CENG311 Computer Architecture

Instructions: Language of the Computer

IZTECH, Fall 2023

19 October 2023

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`

Shift Operations

Shift left logical

- Shift left and fill with 0 bits
- ***sll*** by i bits multiplies by 2^i

Shift right logical

- Shift right and fill with 0 bits
- ***srl*** by i bits divides by 2^i (unsigned only)

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

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 1101 1100 0000
$t1	0000 0000 0000 0000 0011 1100 0000 0000
$t0	0000 0000 0000 0000 0011 1101 1100 0000

NOT Operations

Useful to invert bits in a word

- Change 0 to 1, and 1 to 0

MIPS has NOR (NOT OR) instruction instead of NOT

- a NOR b == NOT (a OR b)
- A NOR 0 = NOT (A OR 0) = NOT A

```
nor $t0, $t1, $zero
```

```
$t1  0000 0000 0000 0000 0011 1100 0000 0000

$t0  1111 1111 1111 1111 1100 0011 1111 1111
```

Instructions for Making Decisions

Branch to a labeled instruction if a condition is true

Otherwise, continue sequentially

beq rs, rt, L1

if (rs == rt) branch to instruction labeled L1;

bne rs, rt, L1

if (rs != rt) branch to instruction labeled L1;

j L1

unconditional jump to instruction labeled L1

Compiling If Statements

C code:

```
    if (i==j) f = g+h;
    else f = g-h;
```

(f, g, h, i, j in $s0, $s1, $s2, $s3, $s4)

Compiling If Statements

C code:

```
    if (i==j) f = g+h;
    else f = g-h;
```

Compiled MIPS code

```
    bne $s3, $s4, Else
    add $s0, $s1, $s2
    j Exit
Else: sub $s0, $s1, $s2
Exit: …
```

Compiling Loop Statements

C code:

```
while (save[i] == k)
        i += 1;
```

(i in $s3, k in $s5, base address of save in $s6)

Compiling Loop Statements

C code:

```
while (save[i] == k)
        i += 1;
```

Compiled MIPS code

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

More Conditional Operations

Set result to 1 if a condition is true

Otherwise, set to 0

slt rd, rs, rt

if (rs < rt) rd = 1; else rd = 0;

slti rt, rs, constant

if (rs < constant) rt = 1; else rt = 0;

Use in combination with beq, bne

```
slt $t0, $s1, $s2  # if ($s1 < $s2)
bne $t0, $zero, L  #   branch to L
```

Signed vs. Unsigned

Signed comparison: slt, slti

Unsigned comparison: sltu, sltui

Example

$s0 = 1111 1111 1111 1111 1111 1111 1111 1111

$s1 = 0000 0000 0000 0000 0000 0000 0000 0001

slt $t0, $s0, $s1 # signed

–1 < +1 => $t0 = 1

sltu $t0, $s0, $s1 # unsigned

+4,294,967,295 > +1 => $t0 = 0

References

Chapter 2.6

Chapter 2.7

(Computer Organization and Design: The Hardware/Software Interface by Hennessy/Patterson, 5th edition)