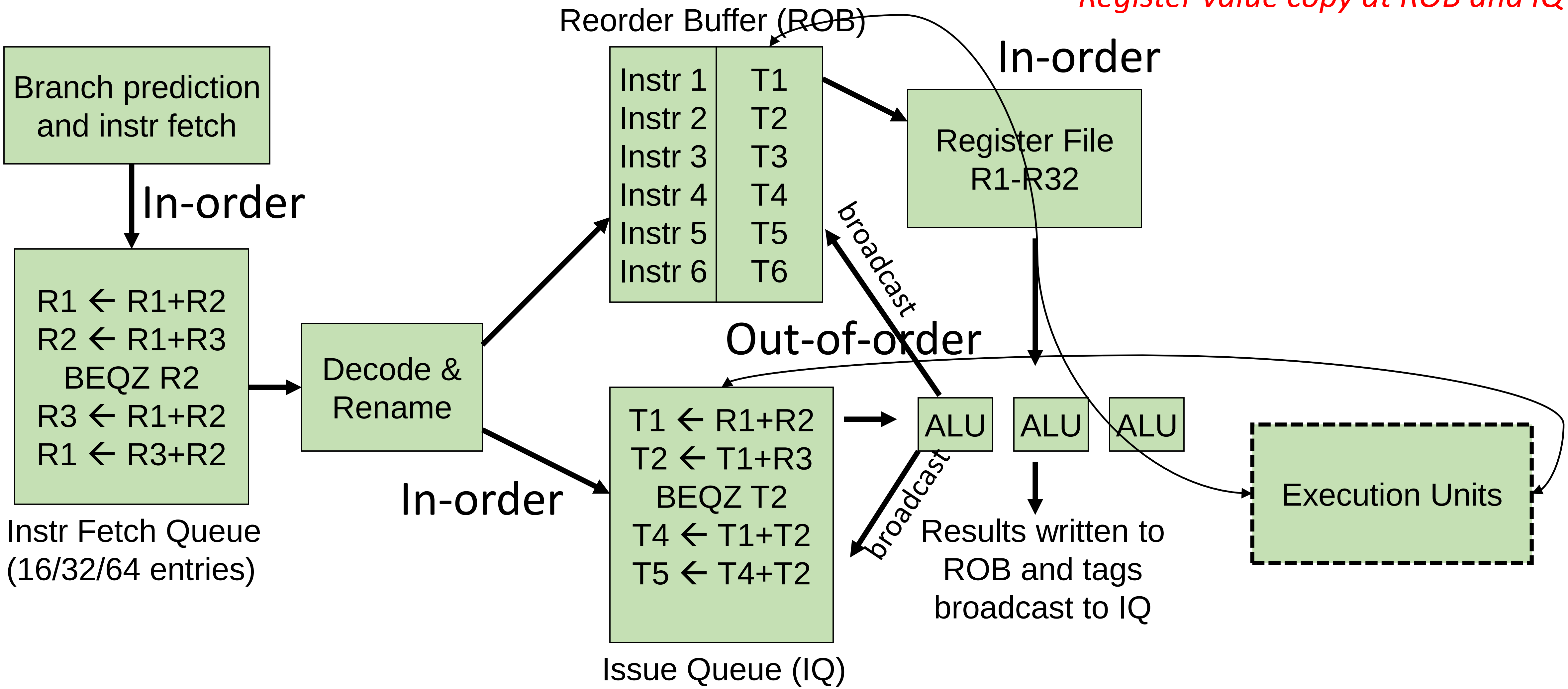
Lecture-12
(Register Renaming+O3 wrap up)
CS422-Spring 2020

Biswa@CSE-IITK

All in One

T1->R1, T5-> R1 when commit

Register value copy at ROB and IQ ☹



Branch prediction and instr fetch

In-order

R1 ← R1+R2
R2 ← R1+R3
BEQZ R2
R3 ← R1+R2
R1 ← R3+R2

Instr Fetch Queue
(16/32/64 entries)

Decode & Rename

In-order

Reorder Buffer (ROB)

Instr 1	T1
Instr 2	T2
Instr 3	T3
Instr 4	T4
Instr 5	T5
Instr 6	T6

In-order

Register File
R1-R32

broadcast

Out-of-order

T1 ← R1+R2
T2 ← T1+R3
BEQZ T2
T4 ← T1+T2
T5 ← T4+T2

Issue Queue (IQ)

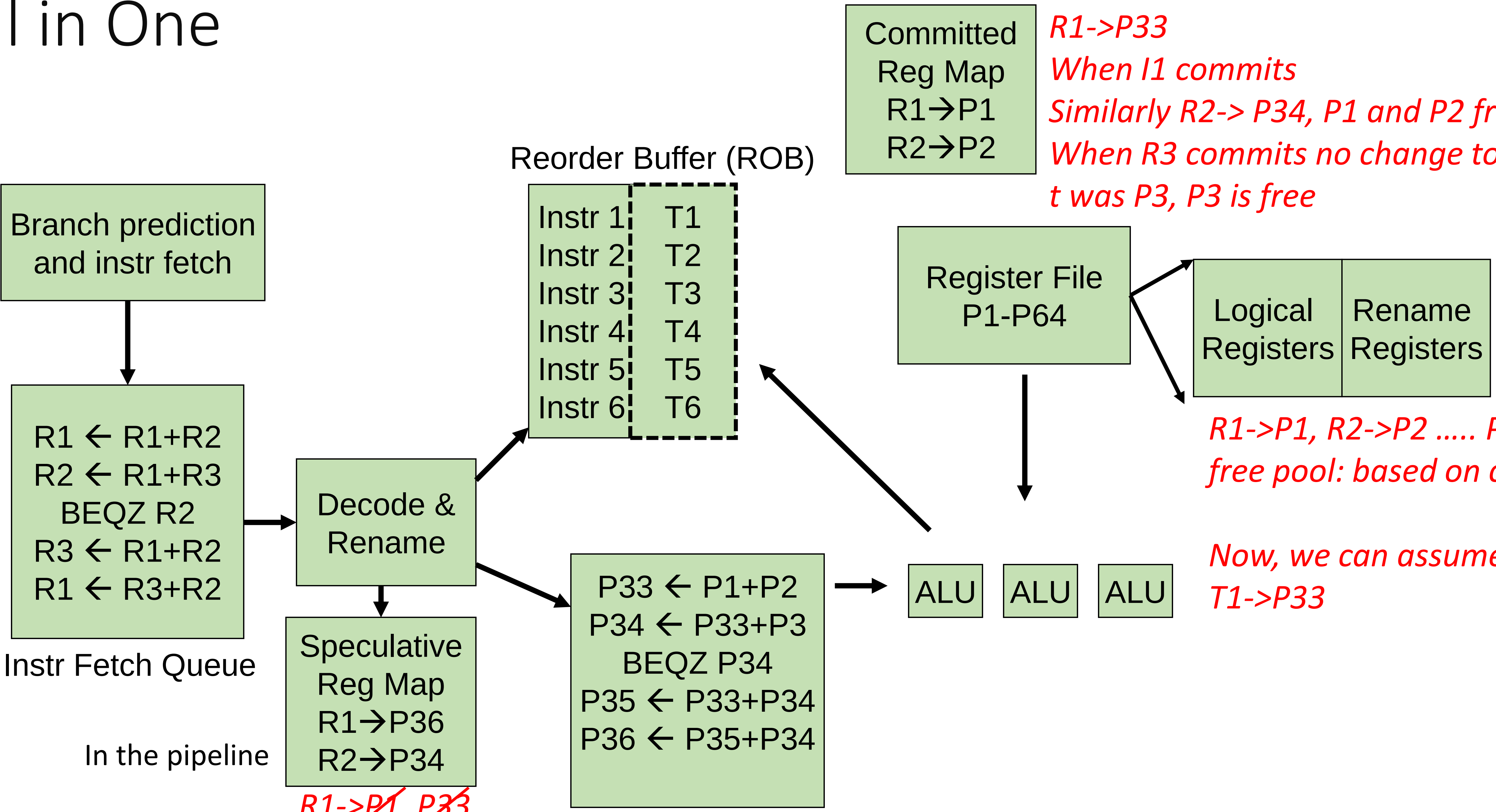
ALU ALU ALU

broadcast

Results written to ROB and tags broadcast to IQ

Execution Units

All in One



Left the pipeline

Committed Reg Map
R1→P1
R2→P2

R1->P33
When I1 commits
Similarly R2-> P34, P1 and P2 free
When R3 commits no change to map, It was P3, P3 is free

Reorder Buffer (ROB)

Instr 1	T1
Instr 2	T2
Instr 3	T3
Instr 4	T4
Instr 5	T5
Instr 6	T6

Branch prediction and instr fetch

Instr Fetch Queue

R1 ← R1+R2
R2 ← R1+R3
BEQZ R2
R3 ← R1+R2
R1 ← R3+R2

Decode & Rename

Speculative Reg Map
R1→P36
R2→P34

In the pipeline

R1->~~P1~~, ~~P33~~

Issue Queue (IQ)

P33 ← P1+P2
P34 ← P33+P3
BEQZ P34
P35 ← P33+P34
P36 ← P35+P34

Register File
P1-P64

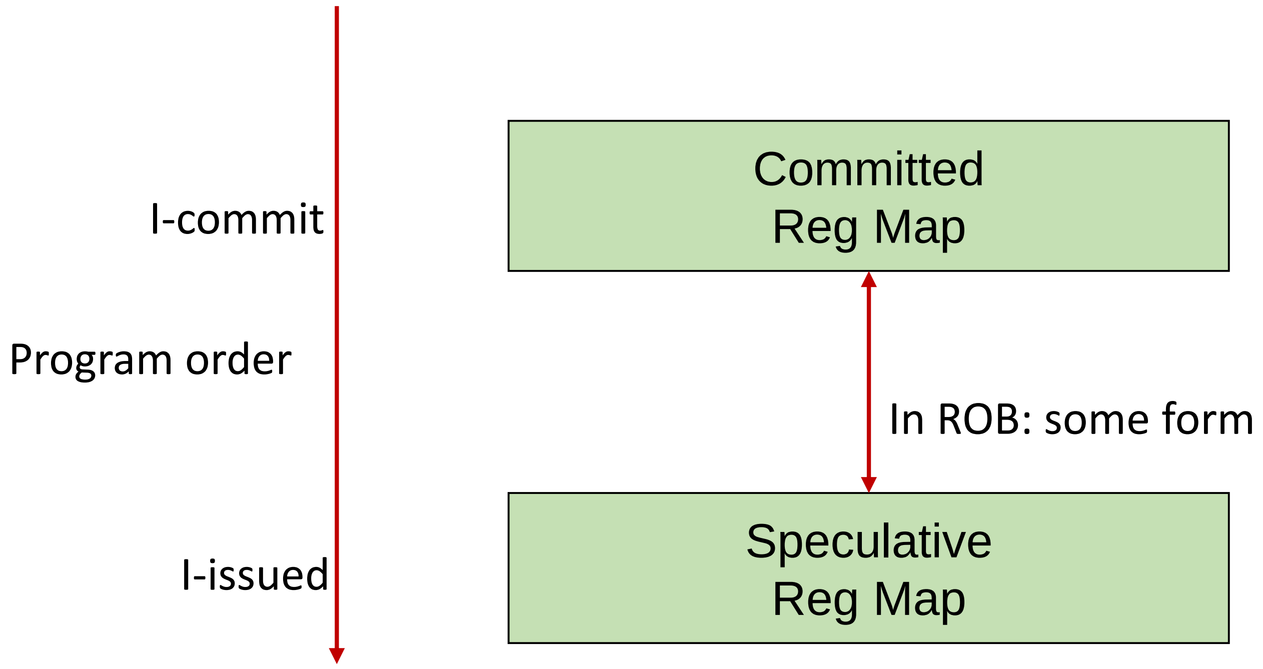
Logical Registers | Rename Registers

R1->P1, R2->P2 ….. P32,
free pool: based on commits

ALU ALU ALU

Now, we can assume
T1->P33

Size of the ROB = physical -logical



I-commit

Program order

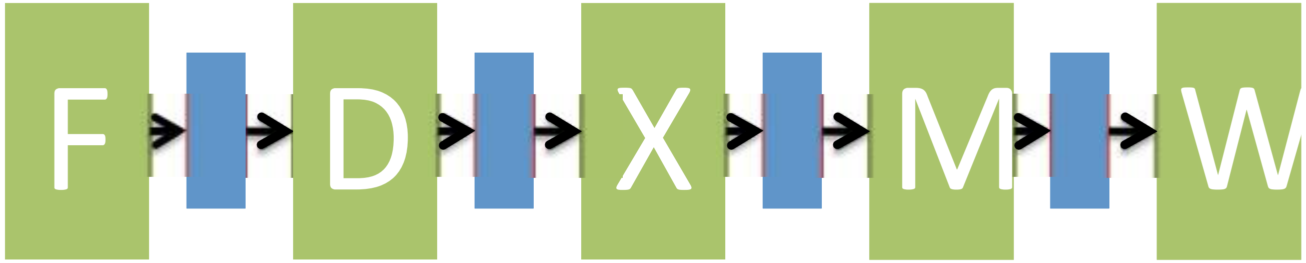
I-issued

Committed Reg Map

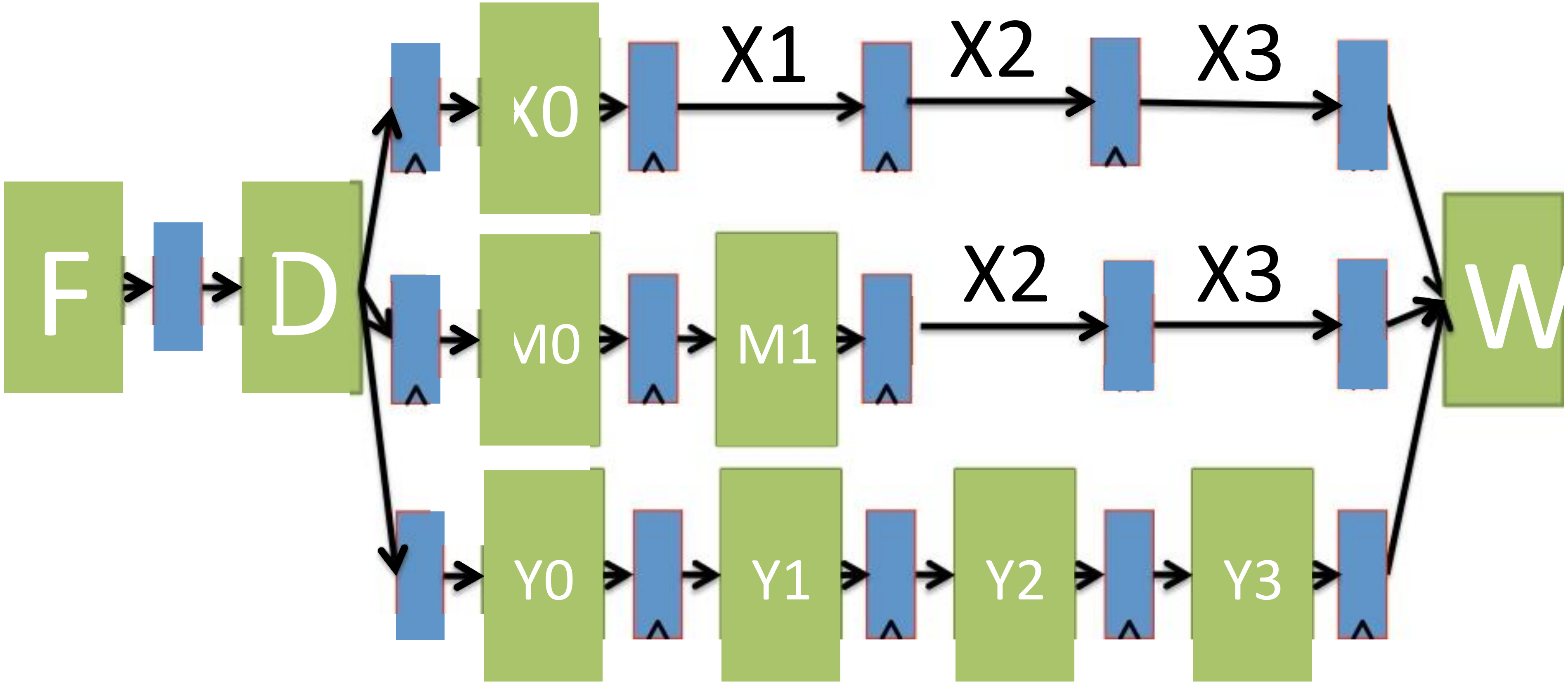
In ROB: some form

Speculative Reg Map

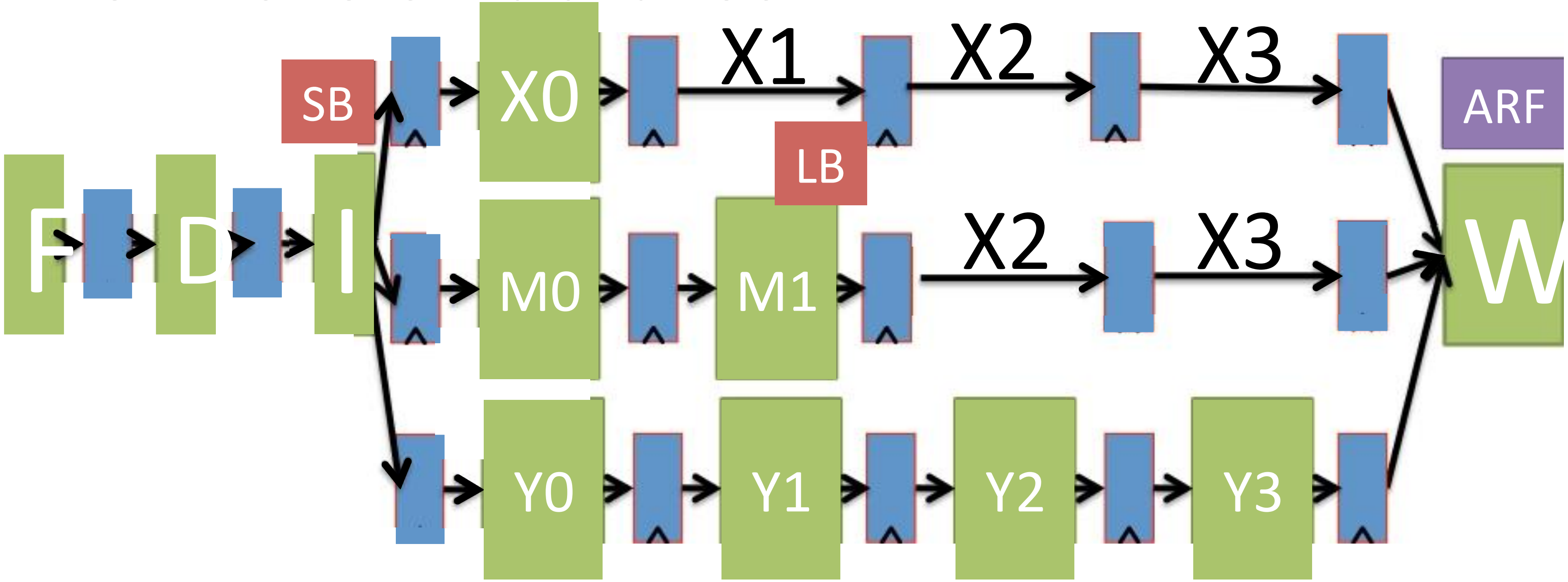
The Complete Picture: From The Beginning (All in-order)



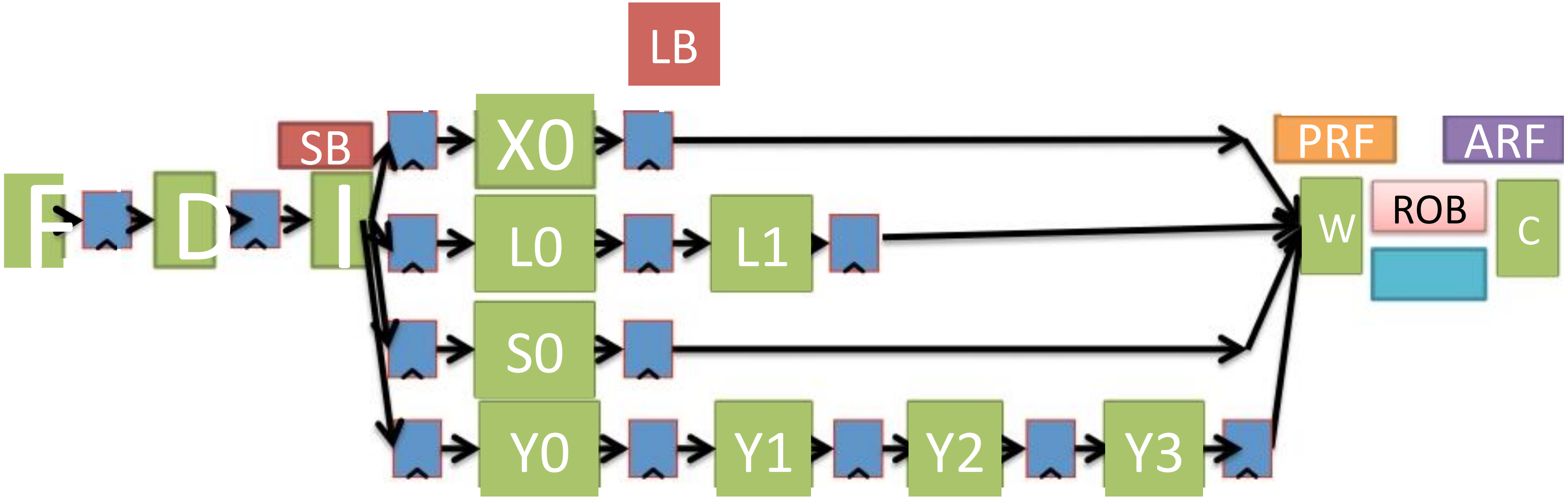
In-order Fetch, Issue, O3 Execution, In-order Write



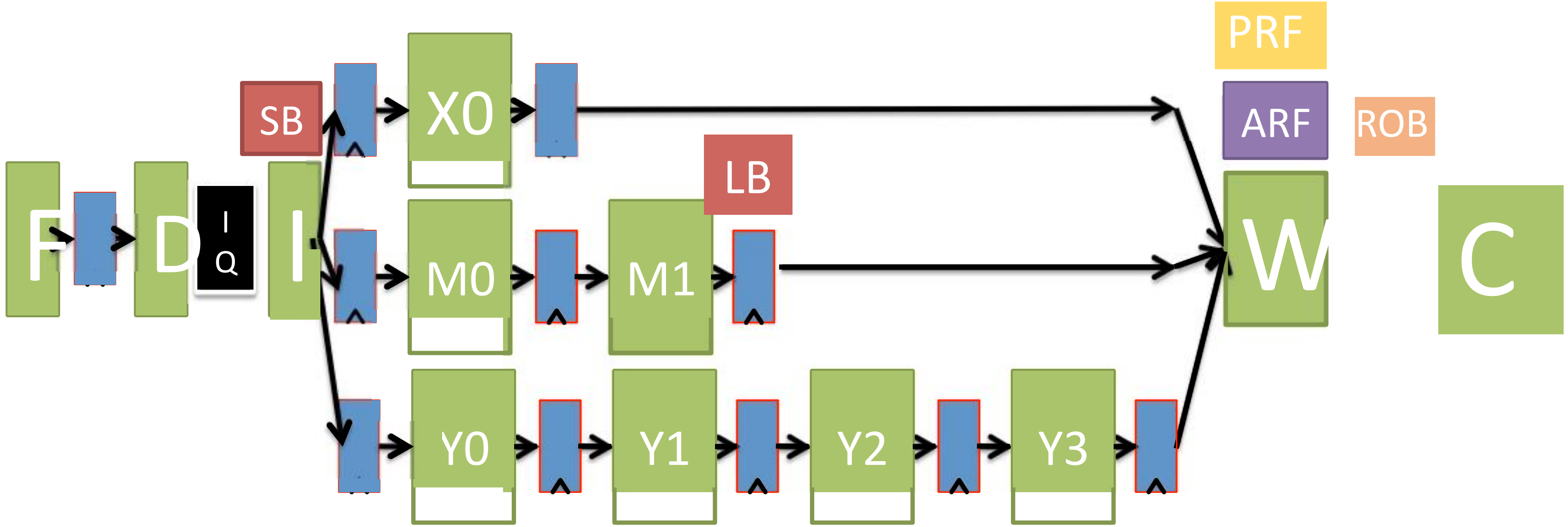
Some More structures



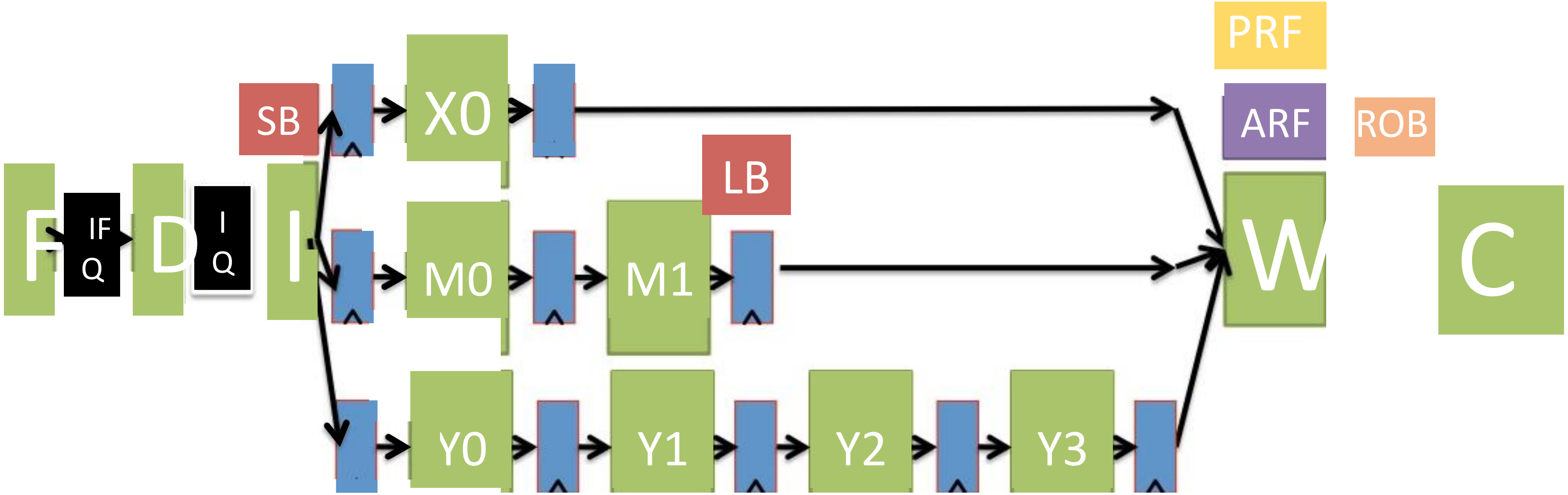
Dig Deep



In-order Frontend, Out-of-order Issue/Writeback/Commit



With SuperScalar

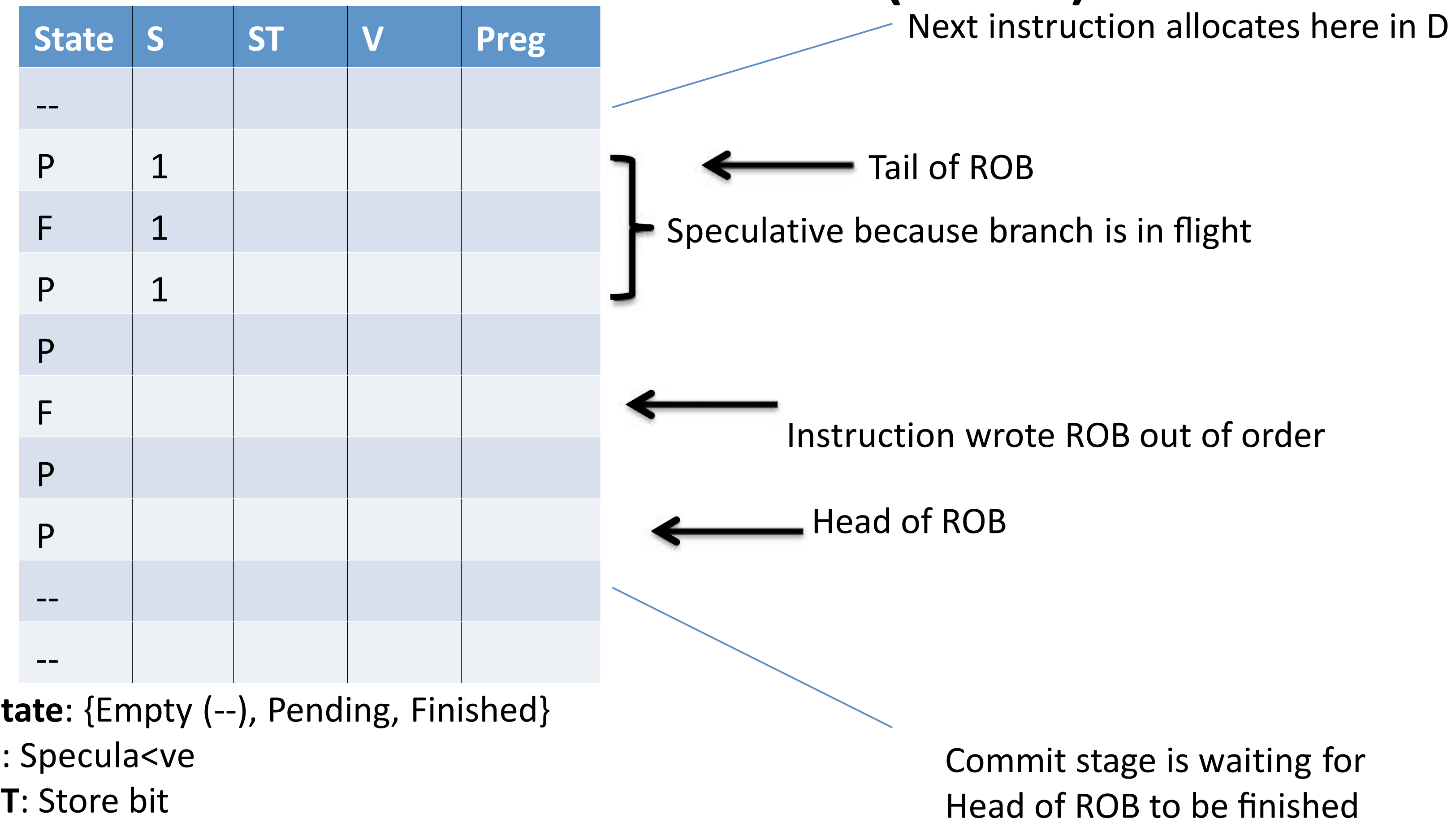


Reorder Buffer (ROB)

State	S	ST	V	Preg
--				
P	1			
F	1			
P	1			
P				
F				
P				
P				
--				
--				

State: {Empty (--), Pending, Finished}
S: Speculative
ST: Store bit
V: Physical Register File Specifier Valid
Preg: Physical Register File Specifier

Reorder Buffer (ROB)



State	S	ST	V	Preg
--				
P	1			
F	1			
P	1			
P				
F				
P				
P				
--				
--				

Next instruction allocates here in D

Tail of ROB

Speculative because branch is in flight

Instruction wrote ROB out of order

Head of ROB

Commit stage is waiting for Head of ROB to be finished

State: {Empty (--), Pending, Finished}

S: Specula<ve

ST: Store bit

V: Physical Register File Specifier Valid

Preg: Physical Register File Specifier

What about LOADs and STOREs

```
st R1, 0(R2)
ld R3, 0(R4)
```

When can we execute the load?

O3 Loads

- Split execution of store instruction into two phases: address calculation and data write

- Can execute load before store, if addresses known and r4 != r2
- Each load address compared with addresses of all previous uncommiped stores

- Don’t execute load if any previous store address not known

Value Prediction

```
st R1, 0(R2)
ld R3, 0(R4)
```

- Guess that r4 != r2
- Execute load before store address known
- Need to hold all completed but uncommitted load/store addresses in program order
- If subsequently find r4==r2, squash load and *all* following instructions

=> Large penalty for inaccurate value prediction