CS2100 Tutorial 2

C and MIPS

(Any question before class just come forward; class starts on :05)

Slides by Theodore, adapted from Prof. Aaron's slides

Overview

Q1) C bitwise operations

Q2) MIPS bitwise operations

Q3) MIPS Arithmetic

Q4) MIPS Tracing

Q1: Bitwise Operations

Couple of bitwise operations that you should know:

- bitwise OR |
- bitwise AND &
- bitwise XOR ^
- bitwise NOT / 1s complement ~
- left-shift <<
- right-shift >>

In-class:
using a = 5 `00000101`
and b = 22 `00010110`

Note that left-shift also effectively multiplies by 2; the opposite for right-shift. Usually more efficient than normal multiplication!

Bonus: check out the difference between logical and arithmetic shift (not in syllabus)

Q1: Ternary Operator

The conditional operator ? : is a shorthand to if-else

condition ? true-part : false-part

Equivalent examples:

```
if (a < 5) {
    b = 10;
} else {
    b = 15;
}
```

```
b = (a < 5) ? 10 : 15;
```

Q2: MIPS Bitwise Operations

a. Set bits 2, 8, 9, 14, and 16 of b to 1. Leave all other bits unchanged.

Note: a register contains a 32-bit value, so b could look like this:

`b = 00110000011100000110110100100001`

and we want to change it to

`b = 00110000011100010110111100100101`

(Note: MSB is bit 31, on left, “upper” bits)

Q2: MIPS Bitwise Operations

Step-by-step thought process:

Q: What do we need to change a bit to 1, regardless of the previous value?

A: Easiest way is to use OR; and OR the bit with 1

A	B	A OR B
1	1	1
0	1	1
1	0	1
0	0	0

Intermediate solution:

```
ori  $s1, $s1, 0b00000000000000010100001100000100
```

Problem:

(Hint: see the immediate value / last operand)

Solution:

Load it up to a 32-bit register

Q2: MIPS Bitwise Operations

Loading `0000 0000 0000 0001 0100 0011 0000 0100` into `$t0:`

Since we can only use 16 bits of imm value, we load the upper 16 bits with `lui`

```
lui $t0, 0b1
```

(Note: `lui` also clears the bottom 16 bits)

Then we load the bottom 16 bits using ori

```
ori $t0, $t0, 0b0100 0011 0000 0100
```

Finally we can call or between $s1 (b) and $t0

```
or  $s1, $s1, $t0
```

Q2: MIPS Bitwise Operations

b) Copy over bits 1, 3 and 7 of b into a, without changing any other bits of a.

Example, if initially:

```
a = ... 1 0 0 0 1 1 1 1 0
b = ... 1 1 1 1 1 0 0 1 0
```

We want the final result to be:

```
a = ... 1 1 0 0 1 0 1 1 0
```

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

Q2: MIPS Bitwise Operations

Step-by-step thought process:

1. Get bits 1, 3, 7 from b (say to `$t0`)
2. Clear bits 1, 3, 7 from a
3. The result is `$t0 OR $s0`

Note: there are other ways to solve this question.

Exam question usually asks for the shortest possible answer, so any other 5-instruction correct answer gets full marks.

Q2: MIPS Bitwise Operations

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

1. Get bits 1, 3, 7 from b/`$s1` (say to `$t0`)

```
$s1  = ... 1 1 1 1 1 0 0 1 0
mask = ...
$t0  = ... 0 1 0 0 0 0 0 1 0
```

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

Q2: MIPS Bitwise Operations

1. Get bits 1, 3, 7 from b/`$s1` (say to `$t0`)

```
$s1  = ... 1 1 1 1 1 0 0 1 0
mask = ... 0 1 0 0 0 1 0 1 0        AND
$t0  = ... 0 1 0 0 0 0 0 1 0
```

Answer:

```
andi $t0, $s1, 0b0000 0000 1000 1010
```

Q2: MIPS Bitwise Operations

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

2. Remove bits 1, 3, 7 from a/`$s0`

```
$s1  = ... 1 0 0 0 1 1 1 1 0
mask = ...
$t1  = ... 1 0 0 0 1 0 1 1 0
```

Q2: MIPS Bitwise Operations

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

2. Remove bits 1, 3, 7 from a/`$s0`

```
$s0        = ... 1 0 0 0 1 1 1 1 0
mask/$t1   = ... 1 0 1 1 1 0 1 0 1  AND
$s0        = ... 1 0 0 0 1 0 1 0 0
```

Note that the mask has 1s all the way to the MSB – we get the same problem as Q2a; solution is the same (lui followed by ori)

Answer:

```
lui $t1, 0b1111111111111111
ori $t1, $t1, 0b1111111101110101
and $s0, $s0, $t1
```

Q2: MIPS Bitwise Operations

A	B	A AND B
0/1	1	0/1 (unchanged)
0/1	0	0

A	B	A OR B
0/1	0	0/1 (unchanged)
0/1	1	1

3. The result is `$t0 OR $s0`

Final answer:

```
andi $t0, $s1, 0b0000 0000 1000 1010
lui  $t1, 0b1111111111111111
ori  $t1, $t1, 0b1111111101110101
and  $s0, $s0, $t1
or   $s0, $s0, $t0
```

Q2: MIPS Bitwise Operations

c) Make bits 2, 4, and 8 of c the inverse of bits 1, 3, and 7 of b

Thought process:

- If we can isolate bits 1, 3, and 7 of b to $t0
  and we move that into bits 2, 4, and 8,
  and we can invert it,
  this reduces to question 2b)

Note: "reduces to" means that we have converted a (usually harder) problem into another (usually solved) problem.
You'll see a lot of these in theoretical CS courses.

Q2: MIPS Bitwise Operations

c) Make bits 2, 4, and 8 of c the inverse of bits 1, 3, and 7 of b

Step-by-step thought process:

1. Isolate bits 1, 3, and 7 of b/$s1 to $t0
2. Flip all bits
3. Shift one position left
4. Clear bits 2, 4, and 8 of c/$s2
5. Answer is $t0 OR $s2

> Note:
> Again, multiple valid answers. This slide does not give the same instructions as in the answer sheet as you will see later

Q2: MIPS Bitwise Operations

c) Make bits 2, 4, and 8 of c the inverse of bits 1, 3, and 7 of b

Prof's process:

1. Flip bits 1, 3, and 7
2. Get bits 1, 3, and 7
3. Shift one position left
4. Clear bits 2, 4, and 8 of c/$s2
5. OR to get the answer

```
xori  $t0, $s1, 0b10001010
andi $t0, $t0, 0b10001010
sll     $t0, $t0, 1
lui     $t1, $t1, 0b1111111111111111
ori     $t1, $t1 0b1111111011101011
and   $s2, $s2, $t1
or      $s2, $s2, $t0
```

Q3. MIPS Arithmetic

Important:
Note the "as few instructions as possible"

a → \$s0 b → \$s2
c → \$s3 d → \$s4

a) c = a + b

```
add $s2, $s0, $s1
```

Q3. MIPS Arithmetic

Important:
Note the "as few instructions as possible"

a → \$s0 b → \$s2
c → \$s3 d → \$s4

b) d = a + b – c

```
add $s3, $s0, $s1
sub $s3, $s3, $s2
```

To think about:
Why do we not need to zero \$s3?

To think about:
What operation(s) zeroes a register?

Q3. MIPS Arithmetic

Important:
Note the “as few instructions as possible”

a → \$s0   b → \$s2
c → \$s3   d → \$s4

c) c = 2b + a – 2

```
add  $s2, $s1, $s1      (alt: sll $s2, $s1, 1)
addi $t0, $s0, -2
add $s2, $s2, $t0
```

Bonus:
Why does this not work?

```
srl  $s2, $s0, 1       # c = a/2
add  $s2, $s2, $s1     # c = a/2 + b
addi $s2, $s2, -1      # c = a/2 + b -1
sll  $s2, $1, 1        # c = (a/2 + b − 1) * 2
```

Q3. MIPS Arithmetic

Important:
Note the “as few instructions as possible”

a → $s0 b → $s2
c → $s3 d → $s4

d) 6a + 3 (b - 2c) = 3(2(a - c) + b)

```
sub $t0, $s0, $s2   # t0 = a − c
sll $t0, $t0, 1     # t0 = 2(a − c)
add $t0, $t0, $s1   # t0 = 2(a - c) + b
sll $t1, $t0, 2     # t1 = 4(2(a - c) + b)
sub $s3, $t1, $t0   # d = 3(2(a − c) + b)
```

Break

Attendance taking

A	B	A XOR B
0/1	0	0/1 (unchanged)
0/1	1	1/0 (flipped)

Q4. MIPS Tracing

In-class tracing

```
    add  $t0, $s0, $zero
    lui  $t1, 0x8000
lp: beq  $t0, $zero, e
    andi $t2, $t0, 1
    beq  $t2, $zero, s
    xor  $s0, $s0, $t1
s:  srl  $t0, $t0, 1
    j    lp
e:
```

End of Tutorial 2

- Slides uploaded on github.com/theodoreleebrant/TA-2425S1
- Email: theo@comp.nus.edu.sg
- Anonymous feedback:
  bit.ly/feedback-theodore
  (or scan on the right)

(Also reminder for me to take attendance)