Logical Instructions and Conditional Execution

Tom Kelliher, CS 240
Sept. 7, 2005

1 Administrivia

Announcements

Assignment

Install a Linux distro on your PC.

From Last Time

Operands and instruction formats.

Outline

1. Logical instructions.

2. Branch and jump instructions.

3. Compiling HLL control structures.

4. Class teamwork assignment.
2 Coming Up

Intro to Linux.

2 Last Time

Operations, operands, and instruction formats.

3 Logical Operations

The basics:

1. NOT: complement the bits of the operand, bit by bit. (~)

2. AND: AND the bits of two operands, bit by bit. (&, not & &).

3. OR: OR the bits of two operands, bit by bit. (|, not | |).

4. Shift: Move the bits of the operand to the left or right a given “distance.”

Details:

1. MIPS has no NOT operation, but it does have NOR: ~(a | b).

   How do you use NOR to get NOT?

   \[ \sim 1101 = 0010 \]

2. 1101 & 1001 = 1001

   and \$s2, \$t0 \$t1

3. 1001 | 0100 = 1101
4. Shifts are “similar” to multiplication and division.

\[ 11001101 \ll 3 = 01101000 \]

Usage example: shift and mask operations in finding a character in a word.

# 4 Branch and Jump Instructions

1. I-format instructions.

2. The idea behind a branch or jump:

```
...  
      
      
br Label  
      
      
      
      Skip over intermediate instructions.
...
```

Label:  
```
...  
...  
...  
```

3. Branch forward or backward \(2^{15}\) words.

The complete set, all synthesized from `beq`, `bne`, and `slt`.

Branch instructions use a signed 16-bit offset field; hence they can jump \(2^{15} - 1\) instructions (not bytes) forward or \(2^{15}\) instructions backwards. The `jump` instruction contains a 26 bit address field (the third instruction format).

b label  
Unconditionally branch to the instruction at the label.

beq Rsrl, Src2, label  
Conditionally branch to the instruction at the label if the contents of register `Rsrl` equals `Src2`.  

---

*b Branch instruction*
beqz Rsra, label \quad \textit{Branch on Equal Zero}
Conditionally branch to the instruction at the label if the contents of Rsra equals 0.

bge Rsra, Src2, label \quad \textit{Branch on Greater Than Equal}
bgeu Rsra, Src2, label \quad \textit{Branch on GTE Unsigned}
Conditionally branch to the instruction at the label if the contents of register Rsra1 are greater than or equal to Src2.

bgez Rsra, label \quad \textit{Branch on Greater Than Equal Zero}
Conditionally branch to the instruction at the label if the contents of Rsra are greater than or equal to 0.

bgt Rsra, Src2, label \quad \textit{Branch on Greater Than}
bgtu Rsra, Src2, label \quad \textit{Branch on Greater Than Unsigned}
Conditionally branch to the instruction at the label if the contents of register Rsra1 are greater than Src2.

bgtz Rsra, label \quad \textit{Branch on Greater Than Zero}
Conditionally branch to the instruction at the label if the contents of Rsra are greater than 0.

ble Rsra, Src2, label \quad \textit{Branch on Less Than Equal}
bleu Rsra, Src2, label \quad \textit{Branch on LTE Unsigned}
Conditionally branch to the instruction at the label if the contents of register Rsra1 are less than or equal to Src2.

blez Rsra, label \quad \textit{Branch on Less Than Equal Zero}
Conditionally branch to the instruction at the label if the contents of Rsra are less than or equal to 0.

blt Rsra, Src2, label \quad \textit{Branch on Less Than}
bltu Rsra, Src2, label \quad \textit{Branch on Less Than Unsigned}
Conditionally branch to the instruction at the label if the contents of register Rsra1 are less than Src2.

bltz Rsra, label \quad \textit{Branch on Less Than Zero}
Conditionally branch to the instruction at the label if the contents of Rsra are less than 0.
bne Rsrl, Src2, label

Conditionally branch to the instruction at the label if the contents of register Rsrl are not equal to Src2.

bnez Rsrl, label

Conditionally branch to the instruction at the label if the contents of Rsrl are not equal to 0.

jal label

Unconditionally jump to the instruction at the label.

jal Rsrl

Jump and Link

Unconditionally jump to the instruction at the label or whose address is in register Rsrl.

Save the address of the next instruction in register 31.

jr Rsrl

Jump Register

Unconditionally jump to the instruction whose address is in register Rsrl.

5 Compiling HLL Control Structures

Write MIPS code fragments corresponding to the following:

1. Compiling an if:
if (i < 12)
    ++i;
else
    --j;

2. Compiling a loop:

i = 1;
j = 0;
while (i < 200)
{
    j += i;
    i *= i;
}
6 Class Teamwork Assignment

The class, working as a team, is to e-mail the solution to the following problems to me (I'll collect the solutions and e-mail them as one to the class.) Let me know who participated in the solution of what problem(s).

1. \( j = 0; \)
   \( \text{for } (i = 0; i < 10; ++i) \)
   \( j += i; \)

2. \( j = 0; \)
   \( \text{for } (i = 0; i < 10; ++i) \)
   \( \text{if } (i > 5) \)
   \( j += i; \)

3. \( \text{while } (i > 0 && i < 10) \)
   \( ++i; \)

4. \( \text{if } (i < 12 && j > 3 || k != 0) \)
   \( ++i; \)
   \( \text{else if } (i == 33) \)
   \( --j; \)
   \( \text{else} \)
   \( k += 2; \)

5. (3.9 from the text) The naive way of compiling

   \( \text{while } (\text{save}[i] == k) \)
   \( i += k; \)

   requires execution of both a conditional branch and an unconditional jump each time through the loop. Produce the naive code.

   Optimize the naive code so that only a conditional branch is executed each time through the loop.

6. (3.24 from the text, a variation) Write a code segment which takes two “parameters:”

   (a) An ASCII character in \$a0.

   (b) A pointer to a NULL-terminated string in \$a1.
and “returns” a count of the number of occurrences of the character in the string in $v0.$