

CSE 210: Computer Architecture

Lecture 10: Logical Operations

Stephen Checkoway

Oberlin College

Oct. 25, 2021

Slides from Cynthia Taylor

Announcements

- Problem Set 3 due Friday
- Lab 2 due Sunday
- Office Hours Tuesday 13:30 to 14:30

Logical Operations

- Instructions for bitwise manipulation

Operation	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bitwise AND	&	&	and, andi
Bitwise OR			or, ori
Bitwise NOT	~	~	nor

- Useful for extracting and inserting groups of bits in a word

Shift Operations



- shamt: how many positions to shift
- Shift left logical
 - Shift left and fill with 0 bits
 - sll by n bits multiplies by 2^n
- Shift right logical
 - Shift right and fill with 0 bits
 - srl by n bits divides by 2^n (unsigned only)

MIPS shift instructions

`t0 = s1 << 2`

`sll $t0, $s1, 2`



Shift left logical

- $0110\ 1001 \ll 2$ in 8 bits
 - Most significant 2 bits are **dropped**
 - 2 0s are **added** to become the least significant bits
 - Result: ~~01~~ 1010 01**00** \Rightarrow 1010 0100

Shift right logical

- 1010 1001 >>> 3 in 8 bits
 - Least significant 3 bits are **dropped**
 - 3 0s are **added** to become the most significant bits
 - Result: **000**1 0101 ~~001~~ => 0001 0101

Shift right arithmetic

- sra rd, rt, shamt
 - Shift right and copy the sign bit
- 1010 1001 >> 3 in 8 bits
 - Least significant 3 bits are **dropped**
 - 3 1s are **added** because the MSB is 1 to become the most significant bits
 - Result: **1111** 0101 ~~001~~ => 1111 0101

A new op HEXSHIFTRIGHT shifts hex numbers right by a digit. HEXSHIFTRIGHT i times is equivalent to

A. Dividing by i

B. Dividing by 2^i

C. Dividing by 16^i

D. Multiplying by 16^i

Remember Boolean Operations?

- and, or, not . . .
- Now we'll apply them to bits!
- Just think of 1 as True, and 0 as False

And Truth Table

	<i>0</i>	<i>1</i>
<i>0</i>	0	0
<i>1</i>	0	1

AND Operations

- Useful to mask bits in a word
 - Select some bits, clear others to 0

and \$t0, \$t1, \$t2

\$t2 0000 0000 0000 0000 0000 1101 1100 0000

\$t1 0000 0000 0000 0000 0011 1100 0000 0000

\$t0 0000 0000 0000 0000 0000 1100 0000 0000

AND identities

- $x \& 0 = 0$
- $x \& 1 = x$

01101001 & 11000111

A. 00010000

B. 01000001

C. 10101110

D. 11101111

If we want to zero out bits 0 – 3 in a byte we should
AND with

A. 00000000

B. 00001111

C. 11110000

D. 11111111

*MSB is on the left, rightmost bit is 0


Or Truth Table

	<i>0</i>	<i>1</i>
<i>0</i>	0	1
<i>1</i>	1	1

OR Operations

- Useful to include bits in a word
 - Set some bits to 1, leave others unchanged

or \$t0, \$t1, \$t2



\$t2	0000 0000 0000 0000 0000 <u>1101 1100 0000</u>
\$t1	0000 0000 0000 0000 0011 <u>1100 0000 0000</u>
\$t0	0000 0000 0000 0000 0011 <u>1101 1100 0000</u>

OR Identities

- $x \mid 0 =$

~~X~~

- $x \mid 1 =$

~~1~~

01101001 | 11000111

A. 00010000

B. 01000001

C. 10101110

D. 11101111

01101001
11000111

11101111

Nor Truth Table

	<i>0</i>	<i>1</i>
<i>0</i>	1	0
<i>1</i>	0	0

NOR Operations

- MIPS has NOR 3-operand instruction
 - $a \text{ NOR } b == \text{NOT} (a \text{ OR } b)$

```
nor $t0, $t1, $t2
```

\$t2	0000 0000 0000 0000 0000 1101 1100 0000
------	---

\$t1	0000 0000 0000 0000 0011 1100 0000 0000
------	---

\$t0	1111 1111 1111 1111 1100 0010 0011 1111
------	---

01101001 NOR 11000111

A. 00010000

B. 01000001

C. 10101110

D. 11101111

NOT operations

- Inverts all the bits in a word
 - Change 0 to 1, and 1 to 0

MIPs does not need a NOT operation because we
can use _____ for
NOT \$t1, \$t2

- A. NOR \$t1, \$t2, \$zero
- B. NOR \$t1, \$t2, \$t3, where all bits in \$t3 are set to 1
- C. NORI \$t1, \$t2, 111111111111111111, where NORI is Nor Immediate
- D. It does require a NOT operation
- E. None of the above are correct

XOR Truth Table

	<i>0</i>	<i>1</i>
<i>0</i>	0	1
<i>1</i>	1	0

XOR Operations

- Exclusive OR
 - Set bits to one only if they are not the same

```
xor $t0, $t1, $t2
```

\$t2	0000 0000 0000 0000 0000 1101 1100 0000
------	---

\$t1	0000 0000 0000 0000 0011 1100 0000 0000
------	---

\$t0	0000 0000 0000 0000 0011 0001 1100 0000
------	---

$$01101001 \wedge 11000111$$

A. 00010000

B. 01000001

C. 10101110

D. 11101111

XOR Identities

- $x \wedge 0 = x$
- $x \wedge 1 = \text{not } x$

10 & 7

- A. 0
- B. 2
- C. 7
- D. 10
- E. None of the above

Set bit 4 in byte x to 1, leaving the rest of the bits unchanged.

A. $x = x \text{ AND } 00010000$

B. $x = x \text{ AND } 11101111$

C. $x = x \text{ OR } 00010000$

D. $x = x \text{ NOR } 11101111$

Invert bits 2-0 of x

A. $x = x \text{ AND } 00000111$

B. $x = x \text{ OR } 00000111$

C. $x = x \text{ NOR } 00000111$

D. $x = x \text{ XOR } 00000111$

Find the 1's complement of x

A. $x \text{ XOR } 00000000$

B. $x \text{ XOR } 11111111$

A. $x \text{ XOR } 11111110$

B. $x \text{ OR } 11111111$

Reading

- Next lecture: Branching instructions
 - Read Section 2.8
- Problem Set 3 due Friday
- Lab 2 due on Sunday