

# 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

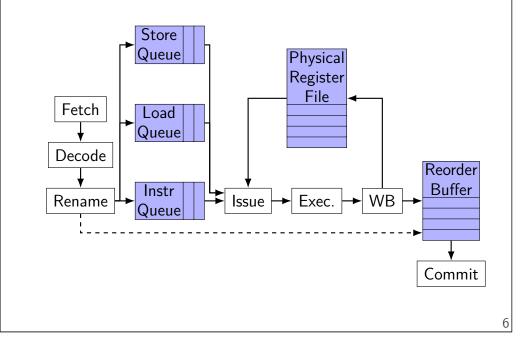
out-of-order:

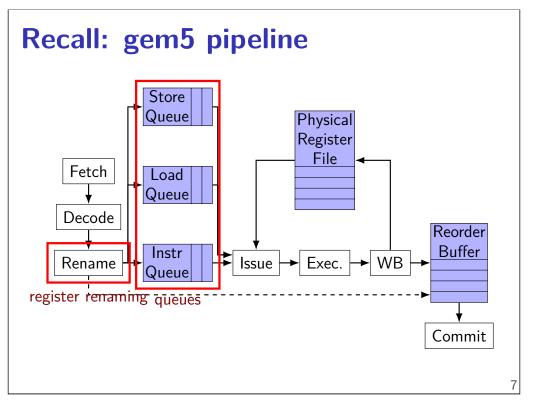
reorder buffers/precise exceptions reg. renaming/reservation stations/instr. queues

cache coherency

vector instructions

# **Recall: gem5 pipeline**





# renaming motivation: false conflicts

			2 // A
R2 -	← R2	. + 4	// B
R4 →	← M[ init	ABE	// (
reg	init valu	or- e der	
R1	1	1	
R2	2	6	
R3	0	3	
R4	0	M[6]	

5

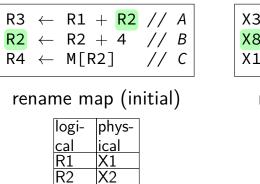
better to compute B earlier (start load f no real dependency between A and B

# renaming motivation: false conflicts

R3	$\leftarrow$	R1	+ R	2 //	A
R2	$\leftarrow$	R2	+ 4		В
R4	←  in	М[	ABE	BAC	/ C
reg		1	or- der	or-	
	-va	ιu	der	der	
R1	1		1	1	
R2	2		6	6	
R3	0		3	7	
R4	0		M[6]	M[6]	

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

### renaming example



X31

X23

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

R3

R4

ХЗ	$\leftarrow$	X1	+	X2	//	Α
X8	$\leftarrow$	Х2	+	4	//	В
X12	2 ~	- M	[X8	3]	//	/ C

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

9

10

exercise: rena	me this	exercise: r	ename this (answer	·)
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	$R3 \leftarrow M[R0]$ $R1 \leftarrow M[R3]$ $R1 \leftarrow R1 + 1$ $R4 \leftarrow R3 - R2$ $M[R3] \leftarrow R1$ IF R4 != 0 GOTO Loop // branch predicted: R3 \leftarrow M[R0] $R1 \leftarrow M[R3]$	final map: $R0 \rightarrow X0$ $R1 \rightarrow X7$ $R2 \rightarrow X5$ $R3 \rightarrow X9$ $R4 \rightarrow X8$ final free list: X11	renamed $X5 \leftarrow M[X0]$ $X6 \leftarrow M[X5]$ $X7 \leftarrow X6 + 1$ $X8 \leftarrow X5 - X2$ $M[X6] \leftarrow X7$ IF X8 != 0 GOTO Loop // branch predicted: $X9 \leftarrow M[X0]$ $X10 \leftarrow M[R3]$	$R3 \leftarrow M$ $R1 \leftarrow M$ $R1 \leftarrow R$ $R4 \leftarrow R$ $M[R3] \leftarrow$ $IF R4 !$ $// bran$ $R3 \leftarrow M$ $R1 \leftarrow M$
	adapted from H&P Fig 3.54	12	adapted fro	om H&P Fig 3.54 13

14

# exercise: reorder buffer contents

PC	log.	reg	prev. phys.	store?	except?	ready?
A	R3		X3	no	none	no

### renamed

$\begin{array}{rcl} X5 \ \leftarrow \ M[X0] & // \ A \\ X6 \ \leftarrow \ M[X5] & // \ B \\ X7 \ \leftarrow \ X1 \ + \ 1 & // \ C \\ X8 \ \leftarrow \ X6 \ - \ X5 & // \ D \\ M[X6] \ \leftarrow \ X7 & // \ E \\ \textbf{IF} \ X8 \ != \ 0 \ \textbf{GOTO Loop} \ // \ F \\ // \ branch \ predicted: \\ X9 \ \leftarrow \ M[X0] \ // \ A \\ X10 \ \leftarrow \ M[R3] \ . \ // \ B \\ \textbf{Original} \end{array}$
$\begin{array}{rcl} R3 & \leftarrow & M[R0] \\ R1 & \leftarrow & M[R3] \\ R1 & \leftarrow & R1 + 1 \\ R4 & \leftarrow & R3 - R2 \\ M[R3] & \leftarrow & R1 \\ \textbf{IF} & R4 & != 0 & \textbf{GOTO Loop} \\ \textit{// branch predicted:} \\ R3 & \leftarrow & M[R0] \\ R1 & \leftarrow & M[R3] \end{array}$
adapted from H&P Fig 3.54

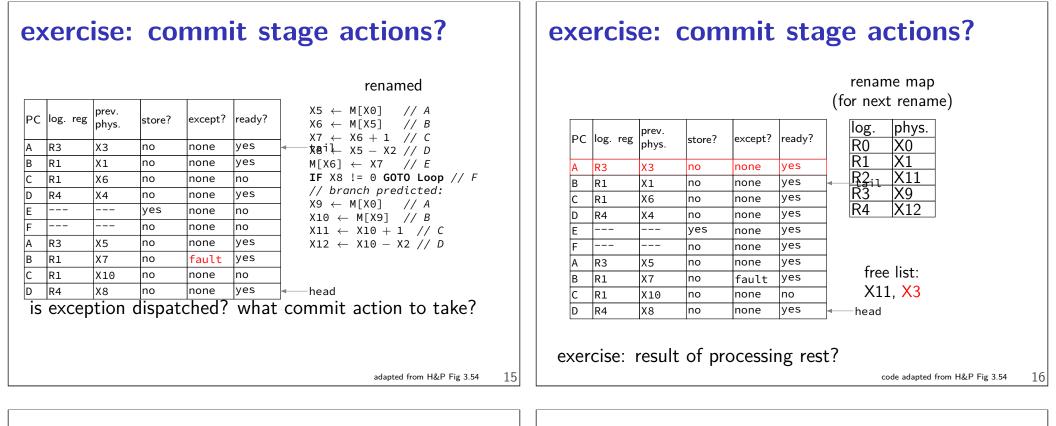
### exercise: reorder buffer contents

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
E			yes	none	no
F			no	none	no
А	R3	X5	no	none	no
В	R1	X6	no	none	no

### renamed

$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$
R3 ← M[R0] R1 ← M[R3] R1 ← R1 + 1 R4 ← R3 - R2 M[R3] ← R1 IF R4 != 0 GOTO Loop // branch predicted: R3 ← M[R0]
$R1 \leftarrow M[R3]$

adapted from H&P Fig 3.54 14



PC

A

в

C

D

E

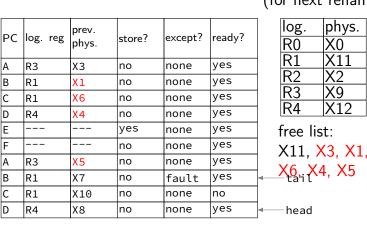
IF.

A

В

С

# exercise: commit stage actions?



rename map (for next rename)

phys.

X0

X11

X2

X9

X12

# exercise: commit stage actions?

prev. except? ready? log. reg store? phys. R3 Х3 no none yes R1 X1 yes no none R1 X6 no none yes R4 X4 ves no none \_\_\_ \_\_\_ yes none yes \_\_\_ --no none yes R3 Χ5 no none yes R1 Χ7 no fault yes R1 X10 no none no R4 X8 no none yes

rename map (for next rename)

log.	phys.
R0	X0
R1	X11
R2	X2
R3	X9
R4	<del>X12</del>

free list: X11, X3, X1, X6, X4, X5, head X12.

tail

exercise: result of processing rest?

exercise: result of processing rest?

code adapted from H&P Fig 3.54

17

code adapted from H&P Fig 3.54 18

### exercise: commit stage actions? exercise: commit stage actions? rename map rename map (for next rename) (for next rename) phys. phys. log. log. prev. prev. PC log. reg except? store? ready? PC log. reg store? except? ready? phys. R0 X0 phys. R0 X0 R1 X11 X10 R1 X11 X10 X7 R3 X3 no none yes A R3 ΧЗ no none yes A R2 X2 R2 X2 в R1 X1 yes в R1 X1 yes no none no none X9 R3 R3 IC. R1 X6 no none yes C R1 X6 no none yes R4 D

X9 X12 X8 X12 X8 R4 R4 X4 no yes D R4 X4 no ves none none \_\_\_ \_\_\_ E \_\_\_ --yes yes yes yes none none \_\_\_ \_\_\_ F \_\_\_\_ --no yes no yes none none free list: free list: Χ5 no none yes R3 Χ5 no none yes A head tail, X11, X3, X1, X11, X3, X1, Χ7 В R1 no fault yes Χ7 no fault yes tail head X6, X4, X5, X6, X4, X5, X10 none С R1 X10 no no no none no X12, X11, D R4 X12, X11, X10 X8 yes Χ8 yes no none no none exercise: result of processing rest? exercise: result of processing rest? code adapted from H&P Fig 3.54 code adapted from H&P Fig 3.54 18 18

# **ROB** exception processing

MIPS R10000 method:

E

F

A R3

В R1

D

R1

R4

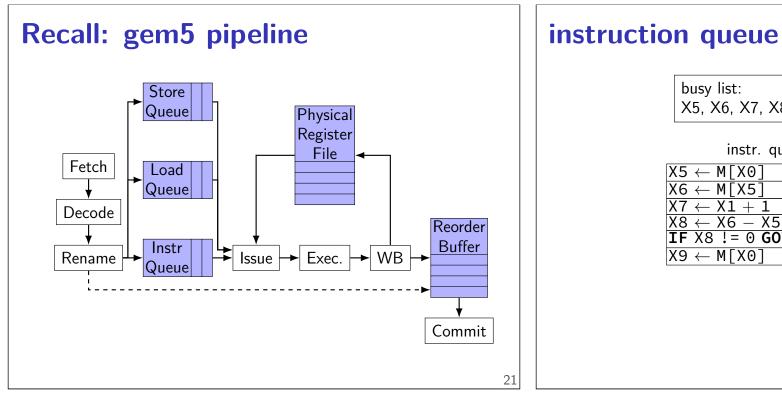
ROB has old mapping

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



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

instr. queue
$X5 \leftarrow M[X0]$
$X6 \leftarrow M[X5]$
$X7 \leftarrow X1 + 1$
$X8 \leftarrow X6 - X5$
X8 ← X6 − X5 IF X8 != 0 GOTO Loop
$X9 \leftarrow M[X0]$

instruction queue

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

instr. queue
$X5 \leftarrow M[X0]$
$X6 \leftarrow M[X5]$
$X7 \leftarrow X1 + 1$
$X8 \leftarrow X6 - X5$
IF X8 != 0 GOTO Loop
$X9 \leftarrow M[X0]$

can't start instructions with busy inputs

### instruction queue

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

instr. queue
$X5 \leftarrow M[X0]$
$X6 \leftarrow M[X5]$
$X7 \leftarrow X1 + 1$
$X8 \leftarrow X6 - X5$
<b>IF</b> X8 != 0 <b>GOTO</b> Loop
$X9 \leftarrow M[X0]$

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

22

instruction queue	Recall:	MOESI				
busy list: <del>X5</del> , X6, X7, X8, X9, X10		value is different than memory <i>and</i> I am the only one who has it				
instr. queue $X5 \leftarrow M[X0]$ $X6 \leftarrow M[X5]$ $X7 \leftarrow X1 + 1$	Owned	value is different than memory <i>and</i> I must update memory				
$\begin{array}{c} X7 \leftarrow X1 + 1 \\ \hline X8 \leftarrow X6 - X5 \\ \hline IF X8 != 0 \text{ GOTO Loop} \\ \hline X9 \leftarrow M[X0] \end{array}$	Exclusive	value is same as memory <i>and</i> I am the only one who has it				
X5 no longer busy — check queue for matches	Shared	value is same as memory <b>or cache</b> <b>in Owned state</b>				
22	Invalid	I don't have the value 23				
cache coherency exercise	cache c	oherency exercise				
Modified/Exclusive/Owned/Shared/Invalid	Modified/E	xclusive/Owned/Shared/Invalid				
invalidation-based protocol	invalidation	invalidation-based protocol				
read from remote caches or memory action CPU1 CPU2 CPU3 CPU4 I I I I	read from r action 	emote caches or memory CPU1 CPU2 CPU3 CPU4 notes I I I I I				

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

${\sf Modified/Exclusive/Owned/Shared/Invalid}$								
invalidatior	i-based	protoc	ol					
read from r	remote	caches	or me	mory				
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								

cache coherency exercise	cache coherency exercise
Modified/Exclusive/Owned/Shared/Invalid	Modified/Exclusive/Owned/Shared/Invalid
invalidation-based protocol	invalidation-based protocol
read from remote caches or memory	read from remote caches or memory
action CPU1 CPU2 CPU3 CPU4 notes	action CPU1 CPU2 CPU3 CPU4 notes
I I I I	I I I I
1: read E I I I read from memory	1: read E I I I read from memory
1: write	1: write M I I I entirely local
2: write	2: write
3: read	3: read
1: read	1: read
2: evict	2: evict
3: write	3: write
3: read	3: read

cache coherency exercise						cache c	oher	ency	v exe	rcise	2
Modified/Exclusive/Owned/Shared/Invalid					Modified/E	Modified/Exclusive/Owned/Shared/Invalid					
invalidation-based protocol					invalidation	invalidation-based protocol					
read from remote caches or memory					read from remote caches or memory						
action	CPU1	CPU2	CPU3	CPU4	notes	action	CPU1	CPU2	CPU3	CPU4	notes
	I	I	I	I			I	I	I	I	
1: read	Е	I	I	I	read from memory	1: read	Е	I	I	I	read from memory
1: write	М	I	I I	Ι	entirely local	1: write	М	I	I	I	entirely local
2: write	I	М	I	Ι	send invalidate	2: write	I	М	I	I	send invalidate
3: read						3: read	I	0	S	I	3 reads from 2
1: read						1: read					
2: evict						2: evict					
3: write						3: write					
3: read						3: read					
					25						25

### cache coherency exercise

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	Е	I	I	I	read from memory
1: write	М	I	I	I	entirely local
2: write	I	М	I	I	send invalidate
3: read	I	0	S	I	3 reads from 2
1: read	S	0	S	I	1 reads from 2
2: evict					
3: write					
3: read					
					25

### cache coherency exercise

Modified/Exclusive/Owned/Shared/Invalid

invalidation-based protocol

read from remote caches or memory CPU1 CPU2 CPU3 CPU4 action notes Ι Ι Ι Ι Е Ι Ι read from memory 1: read Ι Ι Ι entirely local 1: write М Ι Ι 2: write М send invalidate Ι Ι S 3: read Ι 0 Ι 3 reads from 2 S 1: read S 0 Ι 1 reads from 2 2: evict S Ι S Ι 2 writes to memo 3: write 3: read 25

cache coherency exercise						Ca	ache o	oher	rency	v exe	rcise	2
Modified/Exclusive/Owned/Shared/Invalid					M	Modified/Exclusive/Owned/Shared/Invalid				valid		
invalidation-based protocol						in	validatior	n-based	protoc	col		
read from re	emote	caches	or me	mory		re	ad from i	remote	caches	or me	mory	
action	CPU1	CPU2	CPU3	CPU4	notes	a	ction	CPU1	CPU2	CPU3	CPU4	notes
	I	I	I	Ι				I	I	I	I	
1: read	Е	I	I	Ι	read from memory	1	: read	Е	I	Ι	Ι	read from memory
1: write	М	I	I	Ι	entirely local	1	: write	М	I	I	I	entirely local
2: write	I	М	I	I	send invalidate	2	: write	I	М	I	I	send invalidate
3: read	I	0	S	Ι	3 reads from 2	3	: read	I	0	S	Ι	3 reads from 2
1: read	S	0	S	Ι	1 reads from 2	1	: read	S	0	S	I	1 reads from 2
2: evict	S	I	S	I	2 writes to memo	2	: evict	S	I	S	I	2 writes to memo
3: write	I	I	М	Ι	send invalidate	3	: write	I	I	М	Ι	send invalidate
3: read					25	3	: read	I	I	М	I	entirely local 25

### directory states

- Remote-Invalid not stored elsewhere
- Remote-Dirty stored elsewhere and exclusive
- Remote-Shared possibly stored elsewhere

plus list of stored locations

# directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared									
action				_	dirctory	at	1		
 1: read	I E	I I	I I	I I					
1: reau 1: write	_	I	I	I					
	I		I						
	I		S	I					
1: read		S	S	I					
	S	I	S	I					
	I		М	I					
3: read	I	I	М	I					
							27		

# directory-based coherency

Remote-Inv	alid, R	emote-	Dirty,	Remote	e-Shared	
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		
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		

### directory-based coherency

Remote-Inv	valid, R	emote-	Dirty,	Remote	e-Shared
action	CPU1	CPU2 T	CPU3	CPU4	dirctory at 1
1: read	I E	I	I	L T	R-I
1: write	M	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	

# directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared								
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				
						27		

# directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared									
action 	CPU1 I	CPU2 I	CPU3 I	CPU4 I	dirctory at	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: 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					
						27			

# directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared					
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	
3: write	I	I	М	I	
3: read	I	I	М	I	

# directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared					
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	
3: read	I	I	М	I	

### directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared					
action	CPU1	CPU2	CPU3	CPU4	dirctory at
	I	I	I	I	
1: read	Е	I	I	I	R-I
1: write	М	I	I	Ι	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
	action 1: read 1: write 2: write 3: read 1: read 2: evict 3: write	actionCPU1I1: readE1: writeM2: writeI3: readI1: readS2: evictS3: writeI	actionCPU1CPU2II1: readEI1: writeMI2: writeIM3: readIS1: readSS2: evictSI3: writeII	actionCPU1CPU2CPU3III1: readEII1: writeMII2: writeIMI3: readISS1: readSSS2: evictSIS3: writeIIM	actionCPU1CPU2CPU3CPU4IIII1: readEIII1: writeMIII2: writeIMII3: readISSI1: readSSSI2: evictSISI3: writeIIMI

28

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

directory-based coherency

1

27

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

relaxed memory models ex 1	relaxed memory models ex 2
reasons for each reordering: loads before loads loads before stores stores before stores	What can happen? X = Y = 0 CPU1: $R1 \leftarrow X$ $R2 \leftarrow Y$ $Y \leftarrow 1$ CPU2: $R1 \leftarrow X$ $X \leftarrow 1$ $R2 \leftarrow Y$ sequential? move loads after stores?
30	move loads after loads? 31
extra OH?	
I could provide extra office hours this week	
Wednesday morning or afternoon	
Thursday morning	