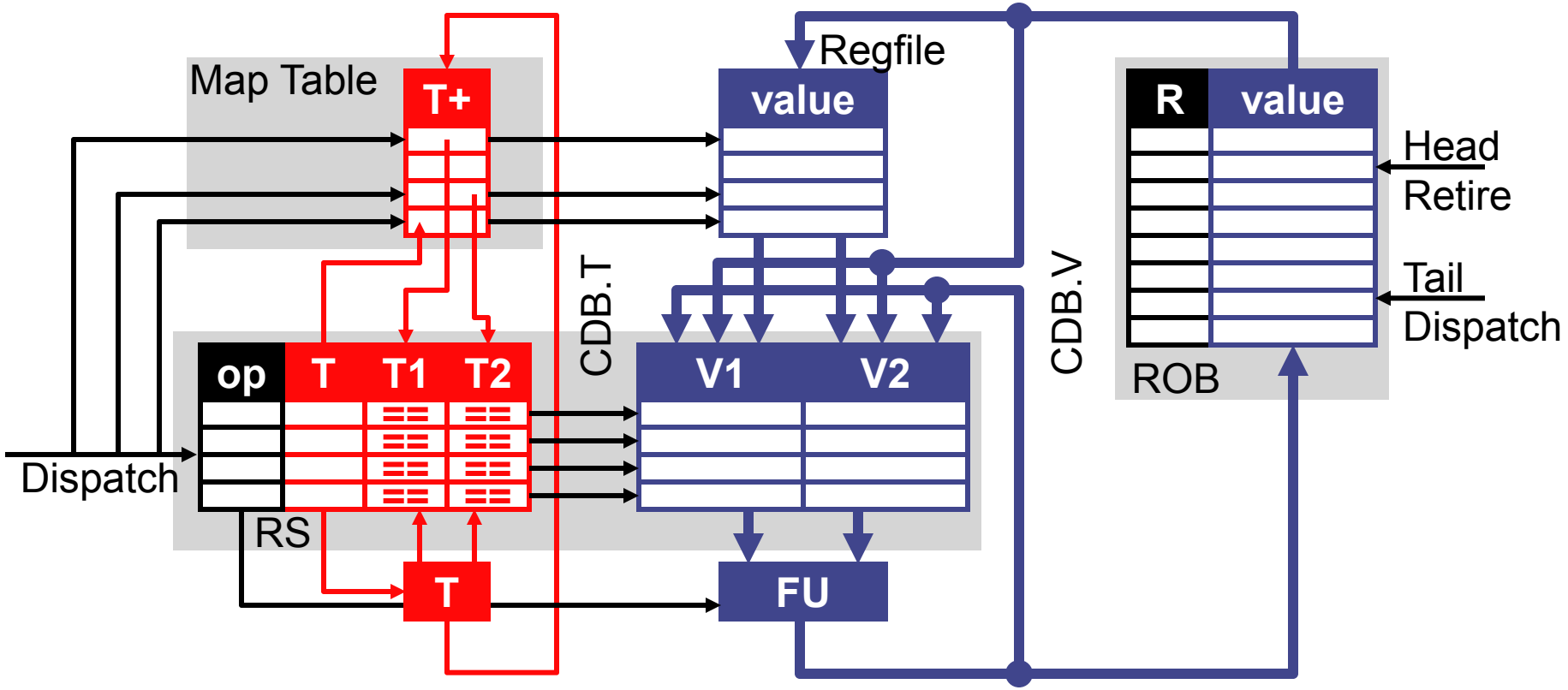
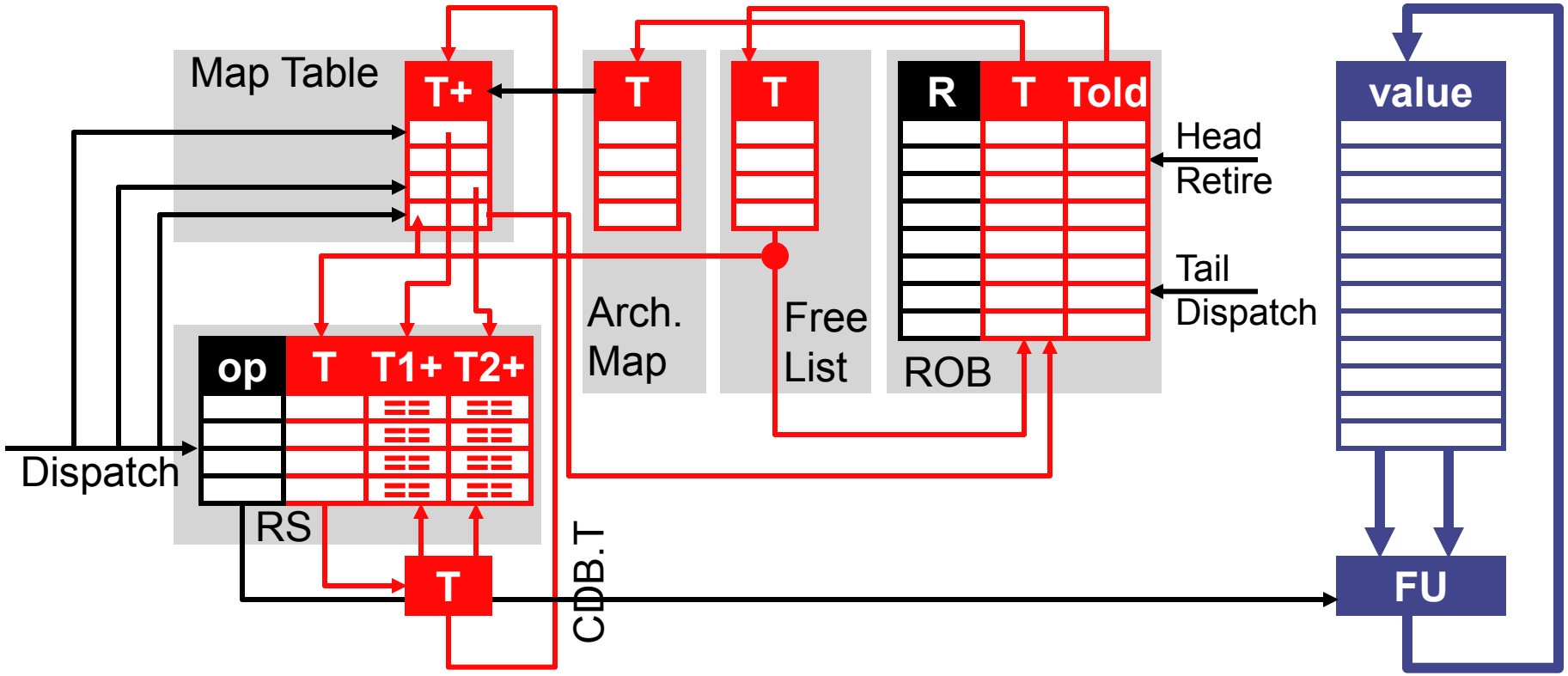
The Problem with P6



Problem for high performance implementations

- Too much **value movement** (regfile/ROB→RS→ROB→regfile)
- Multi-input muxes, long buses complicate routing and slow clock

MIPS R10K: Alternative Implementation



- One big **physical register file** holds all data - no copies
  - \+ Register file close to FUs → small fast data path
  - ROB and RS "on the side" used only for control and tags

Register Renaming in R10K

Architectural register file? Gone

Physical register file holds all values

- #physical registers = #architectural registers + #ROB entries
- Map architectural registers to physical registers
- Removes WAW, WAR hazards (physical registers replace RS copies)

Fundamental change to **map table**

- Mappings cannot be 0 (there is no architectural register file)

Free list keeps track of unallocated physical registers

- ROB is responsible for returning physical registers to free list

Conceptually, this is "true register renaming"

- Have already seen an example

Register Renaming Example

Parameters

- Names: **r1,r2,r3**
- Locations: **p1,p2,p3,p4,p5,p6,p7**
- Original mapping: **r1**→p1, **r2**→p2, **r3**→p3, p**4**–p**7** are "free"

MapTable

r1	r2	r3
p1	p2	p3
p4	p2	p3
p4	p2	p5
p6	p2	p5

FreeList

p4,p5,p6,p7
p5,p6,p7
p6,p7
p7

Raw insns	Renamed insns
add r2,r3,r1	add p2,p3,p4
sub r2,r1,r3	sub p2,p4,p5
mul r2,r3,r1	mul p2,p5,p6
div r1,r3,r2	div p6,p5,p7

Question: how is the insn after div renamed?

- We are out of free locations (physical registers)
- Real question: how/when are physical registers freed?

Freeing Registers in P6 and R10K

P6

- No need to free storage for speculative ("in-flight") values explicitly
- Temporary storage comes with ROB entry
- R: copy speculative value from ROB to register file, free ROB entry

R10K

- Can't free physical register when insn retires
- No architectural register to copy value to
- But…
- Can free physical register previously mapped to same logical register
- Why? All insns that will ever read its value have retired

Freeing Registers in R10K

MapTable

r1	r2	r3
p1	p2	p3
p4	p2	p3
p4	p2	p5
p6	p2	p5

FreeList

p4,p5,p6,p7
p5,p6,p7
p6,p7
p7

Raw insns

```
add r2,r3,r1
sub r2,r1,r3
mul r2,r3,r1
div r1,r3,r2
```

Renamed insns

```
add p2,p3,p4
sub p2,p4,p5
mul p2,p5,p6
div p6,p5,p7
```

- When **`add`** retires, free p1
- When **`sub`** retires, free p3
- When **`mul`** retires, free ?
- When **`div`** retires, free ?
- See the pattern?

R10K Data Structures

New tags (again)

- P6: ROB# → R10K: PR#

ROB

- **T**: physical register corresponding to insn's logical output
- **Told**: physical register previously mapped to insn's logical output

RS

- **T**, **T1**, **T2**: output, input physical registers

Map Table

- **T+**: PR# (never empty) + "ready" bit

Architectural Map Table

- **T**: PR# (never empty)

Free List

- **T**: PR#

No values in ROB, RS, or on CDB

R10K Data Structures

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1					
	2	mulf f0,f1,f2					
	3	stf f2,Z(r1)					
	4	addi r1,4,r1					
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#2+
f2	PR#3+
r1	PR#4+

Arch. Map

Reg	T+
f0	PR#1
f1	PR#2
f2	PR#3
r1	PR#4

Free List

PR#5,PR#6,
PR#7,PR#8

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	no				
4	FP1	no				
5	FP2	no				

Notice I: no values anywhere

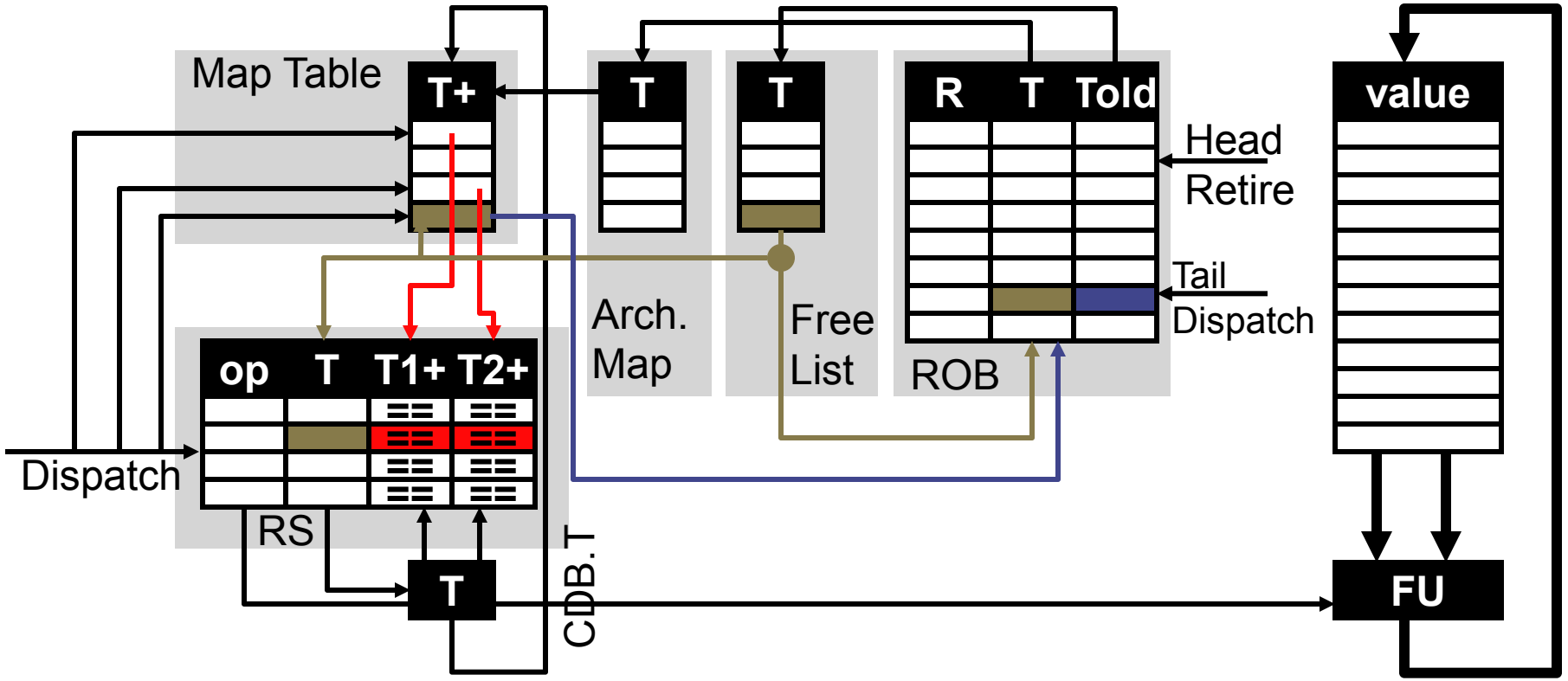
Notice II: MapTable is never empty

R10K Pipeline

R10K pipeline structure: F, **D**, S, X, **C**, **R**

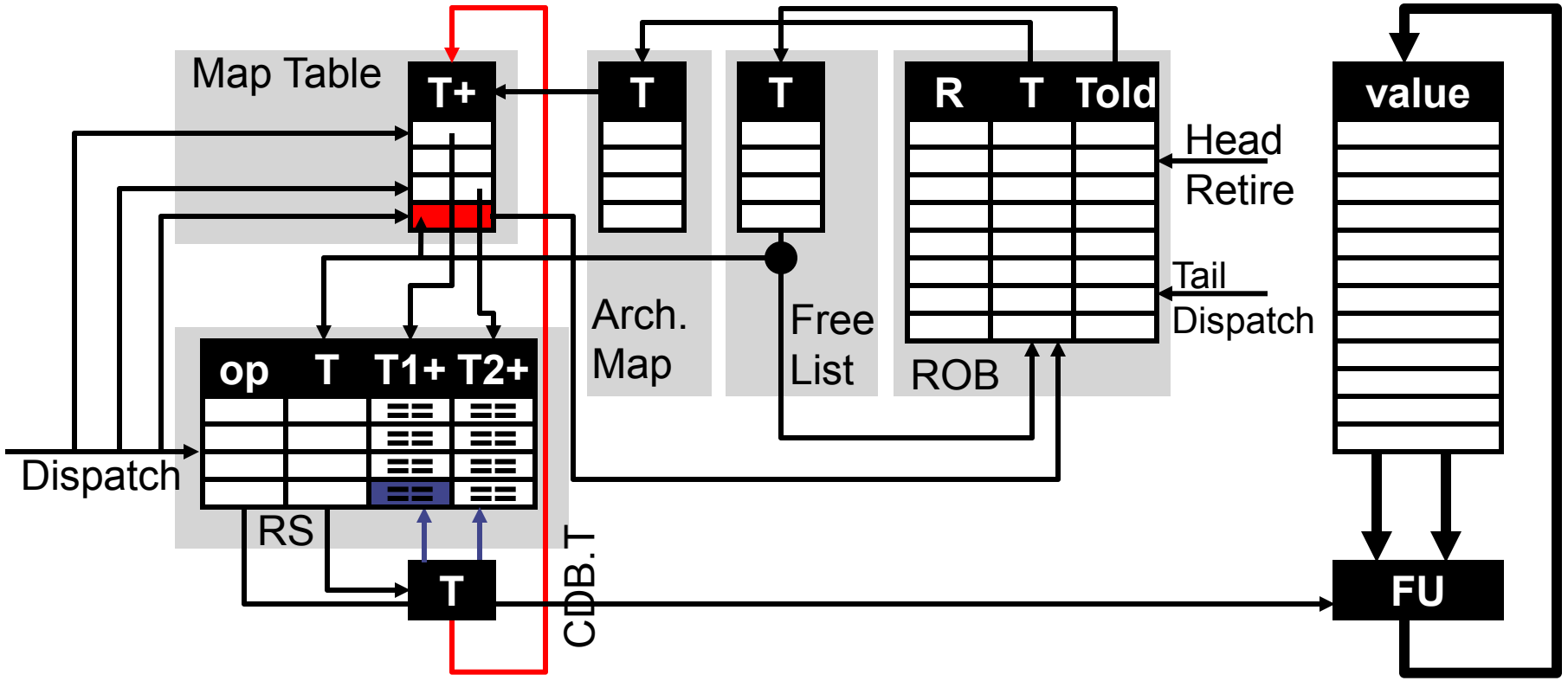
- **D (dispatch)**
  - Structural hazard (RS, ROB, LSQ, **physical registers**) ? stall
  - Allocate RS, ROB, LSQ entries and new physical register (T)
  - **Record previously mapped physical register (Told)**
- **C (complete)**
  - Write destination physical register
- **R (retire)**
  - ROB head not complete ? Stall
  - Handle any exceptions
  - Store write LSQ head to D$
  - Free ROB, LSQ entries
  - **Free previous physical register (Told)**
  - **Record committed physical register (T)**

R10K Dispatch (D)



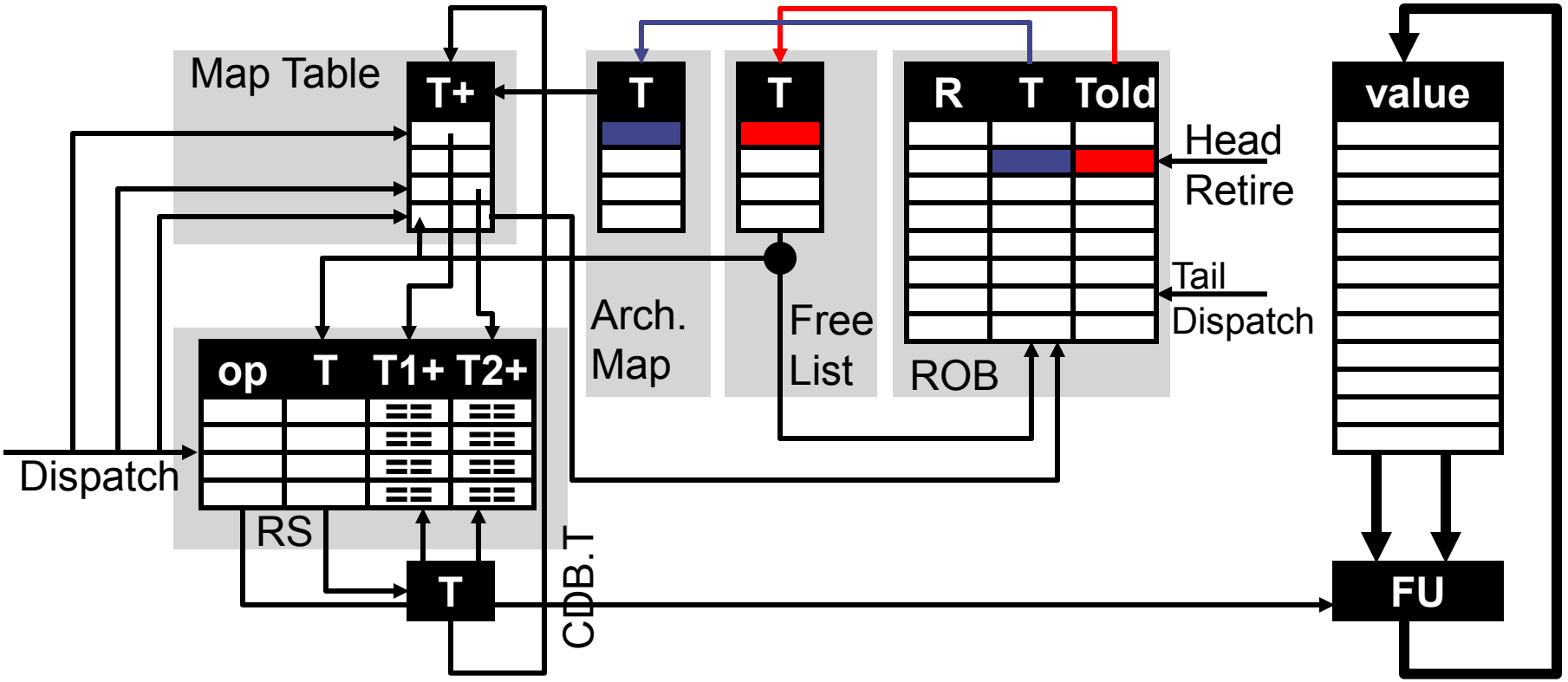
- Read preg (physical register) tags for input registers, store in RS
- Read preg tag for output register, store in ROB (Told)
- Allocate new preg (free list) for output register, store in RS, ROB, Map Table

R10K Complete (C)



- Set insn's output register ready bit in map table
- Set ready bits for matching input tags in RS

R10K Retire (R)



- Return Told of ROB head to free list
- Record T of ROB head in architectural map table

R10K: Cycle 1

ROB

ht	#	Insn	T	Told	S	X	C
ht	1	ldf X(r1),f1	PR#5	PR#2			
	2	mulf f0,f1,f2					
	3	stf f2,Z(r1)					
	4	addi r1,4,r1					
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#3+
r1	PR#4+

Arch. Map

Reg	T+
f0	PR#1
f1	PR#2
f2	PR#3
r1	PR#4

Free List

PR#5, PR#6, PR#7, PR#8

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	yes	ldf	PR#5		PR#4+
3	ST	no				
4	FP1	no				
5	FP2	no				

Allocate new preg (PR#5) to f1

Remember old preg mapped to f1 (PR#2) in ROB

R10K: Cycle 2

ROB

ht	#	Insn	T	Told	S	X	C
h	1	ldf X(r1),f1	PR#5	PR#2	c2		
t	2	mulf f0,f1,f2	PR#6	PR#3			
	3	stf f2,Z(r1)					
	4	addi r1,4,r1					
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#6
r1	PR#4+

Arch. Map

Reg	T+
f0	PR#1
f1	PR#2
f2	PR#3
r1	PR#4

Free List

PR#6, PR#7, PR#8

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	yes	ldf	PR#5		PR#4+
3	ST	no				
4	FP1	yes	mulf	PR#6	PR#1+	PR#5
5	FP2	no				

Allocate new preg (PR#6) to f2

Remember old preg mapped to f3 (PR#3) in ROB

R10K: Cycle 3

ROB

ht	#	Insn	T	Told	S	X	C
h	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	
	2	mulf f0,f1,f2	PR#6	PR#3			
t	3	stf f2,Z(r1)					
	4	addi r1,4,r1					
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#6
r1	PR#4+

Arch. Map

Reg	T+
f0	PR#1
f1	PR#2
f2	PR#3
r1	PR#4

Free List

PR#7,PR#8

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	yes	mulf	PR#6	PR#1+	PR#5
5	FP2	no				

Stores are not allocated pregs

Free

R10K: Cycle 4

ROB

ht	#	Insn	T	Told	S	X	C
h	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
	2	mulf f0,f1,f2	PR#6	PR#3	c4		
	3	stf f2,Z(r1)					
t	4	addi r1,4,r1	PR#7	PR#4			
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#7

Arch. Map

Reg	T+
f0	PR#1
f1	PR#2
f2	PR#3
r1	PR#4

Free List

PR#7, PR#8

CDB

T
PR#5

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	yes	mulf	PR#6	PR#1+	PR#5+
5	FP2	no				

ldf completes
set MapTable ready bit

Match PR#5 tag from CDB & issue

R10K: Cycle 5

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
h	2	mulf f0,f1,f2	PR#6	PR#3	c4	c5	
	3	stf f2,Z(r1)					
	4	addi r1,4,r1	PR#7	PR#4	c5		
t	5	ldf X(r1),f1	PR#8	PR#5			
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#8
f2	PR#6
r1	PR#7

Arch. Map

Reg	T+
f0	PR#1
f1	PR#5
f2	PR#3
r1	PR#4

Free List

PR#8, PR#2

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	yes	ldf	PR#8		PR#7
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

Free

ldf retires
Return PR#2 to free list
Record PR#5 in Arch map

Precise State in R10K

- Problem with R10K design? Precise state is more difficult
  - Physical registers are written out-of-order (at C)
  - That's OK, there is no architectural register file
  - We can "free" written registers and "restore" old ones
  - Do this by manipulating the Map Table, Arch Table, Free List, not regfile

- Two ways of restoring Map Table and Free List
  - Option I: serial rollback using T, T_{old} ROB fields
    ± Slow, but simple
  - Option II: single-cycle restoration from some checkpoint
    ± Fast, but checkpoints are expensive
  - Modern processor compromise: **make common case fast**
    - Checkpoint only (low-confidence) branches (frequent rollbacks)
    - Serial recovery for page-faults and interrupts (rare rollbacks)

R10K: Cycle 5 (with precise state)

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
h	2	mulf f0,f1,f2	PR#6	PR#3	c4	c5	
	3	stf f2,Z(r1)					
	4	addi r1,4,r1	PR#7	PR#4	c5		
t	5	ldf X(r1),f1	PR#8	PR#5			
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#8
f2	PR#6
r1	PR#7

CDB

T

Free List

PR#8, PR#2

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	yes	ldf	PR#8		PR#7
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

undo insns 3-5
(doesn't matter why)
use serial rollback

R10K: Cycle 6 (with precise state)

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
h	2	mulf f0,f1,f2	PR#6	PR#3	c4	c5	
	3	stf f2,Z(r1)					
t	4	addi r1,4,r1	PR#7	PR#4	c5		
	5	ldf X(r1),f1	~~PR#8~~	~~PR#5~~			
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#7

Free List

PR#2, PR#8

CDB

T

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

undo ldf (ROB#5)
1. free RS
2. free T (PR#8), return to FreeList
3. restore MT[f1] to Told (PR#5)
4. free ROB#5

insns may execute during rollback (not shown)

R10K: Cycle 7 (with precise state)

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
h	2	mulf f0,f1,f2	PR#6	PR#3	c4	c5	
t	3	stf f2,Z(r1)					
	4	addi r1,4,r1	PR#7	PR#4	c5		
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#4+

CDB

T

Free List

PR#2, PR#8, PR#7

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

undo addi (ROB#4)
1. free RS
2. free T (PR#7), return to FreeList
3. restore MT[r1] to Told (PR#4)
4. free ROB#4

R10K: Cycle 8 (with precise state)

ROB

ht	#	Insn	T	Told	S	X	C
	1	ldf X(r1),f1	PR#5	PR#2	c2	c3	c4
ht	2	mulf f0,f1,f2	PR#6	PR#3	c4	c5	
	3	stf f2,Z(r1)					
	4	addi r1,4,r1					
	5	ldf X(r1),f1					
	6	mulf f0,f1,f2					
	7	stf f2,Z(r1)					

Map Table

Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#4+

CDB

T

Free List

PR#2, PR#8, PR#7

Reservation Stations

#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	no				
4	FP1	no				
5	FP2	no				

undo stf (ROB#3)
1. free RS
2. free ROB#3
3. no registers to restore/free
4. how is D$ write undone?

P6 vs. R10K (Renaming)

Feature	P6	R10K
Value storage	ARF,ROB,RS	PRF
Register read	@D: ARF/ROB → RS	@S: PRF → FU
Register write	@R: ROB → ARF	@C: FU → PRF
Speculative value free	@R: automatic (ROB)	@R: overwriting insn
Data paths	ARF/ROB → RS RS → FU FU → ROB ROB → ARF	PRF → FU FU → PRF
Precise state	Simple: clear everything	Complex: serial/checkpoint

- R10K-style became popular in late 90's, early 00's
  - E.g., MIPS R10K (duh), DEC Alpha 21264, Intel Pentium4
- P6-style is perhaps making a comeback
  - Why? Frequency (power) is on the retreat, simplicity is important

Summary

- Modern dynamic scheduling must support precise state
  - A software sanity issue, not a performance issue
- Strategy: Writeback → Complete (OoO) + Retire (iO)

- Two basic designs
  - P6: Tomasulo + re-order buffer, copy based register renaming
    - ± Precise state is simple, but fast implementations are difficult
  - R10K: implements true register renaming
    - ± Easier fast implementations, but precise state is more complex

Dynamic Scheduling Summary

- Out-of-order execution: a performance technique
  - Easier/more effective in hardware than software (isn't everything?)
  - Idea: make scheduling transparent to software
- Feature I: Dynamic scheduling (iO $\rightarrow$ OoO)
  - "Performance" piece: re-arrange insns into high-performance order
  - Decode (iO) $\rightarrow$ dispatch (iO) + issue (OoO)
  - Two algorithms: Scoreboard, Tomasulo
- Feature II: Precise state (OoO $\rightarrow$ iO)
  - "Correctness" piece: put insns back into program order
  - Writeback (OoO) $\rightarrow$ complete (OoO) + retire (iO)
  - Two designs: P6, R10K
- Next: memory scheduling