Exam Review 1

exam length

approx. 75 minutes

- approx. 3 minutes for less-than-sentence answer
- 1-2 minutes for multiple choice/true false
- 5 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

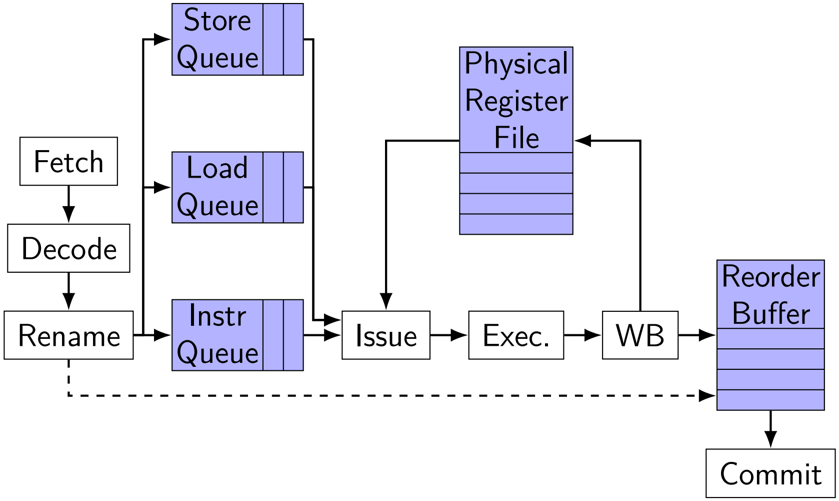
out-of-order:

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

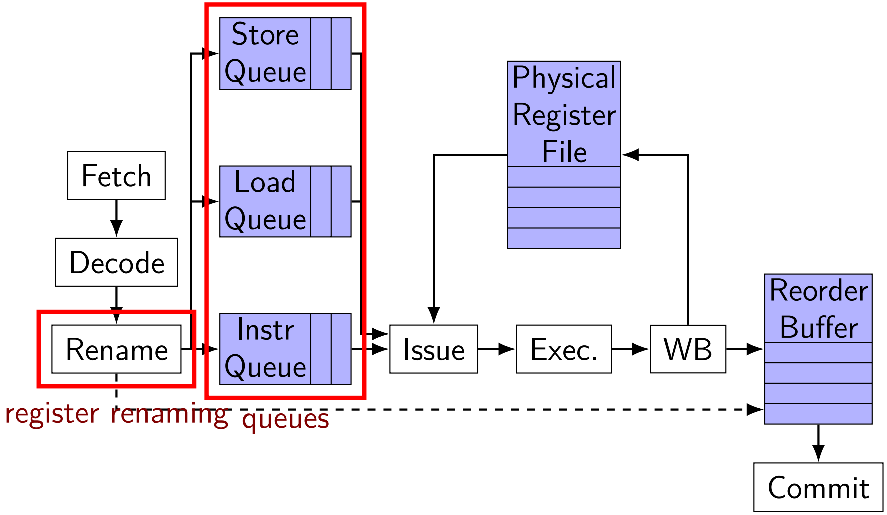
cache coherency

vector instructions

Recall: gem5 pipeline



Recall: gem5 pipeline



renaming motivation: false conflicts

```
R3 ← R1 + R2   // A
R2 ← R2 + 4    // B
R4 ← M[R2]      // C
```

reg	init. value	ABC order
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

```
R3 ← R1 + R2  // A
R2 ← R2 + 4   // B
R4 ← M[R2]    // C
```

reg	init. value	ABC order	BAC order
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

```
R3 ← R1 + R2 // A
R2 ← R2 + 4  // B
R4 ← M[R2]   // C
```

```
X3  ← X1 + X2 // A
X8  ← X2 + 4  // B
X12 ← M[X8]   // C
```

rename map (initial)

logi-cal	phys-ical
R1	X1
R2	X2
R3	X31
R4	X23

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 ← M[R0]
      R1 ← M[R3]
      R1 ← R1 + 1
      R4 ← R3 − R2
      M[R3] ← R1
      IF R4 != 0 GOTO Loop
```

exercise: rename this

initial map:

- R0→X0
- R1→X1
- R2→X2
- R3→X3
- R4→X4

initial free list:
X5, X6, X7, X8,
X9, X10, X11

```
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 ← M[R3]
```

exercise: rename this (answer)

final map:

R0→X0

R1→X7

R2→X5

R3→X9

R4→X8

final free list:

X11

renamed

```
X5 ← M[X0]
X6 ← M[X5]
X7 ← X6 + 1
X8 ← X5 − X2
M[X6] ← X7
IF X8 != 0 GOTO Loop
// branch predicted:
X9 ← M[X0]
X10 ← M[R3]
```

```
R3 ← M
R1 ← M
R1 ← R
R4 ← R
M[R3] ←
IF R4 !
// bran
R3 ← M
R1 ← M
```

exercise: reorder buffer contents

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

renamed

```
X5 ← M[X0]    // A
X6 ← M[X5]    // B
X7 ← X1 + 1   // C
X8 ← X6 − X5  // D
M[X6] ← X7    // E
IF X8 != 0 GOTO Loop // F
// branch predicted:
X9 ← M[X0]    // A
X10 ← M[R3]   // B
```

original

```
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 ← M[R3]
```

exercise: reorder buffer contents

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	no
B	R1	X1	no	none	no
C	R1	X6	no	none	no
D	R4	X4	no	none	no
E	---	---	yes	none	no
F	---	---	no	none	no
A	R3	X5	no	none	no
B	R1	X6	no	none	no

renamed

```
X5 ← M[X0]    // A
X6 ← M[X5]    // B
X7 ← X1 + 1   // C
X8 ← X6 − X5  // D
M[X6] ← X7    // E
IF X8 != 0 GOTO Loop // F
// branch predicted:
X9 ← M[X0]    // A
X10 ← M[R3]   // B
```

original

```
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 ← M[R3]
```

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	yes
B	R1	X1	no	none	yes
C	R1	X6	no	none	no
D	R4	X4	no	none	yes
E	---	---	yes	none	no
F	---	---	no	none	no
A	R3	X5	no	none	yes
B	R1	X7	no	fault	yes
C	R1	X10	no	none	no
D	R4	X8	no	none	yes

tail

head

renamed

```
X5 ← M[X0]    // A
X6 ← M[X5]    // B
X7 ← X6 + 1   // C
X8 ← X5 − X2  // D
M[X6] ← X7    // E
IF X8 != 0 GOTO Loop // F
// branch predicted:
X9 ← M[X0]    // A
X10 ← M[X9]   // B
X11 ← X10 + 1  // C
X12 ← X10 − X2 // D
```

is exception dispatched? what commit action to take?

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	yes
B	R1	X1	no	none	yes
C	R1	X6	no	none	yes
D	R4	X4	no	none	yes
E	---	---	yes	none	yes
F	---	---	no	none	yes
A	R3	X5	no	none	yes
B	R1	X7	no	fault	yes
C	R1	X10	no	none	no
D	R4	X8	no	none	yes

tail

head

rename map
(for next rename)

log.	phys.
R0	X0
R1	X1
R2	X11
R3	X9
R4	X12

free list:
X11, X3

exercise: result of processing rest?

code adapted from H&P Fig 3.54

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?	
A	R3	X3	no	none	yes	
B	R1	X1	no	none	yes	
C	R1	X6	no	none	yes	
D	R4	X4	no	none	yes	
E	---	---	yes	none	yes	
F	---	---	no	none	yes	
A	R3	X5	no	none	yes	
B	R1	X7	no	fault	yes	tail
C	R1	X10	no	none	no	
D	R4	X8	no	none	yes	head

rename map
(for next rename)

log.	phys.
R0	X0
R1	X11
R2	X2
R3	X9
R4	X12

free list:
X11, X3, X1, X6, X4, X5

exercise: result of processing rest?

code adapted from H&P Fig 3.54

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	yes
B	R1	X1	no	none	yes
C	R1	X6	no	none	yes
D	R4	X4	no	none	yes
E	---	---	yes	none	yes
F	---	---	no	none	yes
A	R3	X5	no	none	yes
B	R1	X7	no	fault	yes
C	R1	X10	no	none	no
D	R4	X8	no	none	yes

tail

head

rename map
(for next rename)

log.	phys.
R0	X0
R1	X11
R2	X2
R3	X9
R4	~~X12~~ X8

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

exercise: result of processing rest?

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	yes
B	R1	X1	no	none	yes
C	R1	X6	no	none	yes
D	R4	X4	no	none	yes
E	---	---	yes	none	yes
F	---	---	no	none	yes
A	R3	X5	no	none	yes
B	R1	X7	no	fault	yes
C	R1	X10	no	none	no
D	R4	X8	no	none	yes

tail, head

rename map
(for next rename)

log.	phys.
R0	X0
R1	~~X11~~ X10
R2	X2
R3	X9
R4	~~X12~~ X8

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

exercise: result of processing rest?

code adapted from H&P Fig 3.54

exercise: commit stage actions?

PC	log. reg	prev. phys.	store?	except?	ready?
A	R3	X3	no	none	yes
B	R1	X1	no	none	yes
C	R1	X6	no	none	yes
D	R4	X4	no	none	yes
E	---	---	yes	none	yes
F	---	---	no	none	yes
A	R3	X5	no	none	yes
B	R1	X7	no	fault	yes
C	R1	X10	no	none	no
D	R4	X8	no	none	yes

head

tail

rename map
(for next rename)

log.	phys.
R0	X0
R1	~~X11~~ ~~X10~~ X7
R2	X2
R3	X9
R4	~~X12~~ X8

free list:
X11, X3, X1,
X6, X4, X5,
X12, X11, X10

exercise: result of processing rest?

ROB exception processing

MIPS R10000 method:

ROB has old mapping

forwards: add to free list until exception

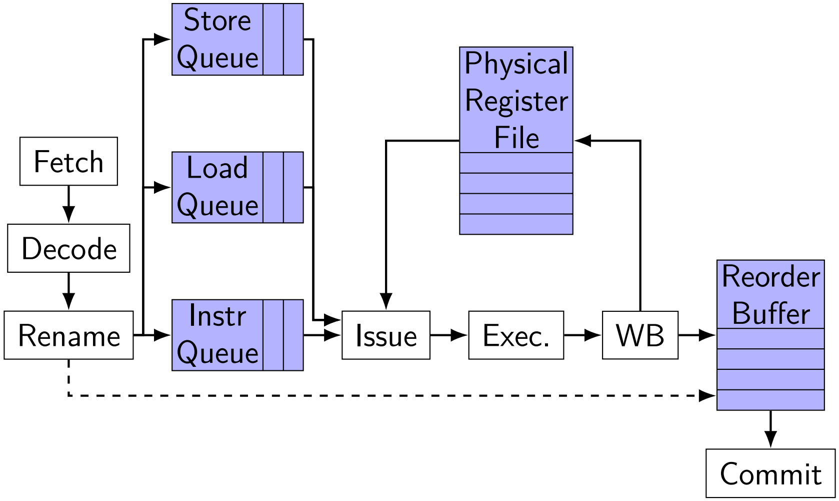
backwards: update mapping until/including exception

alternate ROB organization

can store current physical register instead of previous

commit stage maintains separate name map

Recall: gem5 pipeline



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]`

instruction queue

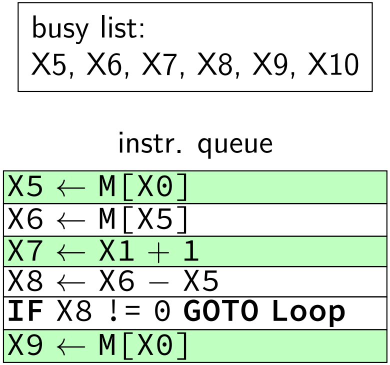
busy list:
X5, X6, X7, X8, X9, X10

instr. queue

`X5 ← M[X0]`
`X6 ← M[X5]`
`X7 ← X1 + 1`
`X8 ← X6 − X5`
`IF X8 != 0 GOTO Loop`
`X9 ← M[X0]`

can't start instructions with busy inputs

instruction queue



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

instr. queue

```
X5 ← M[X0]
X6 ← M[X5]
X7 ← X1 + 1
X8 ← X6 − X5
IF X8 != 0 GOTO Loop
X9 ← M[X0]
```

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

instruction queue

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

instr. queue

X5 ← M[X0]
X6 ← M[X5]
X7 ← X1 + 1
X8 ← X6 − X5
IF X8 != 0 **GOTO Loop**
X9 ← M[X0]

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

cache coherency exercise

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
```

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
1: write
2: write
3: read
1: read
2: evict
3: write
3: read
```

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

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    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
```

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      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
```

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     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     I     O     S     I   3 reads from 2
1: read
2: evict
3: write
3: read
```

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     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     I     O     S     I   3 reads from 2
1: read     S     O     S     I   1 reads from 2
2: evict
3: write
3: read
```

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	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	I	O	S	I	3 reads from 2
1: read	S	O	S	I	1 reads from 2
2: evict	S	I	S	I	2 writes to memo
3: write					
3: read					

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     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     I     O     S     I   3 reads from 2
1: read     S     O     S     I   1 reads from 2
2: evict    S     I     S     I   2 writes to memo
3: write    I     I     M     I   send invalidate
3: read
```

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	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	I	O	S	I	3 reads from 2
1: read	S	O	S	I	1 reads from 2
2: evict	S	I	S	I	2 writes to memo
3: write	I	I	M	I	send invalidate
3: read	I	I	M	I	entirely local

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     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I
1: write     M     I     I     I
2: write     I     M     I     I
3: read      I     S     S     I
1: read      S     S     S     I
2: evict     S     I     S     I
3: write     I     I     M     I
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I
2: write     I     M     I     I
3: read      I     S     S     I
1: read      S     S     S     I
2: evict     S     I     S     I
3: write     I     I     M     I
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I   R-I
2: write     I     M     I     I
3: read      I     S     S     I
1: read      S     S     S     I
2: evict     S     I     S     I
3: write     I     I     M     I
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I   R-I
2: write     I     M     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     M     I
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action    CPU1  CPU2  CPU3  CPU4  dirctory at 1
---        I     I     I     I
1: read    E     I     I     I    R-I
1: write   M     I     I     I    R-I
2: write   I     M     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     M     I
3: read    I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I   R-I
2: write     I     M     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     M     I
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action    CPU1  CPU2  CPU3  CPU4  dirctory at 1
---        I     I     I     I
1: read    E     I     I     I    R-I
1: write   M     I     I     I    R-I
2: write   I     M     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     M     I
3: read    I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I   R-I
2: write     I     M     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     M     I   R-D 3
3: read      I     I     M     I
```

directory-based coherency

Remote-Invalid, Remote-Dirty, Remote-Shared

```
action     CPU1  CPU2  CPU3  CPU4  dirctory at 1
---          I     I     I     I
1: read      E     I     I     I   R-I
1: write     M     I     I     I   R-I
2: write     I     M     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     M     I   R-D 3
3: read      I     I     M     I   R-D 3
```

vector exercise

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

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) {
        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:
```

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?

```
X = Y = 0
CPU1:
R1 ← X
R2 ← Y
Y ← 1
CPU2:
R1 ← X
X ← 1
R2 ← Y
```

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