EECS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

Midterm Exam Review Session

October 6, 2025

Slides from Mustafa Miyaziwala, W25

Midterm Exam Information

- Wednesday (10/8), 6-8 PM
- Coverage:
  - Lecture 1-10f (up to and including R10K)
  - All project/lab content
- What to bring:
  - Calculator (Please bring one!)
  - One-sided 8.5x11 note sheet

Exam Prep Tips

- Take past exams
  - 6 exams posted on website
- Review homeworks
- Review lecture slides
- Ask questions in office hours/Piazza
- **Take care of yourself**

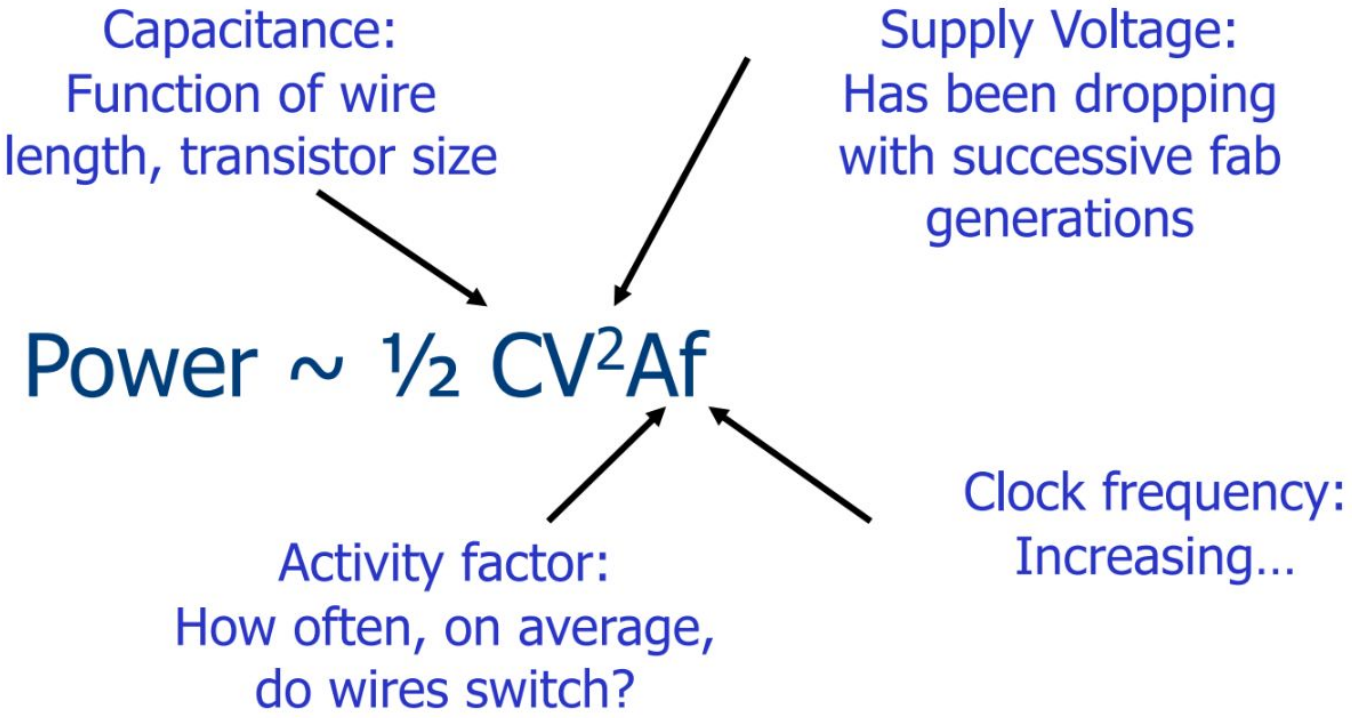
Review Session Agenda

This review session will cover:

1. Power and performance
2. Out-of-order Architectures
    a. Scoreboarding
    b. Tomasulo's
    c. P6
    d. R10K

Power and Performance

Power and DVFS



Performance Key Points

Amdahl's law

$$S_{overall} = 1 / ((1-f) + f/S)$$

Iron law

$$\frac{Time}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Time}{Cycle}$$

Averaging Techniques

Arithmetic Time

$$\frac{1}{n}\sum_{i=1}^{n} Time_i$$

Harmonic Rates

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{Rate_i}}$$

Geometric Ratios

$$\sqrt[n]{\prod_{i=1}^{n} Ratio_i}$$

Out-of-order Architectures

What is ILP?

What is the ILP of the following program (assume each instruction takes one unit of time to execute):

R1 = R1 + R2
R3 = R3 + R4
R4 = R1 + R2

True and False Dependencies

RAW Hazard (true dependency):

- Later instruction reads the register written to by an earlier one

WAW Hazard (false dependency):

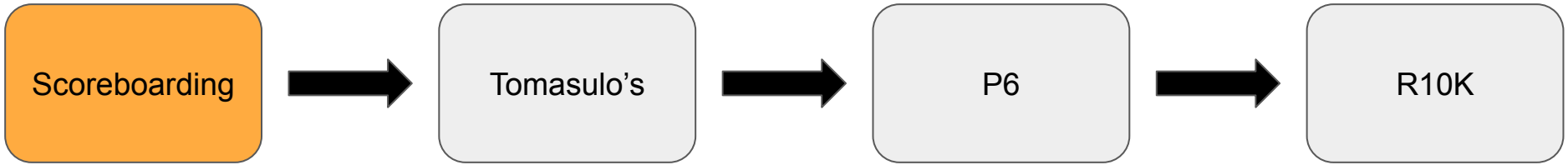
- Earlier instruction overrides a register already written to by a later instruction

WAR Hazard (false dependency):

- Later instruction writes to a register read by an earlier instruction

```
R1 = R3
R2 = R1 + R1
R1 = R4
```

Out-of-order Roadmap



Scoreboarding Structures

- Architectural register file
- Functional Unit Status Table (FUST)
  - Which FUs are busy and the status of arguments
- Register Status Table
  - Which FUs will write to a certain register
- Instruction Status Table
  - Tracks an instruction's progress through stages

Scoreboarding Structures

Insn Status				
Insn	D	S	X	W
`ldf X(r1),f1`	c1	c2	c3	
`mulf f0,f1,f2`	c2			
`stf f2,Z(r1)`	c3			
`addi r1,4,r1`				
`ldf X(r1),f1`				
`mulf f0,f1,f2`				
`stf f2,Z(r1)`				

Reg Status	
Reg	T
f0	
f1	LD
f2	FP1
r1	

Functional unit status							
FU	busy	op	R	R1	R2	T1	T2
ALU	no						
LD	yes	ldf	f1	-	r1	-	-
ST	yes	stf	-	f2	r1	FP1	-
FP1	yes	mulf	f2	f0	f1	-	LD
FP2	no						

allocate

Scoreboarding Stages

Fetch:
- Same as always

Dispatch:
- Structural or WAW hazard ? stall : allocate FUST, RST, and IST

Issue:
- RAW hazard ? wait : read registers and go to execute

Execute:
- Perform operation, notify instruction status table when done

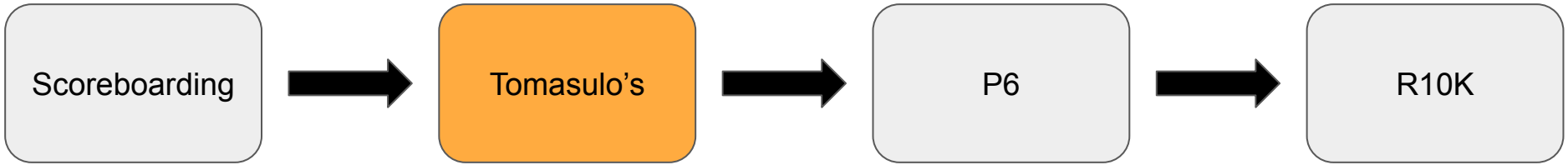
Writeback:
- WAR hazard ? wait : write to register file, clear tags of waiting instructions, free FUST, RST, and IST

Some Scoreboarding Questions

Why must dispatch stall for WAW hazards? What could happen if we didn't stall dispatch?

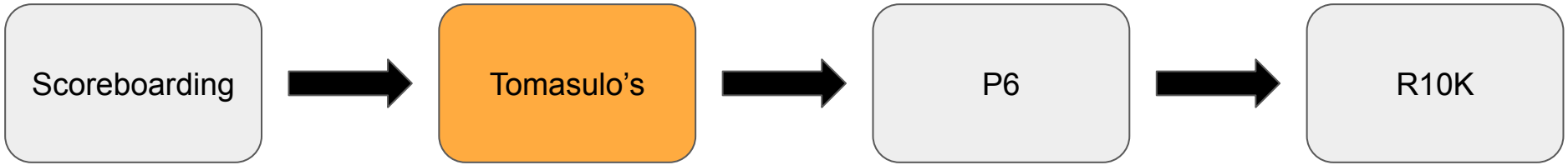
Why must writeback stall for WAR hazards? Why can't it immediately write to the register file once execute finishes?

Out-of-order Roadmap



Out-of-order Roadmap

Copy-based register renaming →
No WAR/WAW stalls



Tomasulo's Structures

Reservation Station (RS): Mostly same as FUST, except:

- Tags are other RS indices rather that FUs
- Values of source operands are copied into RS when ready

Map Table: Maps architectural registers to the RS entry that will produces its value

Common Data Bus (CDB): Broadcasts <RS#, Value> of instructions completing execution

Tomasulo's Structures

Insn Status				
Insn	D	S	X	W
ldf X(r1),f1	c1	c2	c3	
mulf f0,f1,f2	c2			
stf f2,Z(r1)	c3			
addi r1,4,r1				
ldf X(r1),f1				
mulf f0,f1,f2				
stf f2,Z(r1)				

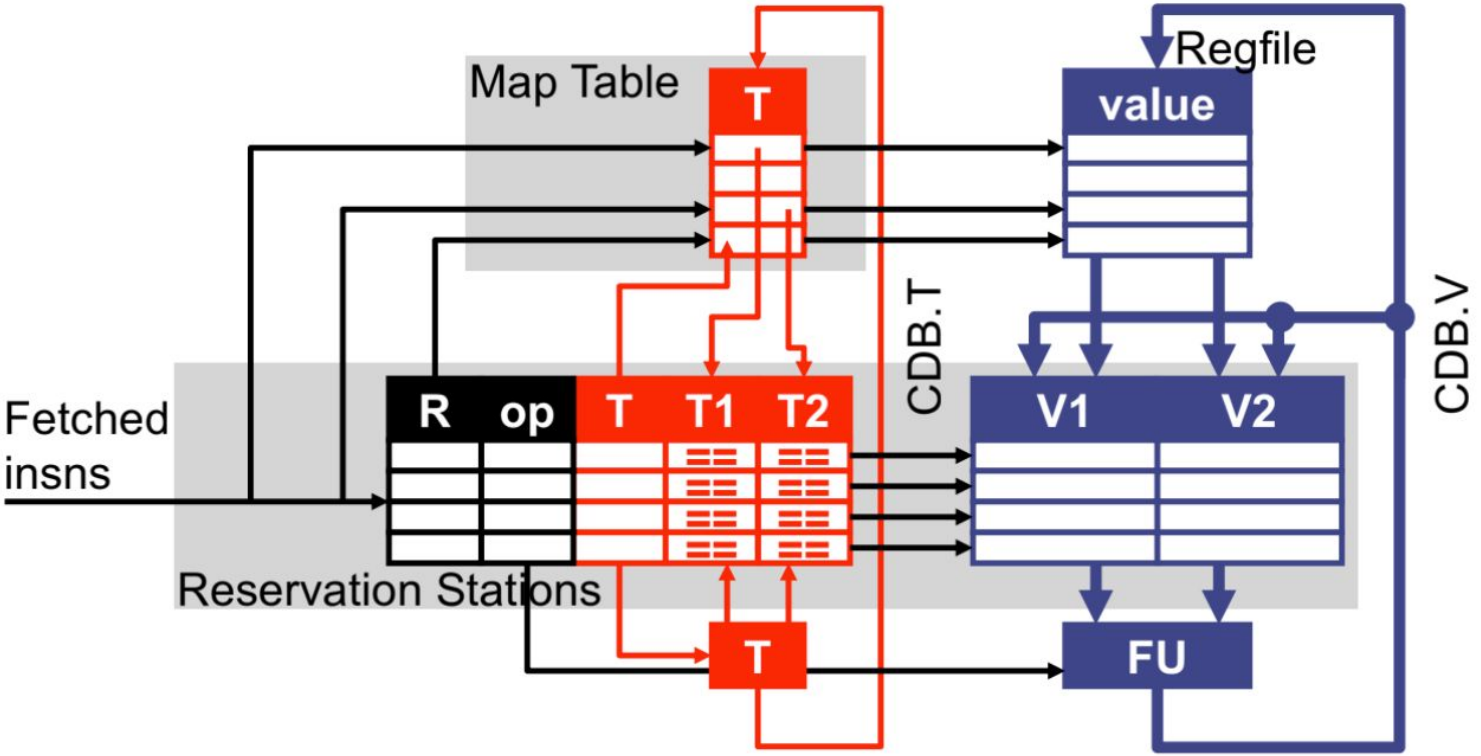
Map Table	
Reg	T
f0	
f1	RS#2
f2	RS#4
r1	

CDB	
T	V

Reservation Stations								
T	FU	busy	op	R	T1	T2	V1	V2
1	ALU	no						
2	LD	yes	ldf	f1	-	-	-	[r1]
3	ST	yes	stf	-	RS#4	-	-	[r1]
4	FP1	yes	mulf	f2	-	RS#2	[f0]	-
5	FP2	no						

allocate

Tomasulo's Structures - Another View



Tomasulo's Stages

Fetch:
- Same as always

Dispatch:
- Structural hazard ? stall : allocate RS entry, copy ready register values, set tags for non-ready values, update map table

Issue:
- RAW hazard ? wait : read registers and go to execute

Execute:
- Perform operation, notify instruction status table when done

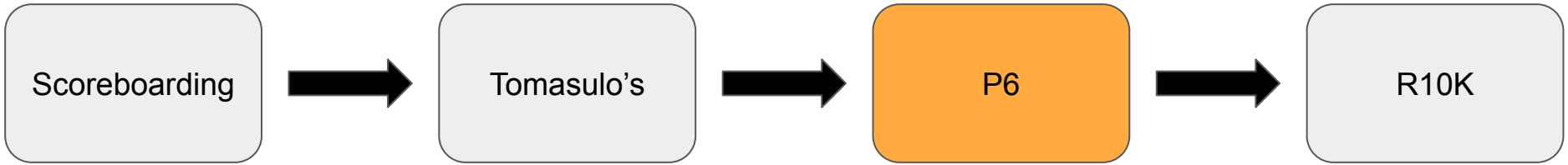
Writeback:
- Destination register still matches map table ? clear map table and write to register file.
- Broadcast on CDB and update waiting RS entries
- Free RS entry

A Tomasulo's Question

- What exact mechanisms prevent WAR and WAW hazards in Tomasulo's?

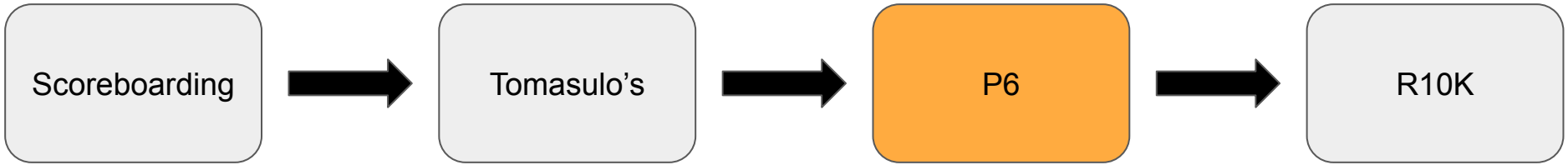
Out-of-order Roadmap

Copy-based register renaming →
No WAR/WAW stalls



Out-of-order Roadmap

Copy-based register renaming →
No WAR/WAW stalls



Precise state - can handle
exceptions and mispredictions

Precise State

What is precise state?

P6 Overview

- P6 adds the ROB (reorder buffer) to support precise state
  - Split up writeback into complete & retire
- Like Tomasulo's, P6 uses **copy-based register renaming** to eliminate false dependencies
- On a misprediction:
  1. Wait until the head of the ROB reaches the mispredicted branch.
  2. Move tail of ROB to one past the branch, squash the reservation station and in-flight instructions.

P6 Structures

ROB							
ht	#	Insn	R	V	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	
f1	
f2	
r1	

CDB	
T	V

Reservation Stations								
#	FU	busy	op	T	T1	T2	V1	V2
1	ALU	no						
2	LD	no						
3	ST	no						
4	FP1	no						
5	FP2	no						

P6 Cycles - Dispatch and Issue

Dispatch: Stall on structural hazard in ROB/RS

- Allocate ROB and RS entry
- Set RS tag to ROB#
- Set Map Table entry to ROB# and clear “ready-in-ROB” bit (i.e. the “+”)
- Read ready registers into the RS (either from the ROB or ARF)

Issue: Stall on structural hazard with FUs, or if source register values are not ready

P6 Cycles - Execute and Complete

Execute: immediately follows issue

- RS entry is cleared

Complete: stall on structural hazard in the CDB

- Broadcast ROB# and value on CDB
- Mark ROB entry as complete and write value into ROB
- Write value into RS for entries with a matching ROB# tag
- If not overwritten, mark Map Table entry “ready-in-ROB” bit (+)

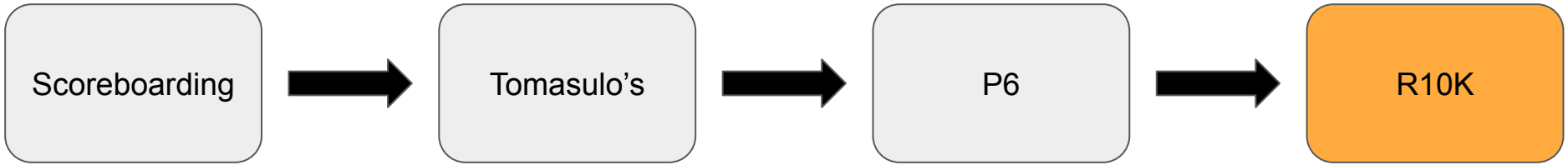
P6 Cycles - Retire (aka Commit)

Commit: stall if the head of the ROB is not complete

- Handle branch mispredictions/exceptions
- Write ROB head value into ARF
- Free ROB entry

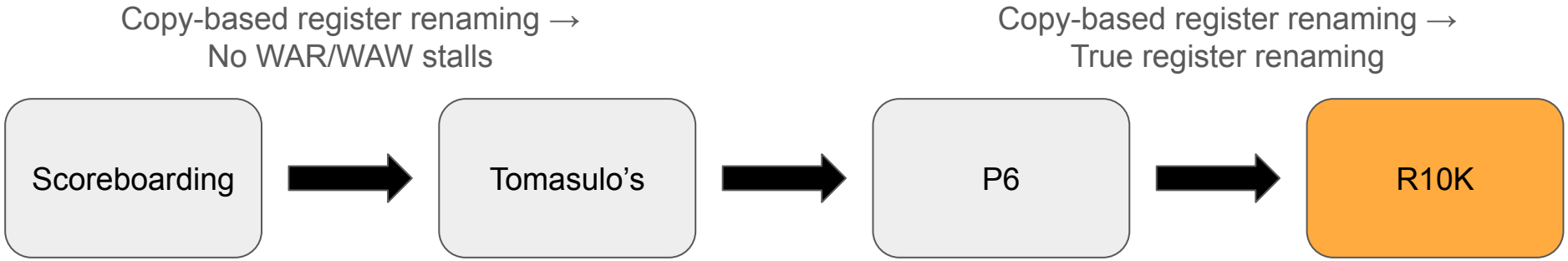
Out-of-order Roadmap

Copy-based register renaming →
No WAR/WAW stalls



Precise state - can handle
exceptions and mispredictions

Out-of-order Roadmap



Precise state - can handle
exceptions and mispredictions

R10K Overview

- P6 copy-based register renaming scheme results in a lot of value movement
  - Large buses, complicated routing, slower clock
- Instead of copies, have one larger physical register file (PRF) that stores all values
  - Just need to pass around pointers to PRF entries
  - Can physically place right next to functional units
  - Implements **true register renaming**
- Free list tracks unallocated physical registers.

R10K 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				

R10K Cycles - Dispatch and Issue

Dispatch: Stall on structural hazard in ROB/RS/Physical registers

- Allocate ROB and RS entry
- Allocate new physical register and update map table
- Record previously mapped physical register (Told)

Issue: Stall on structural hazard with FUs, or if source registers are not ready

R10K Cycles - Execute and Complete

Execute: immediately follows issue

- RS entry is cleared

Complete: stall on structural hazard in the CDB

- Write value to PRF
- Broadcast physical register tag on CDB
- Mark ROB entry as complete
- Mark RS entries with matching tags as ready
- If not overwritten, mark Map Table entry “ready” bit (+)

R10K Cycles - Retire

Commit: stall if the head of the ROB is not complete

- Handle branch mispredictions/exceptions
- Free ROB entry
- Free previous physical register (Told)
- Commit new physical register (T) by writing to architectural map table