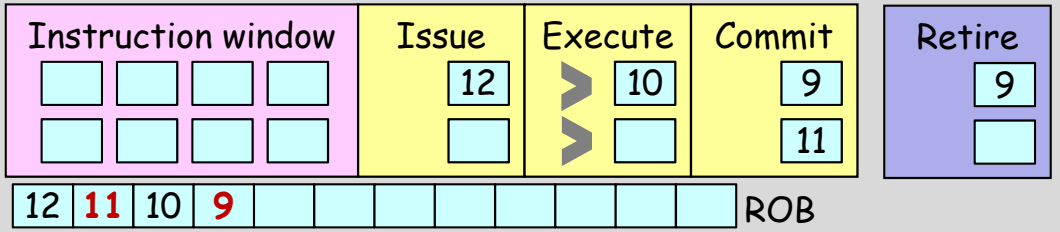
Cycle 7



Cycle 8



Cycle 9



Cycle 10