### Exam Review 1

# exam length

#### approx. 75 minutes

approx. 3 minutes for less-than-sentence answer1-2 minutes for multiple choice/true false5 minutes for long answer/calculation

hope to get room until 7pm

## exam format

short answer questions less than one sentence answers multiple choice/true/false a lot about CPU design techniques

a few longer questions write (pseudo)code one-to-two sentence explanation

## exam focus

- will not ask "what was done in paper X"
- focus on conceptual questions not definitions
- few "what will ROB/CPU/CC/etc." do questions should all be generic enough to not require memorizing CPU
- code to read/write in generic assembly or C

### most requested topics

out-of-order:

reorder buffers/precise exceptions reg. renaming/reservation stations/instr. queues

cache coherency

vector instructions

## Recall: gem5 pipeline



# Recall: gem5 pipeline



## renaming motivation: false conflicts

С

R3 🗸	— R1	. + R	2 // A
R2 ∢	⊢ R2	. + 4	// B
R4 ∢	<u> </u>	ABE	//
reg	valu	or- e der	
R1	1	1	
R2	2	6	
R3	0	3	
R4	0	M[6]	

better to compute B earlier (start load f no real dependency between A and B

## renaming motivation: false conflicts



better to compute B earlier (start load f no real dependency between A and B

## renaming example

rename map (initial)

initial free list: X3, X8, X12, X15, X21

rename map (final)

final free list: X15, X21

## renaming data structures

current name map update oninstruction rename

name map for exceptions update on instruction commit

free list

remove from on instruction rename add to on instruction commit

## a code example

```
Loop: R3 \leftarrow M[R0]
R1 \leftarrow M[R3]
R1 \leftarrow R1 + 1
R4 \leftarrow R3 - R2
M[R3] \leftarrow R1
IF R4 != 0 GOTO Loop
```

#### exercise: rename this

initial map:  $R0 \rightarrow X0$  $R1 \rightarrow X1$  $R2 \rightarrow X2$  $R3 \rightarrow X3$  $R4 \rightarrow X4$ initial free list: X5, X6, X7, X8, X9, X10, X11

 $\begin{array}{rcl} \mathsf{R3} & \leftarrow & \mathsf{M}[\mathsf{R0}] \\ \mathsf{R1} & \leftarrow & \mathsf{M}[\mathsf{R3}] \\ \mathsf{R1} & \leftarrow & \mathsf{R1} + 1 \\ \mathsf{R4} & \leftarrow & \mathsf{R3} - \mathsf{R2} \\ \mathsf{M}[\mathsf{R3}] & \leftarrow & \mathsf{R1} \\ \textbf{IF} & \mathsf{R4} & != & 0 & \textbf{GOTO Loop} \\ \textit{// branch predicted:} \\ \mathsf{R3} & \leftarrow & \mathsf{M}[\mathsf{R0}] \\ \mathsf{R1} & \leftarrow & \mathsf{M}[\mathsf{R3}] \end{array}$ 

# exercise: rename this (answer)

	i chunicu	
final map:	X5 ← M[X0]	$R3 \leftarrow M$
R0→X0	X6 ← M[X5]	$\texttt{R1} \leftarrow \texttt{M}$
$R1 \rightarrow X7$	$X7 \leftarrow X6 + 1$	$\texttt{R1} \leftarrow \texttt{R}$
$R2 \rightarrow X5$	$X8 \leftarrow X5 - X2$	$R4 \leftarrow R$
	$M[X6] \leftarrow X7$	M[R3] ↔
R3→X9	IF X8 != 0 GOTO Loop	IF R4 !
$R4 \rightarrow X8$	<pre>// branch predicted:</pre>	// bran
final free list:	$X9 \leftarrow M[X0]$	$R3 \leftarrow M$
X11	X10 $\leftarrow$ M[R3]	$\texttt{R1} \leftarrow \texttt{M}$

renamed

## exercise: reorder buffer contents

#### renamed

$X5 \leftarrow M[X0] // A$
$X6 \leftarrow M[X5] // B$
$X7 \leftarrow X1 + 1 // C$
$X8 \leftarrow X6 - X5 // D$
$M[X6] \leftarrow X7 // E$
<b>IF</b> X8 != 0 <b>GOTO</b> Loop // F
// branch predicted:
$X9 \leftarrow M[X0] // A$
$X10 \leftarrow M[R3] = 10^{-1} B$
original
R3 ← M[R0]
$R_1 \leftarrow M[R_3]$
$R_1 \leftarrow R_1 + 1$
$R_1 \leftarrow R_3 - R_2$
$M[R3] \leftarrow R1$
TE R4 $\downarrow$ = 0 GOTO Loop
// branch predicted:
$R_3 \leftarrow M[R_0]$
$R1 \leftarrow M[R3]$
KT V HEKOL

log.	reg	prev. phys.	store?	except?	ready?
R3		Х3	no	none	no
	log. R3	log. reg R3	log. reg phys. phys. R3 X3	log. reg phys. store? R3 X3 no	log. regphys.store?except?R3X3nonone

### exercise: reorder buffer contents

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	Х3	no	none	no
В	R1	X1	no	none	no
С	R1	X6	no	none	no
D	R4	X4	no	none	no
Е			yes	none	no
F			no	none	no
А	R3	X5	no	none	no
В	R1	X6	no	none	no

#### renamed

$$\begin{array}{rcl} X5 &\leftarrow & M[X0] & // A \\ X6 &\leftarrow & M[X5] & // B \\ X7 &\leftarrow X1 + 1 & // C \\ X8 &\leftarrow X6 - X5 & // D \\ M[X6] &\leftarrow X7 & // E \\ \textbf{IF X8 } != 0 & \textbf{GOTO Loop} & // F \\ // branch predicted: \\ X9 &\leftarrow & M[X0] & // A \\ X10 &\leftarrow & M[R3] & // B \\ \hline & & \textbf{Original} \\ R3 &\leftarrow & M[R0] \\ R1 &\leftarrow & R[R3] \\ R1 &\leftarrow & R1 + 1 \\ R4 &\leftarrow & R3 - R2 \\ M[R3] &\leftarrow & R1 \\ \textbf{IF R4 } != 0 & \textbf{GOTO Loop} \\ // branch predicted: \\ R3 &\leftarrow & M[R0] \\ R1 &\leftarrow & M[R3] \\ \end{array}$$

renameu	rer	ıar	ned
---------	-----	-----	-----

PC	log. reg	prev. phys.	store?	except?	ready?	$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$
А	R3	ХЗ	no	none	yes	₩ \$8 <sup>i</sup> × x <sup>5</sup> - x <sup>2</sup> // D
В	R1	X1	no	none	yes	M[X6] ← X7 // E
С	R1	X6	no	none	no	<b>IF</b> X8 != 0 <b>GOTO</b> Loop // F
D	R4	X4	no	none	yes	// branch predicted:
Е			yes	none	no	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
F			no	none	no	$X10 \leftarrow M[X3] / B$ $X11 \leftarrow X10 + 1 / C$
A	R3	X5	no	none	yes	$X12 \leftarrow X10 - X2 // D$
В	R1	X7	no	fault	yes	
С	R1	X10	no	none	no	
D	R4	X8	no	none	yes	<b>⊸</b> head
		+ion o	licnat	had?	what	aammit action to taka?

is exception dispatched? what commit action to take?

#### rename map (for next rename)

PC	log. reg	prev. phys.	store?	except?	ready?	log. phys. R0 X0
A	R3	Х3	no	none	yes	R1 X1
в	R1	X1	no	none	yes	$\mathbb{R}_{11}^2 \mathbb{X}_{11}^1$
С	R1	X6	no	none	yes	<u>R3 X9</u>
D	R4	X4	no	none	yes	R4 X12
E			yes	none	yes	
F			no	none	yes	
А	R3	X5	no	none	yes	fund line
В	R1	X7	no	fault	yes	tree list:
С	R1	X10	no	none	no	X11, <mark>X3</mark>
D	R4	X8	no	none	yes	<b>↓</b> —head

#### rename map (for next rename)

PC	log. reg	prev. phys.	store?	except?	ready?	log. phys. R0 X0
А	R3	Х3	no	none	yes	
В	R1	X1	no	none	yes	$\frac{R2}{R2}$
С	R1	X6	no	none	yes	R3 X9
D	R4	X4	no	none	yes	<u> </u>
E			yes	none	yes	free list:
F			no	none	yes	X11 X3 X1
А	R3	X5	no	none	yes	
В	R1	X7	no	fault	yes	▲ 10, 14, 10
С	R1	X10	no	none	no	
D	R4	X8	no	none	yes	<b>↓</b> —head

#### rename map (for next rename)

						_		
DC	log reg	prev.	at a wa?	ovcont?	roady?		log.	phys.
PC	log. leg	phys.	store?	excepti	ready!		R0	X0
A	R3	X3	no	none	yes		R1	X11
B	R1	X 1	no	none	ves		R2	X2
0		N1 N6		none	yee	- 1	R3	X9
C	КТ	X6	no	none	yes			
D	R4	X4	no	none	yes		R4	$\frac{12}{10}$
E			yes	none	yes			
F			no	none	yes	1		free list.
А	R3	X5	no	none	yes		_	V11 $V2$ $V1$
В	R1	X7	no	fault	yes		tail	XII, X3, XI,
С	R1	X10	no	none	no		head	X6, X4, X5,
D	R4	X8	no	none	yes			X12,

#### rename map (for next rename)

		prev.					log.	phys.
PC	log. reg	phys.	store?	except?	ready?		R0	X0
Α	R3	X3	no	none	yes	1	R1	<del>X11</del> X10
B	R1	X1	no	none	ves	1	R2	X2
c		X6	no	none	ves	-	R3	X9
		NU V4	no	none	ves	-	R4	<del>X12</del> X8
5	R <del>4</del>	Λ <del>4</del>	NOS	none	VOS	-		
E			yes	none	yes			
F			no	none	yes			free list <sup>.</sup>
A	R3	X5	no	none	yes			V11 V2 V1
В	R1	X7	no	fault	yes		_tail,	XII, X3, XI,
С	R1	X10	no	none	no	-		X6, X4, X5,
D	R4	X8	no	none	yes			X12, X11,

#### rename map (for next rename)

PC	log. reg	prev. phys.	store?	except?	ready?		log. R0	phys. X0
A	R3	ХЗ	no	none	yes	1	<u>R1</u>	X11 X10 X7
В	R1	X1	no	none	yes	1	R2	X2
- C	D1	Y6	no	none	ves	1	R3	X9
C	NT.	<u></u>		lione	,	-	R4	X12 X8
D	R4	Х4	no	none	yes			//12 //0
Е			yes	none	yes			
F			no	none	yes	]		free list.
A	R3	X5	no	none	yes		head	$V_{11}$ $V_{2}$ $V_{1}$
В	R1	X7	no	fault	yes		-tail	$\Lambda II, \Lambda 3, \Lambda I,$
С	R1	X10	no	none	no		_	X6, X4, X5,
D	R4	X8	no	none	yes	]		X12, X11, X10

# **ROB** exception processing

- MIPS R10000 method:
- ROB has old mapping
- forwards: add to free list until exception
- backwards: update mapping until/including exception

## alternate ROB organization

can store current physical register instead of previous

commit stage maintains separate name map

## Recall: gem5 pipeline



busy list: X5, X6, X7, X8, X9, X10

#### instr. queue

Χ5	$\leftarrow M[X0]$
Х6	$\leftarrow M[X5]$
Χ7	$\leftarrow \texttt{X1} + \texttt{1}$
Χ8	$\leftarrow$ X6 - X5
IF	X8 != 0 GOTO Loop
Х9	$\leftarrow M[X0]$

busy list: X5, X6, X7, X8, X9, X10



can't start instructions with busy inputs

busy list: X5, X6, X7, X8, X9, X10



can start these (requirements not busy) (how many? depends on available functional units)

busy list: <del>X5</del>, X6, X7, X8, X9, X10



X5 no longer busy — check queue for matches

# Recall: MOESI

**Modified** value is different than memory *and* I am the only one who has it

- **Owned** value is different than memory *and* I must update memory
- **Exclusive** value is same as memory *and* I am the only one who has it
- Shared value is same as memory or cache in Owned state
- Invalid I don't have the value

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 --- I I I I

- 1: read
- 1: write
- 2: write
- 3: read
- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes --- I I I I

- 1: read
- 1: write
- 2: write
- 3: read
- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes --- I I I I

- 1: read E I I I read from memory
- 1: write
- 2: write
- 3: read
- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory
action CPU1 CPU2 CPU3 CPU4 notes
--- I I I I
1: read E I I I read from memory
1: write M I I I entirely local

- 2: write
- 3: read
- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory
action CPU1 CPU2 CPU3 CPU4 notes
--- I I I I
1: read E I I I read from memory
1: write M I I I entirely local
2: write I M I I send invalidate

- 3: read
- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes Т Τ Ι Ι 1: read E I I I read from memory 1: write M I I I entirely local I I send invalidate 2: write I M I 3 reads from 2 Т S 3: read 0

- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes Т Т Τ Ι 1: read E I I I read from memory I I 1: write M I entirely local I I 2: write I M send invalidate S I 3 reads from 2 I 3: read 0 I 1 reads from 2 S 0 S

- 1: read
- 2: evict
- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes Т Т Τ Ι 1: read E I I I read from memory I Ι 1: write M Ι entirely local Ι 2: write I М Ι send invalidate I 3 reads from 2 I S 3: read 0 I 1 reads from 2 1: read S 0 S Т S Ι S 2: evict 2 writes to memo

- 3: write
- 3: read

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

3: read

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes Т Т Т Ι I E Ι I 1: read read from memory I Τ 1: write M Τ entirely local I send invalidate 2: write I М I Ι I 3: read S 3 reads from 2 0 Ι S 0 S 1 reads from 2 1: read I 2 writes to memo S Ι S 2: evict Τ send invalidate 3: write Т Τ М

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 notes Т Т Т Τ I E Ι 1: read I read from memory Τ I entirely local 1: write М Τ Ι 2: write I М Ι send invalidate Ι 3: read I S 3 reads from 2 0 Ι S S 1 reads from 2 1: read 0 S S I 2: evict Ι 2 writes to memo I 3: write Ι Т send invalidate М Т Т Ι entirely local 3: read М

## directory states

Remote-Invalid — not stored elsewhere

Remote-Dirty — stored elsewhere and exclusive

Remote-Shared — possibly stored elsewhere

plus list of stored locations

ac	tion	CPU1	CPU2	CPU3	CPU4	dirctory	at	1
	-	I	I	I	I			
1:	read	Е	I	I	I			
1:	write	М	I	I	I			
2:	write	I	М	I	I			
3:	read	I	S	S	I			
1:	read	S	S	S	I			
2:	evict	S	I	S	I			
3:	write	I	I	М	I			
3:	read	I	I	М	I			

act	tion	CPU1	CPU2	CPU3	CPU4	dirctory at 1	
	-	I	I	I	I		
1:	read	Е	I	I	I	R-I	
1:	write	М	I	I	I		
2:	write	I	М	I	I		
3:	read	I	S	S	I		
1:	read	S	S	S	I		
2:	evict	S	I	S	I		
3:	write	I	I	М	I		
3:	read	I	I	М	I		

act	tion	CPU1	CPU2	CPU3	CPU4	dirctory	at	1
	_	I	I	I	I			
1:	read	Е	I	I	I	R-I		
1:	write	М	I	I	I	R-I		
2:	write	I	М	I	I			
3:	read	I	S	S	I			
1:	read	S	S	S	I			
2:	evict	S	I	S	I			
3:	write	I	I	М	I			
3:	read	I	I	М	I			

action		CPU1	CPU2	CPU3	CPU4	dirctory	at	1
	-	I	I	I	I			
1:	read	Е	I	I	I	R-I		
1:	write	М	I	I	I	R-I		
2:	write	I	М	I	I	R-D 2		
3:	read	I	S	S	I			
1:	read	S	S	S	I			
2:	evict	S	I	S	I			
3:	write	I	I	М	I			
3:	read	I	I	М	I			

act	ion	CPU1	CPU2	CPU3	CPU4	dirctory at 1
	-	I	I	I	I	
1:	read	Е	I	I	I	R-I
1:	write	М	I	I	I	R-I
2:	write	I	М	I	I	R-D 2
3:	read	I	S	S	I	R-S 23
1:	read	S	S	S	I	
2:	evict	S	I	S	I	
3:	write	I	I	М	I	
3:	read	I	I	М	I	

act	cion	CPU1	CPU2	CPU3	CPU4	dirctory at 1
	-	I	I	I	I	
1:	read	Е	I	I	I	R-I
1:	write	М	I	I	I	R-I
2:	write	I	М	I	I	R-D 2
3:	read	I	S	S	I	R-S 23
1:	read	S	S	S	I	R-S 123
2:	evict	S	I	S	I	
3:	write	I	I	М	I	
3:	read	т	т	М	Т	

act	tion	CPU1	CPU2	CPU3	CPU4	dirctory at 1
	-	I	I	I	I	
1:	read	Е	I	I	I	R-I
1:	write	М	I	I	I	R-I
2:	write	I	М	I	I	R-D 2
3:	read	I	S	S	I	R-S 23
1:	read	S	S	S	I	R-S 123
2:	evict	S	I	S	I	R-S 123
3:	write	I	I	М	I	
3:	read	Ι	Ι	М	Ι	

act	ion	CPU1	CPU2	CPU3	CPU4	dirctory at 1
		I	I	I	I	
1: 1	read	Е	I	I	I	R-I
1: \	write	М	I	I	I	R-I
2: \	write	I	М	I	I	R-D 2
3: 1	read	I	S	S	I	R-S 23
1: 1	read	S	S	S	I	R-S 123
2: 6	evict	S	I	S	I	R-S 123
3: \	write	I	I	М	I	R-D 3
3: 1	read	Ι	Ι	М	Ι	

action	CPU1	CPU2	CPU3	CPU4	dirctory at 1
	I	I	I	I	
1: read	Е	I	I	I	R-I
1: write	М	I	I	I	R-I
2: write	I	М	I	I	R-D 2
3: read	I	S	S	I	R-S 23
1: read	S	S	S	I	R-S 123
2: evict	S	I	S	I	R-S 123
3: write	I	I	М	I	R-D 3
3: read	I	I	М	I	R-D 3

#### vector exercise

```
void vector_add_one(int *x, int length) {
    for (int i = 0; i < length; ++i) {
        x[i] += 1;
    }
}</pre>
```

exercise: write as a vector machine program with 64-element vectors

vector length register or predicate (mask) registers

#### vector exercise answer

```
void vector_add_one(int *x, int length) {
     for (int i = 0; i < length; ++i) {</pre>
          x[i] += 1;
     }
}
// R1 contains X, R2 contains length
       VL \leftarrow R2 \text{ MOD } 64
Loop: IF R2 <= 0, goto End
       V1 \leftarrow MEMORY[R1]
       V1 \leftarrow V1 + 1
       MEMORY[R1] \leftarrow V1
        R2 \leftarrow R2 - VL
       VL \leftarrow 64
       goto Loop
End:
```

## relaxed memory models ex 1

reasons for each reordering:

loads before loads loads before stores stores before stores

## relaxed memory models ex 2

What can happen?

sequential?

move loads after stores?

move loads after loads?

## extra OH?

I could provide extra office hours this week...

Wednesday morning or afternoon

Thursday morning