# CSE 410
# Computer Systems

Hal Perkins

Spring 2010

Lecture 5 – Control Flow: Decisions & Loops

# Reading and References

- Computer Organization and Design
  - Section 2.6, Logical Operations
  - Section 2.7, Instructions for Making Decisions
  - Section B.9, SPIM
  - Section B.10 through page B-50, MIPS Assembly Language

# Control flow in high-level languages

- The instructions in a program usually execute one after another, but it's often necessary to alter the normal control flow.
- Conditional statements execute only if some test expression is true.

```
// Find the absolute value of a0
v0 = a0;
if (v0 < 0)
    v0 = -v0;               // This might not be executed
v1 = v0 + v0;
```

- Loops cause some statements to be executed many times.

```
// Sum the elements of a five-element array a0
v0 = 0;
t0 = 0;
while (t0 < 5) {
    v0 = v0 + a0[t0];       // These statements will
    t0++;                   // be executed five times
}
```

# Control-flow graphs

```
// Find the absolute value of a0
v0 = a0;
if (v0 < 0)
    v0 = -v0;
v1 = v0 + v0;
```

```
// Sum the elements of a0
v0 = 0;
t0 = 0;
while (t0 < 5) {
    v0 = v0 + a0[t0];
    t0++;
}
```

# MIPS control instructions

- MIPS's control-flow instructions:

  j // for unconditional jumps

  bne and beq // for conditional branches

  slt and slti // set if less than (reg. & immediate)

- Usage:

```
j  there                  // next instruction at label "there"
beq  $t0, $t1, xyz   // if $t0==$t1, next instr. at "xyz"
slt     $t3, $a1, $s0 // if $a1<$s0, $t3=1 else $t3=0
```

# Pseudo-branches

- The MIPS processor only supports two branch instructions, beq and bne, but to simplify your life the assembler provides the following other branches:

```
blt $t0, $t1, L1    // Branch if $t0 < $t1
ble $t0, $t1, L2    // Branch if $t0 <= $t1
bgt $t0, $t1, L3    // Branch if $t0 > $t1
bge $t0, $t1, L4    // Branch if $t0 >= $t1
```

- There are also immediate versions of these branches, where the second source is a constant instead of a register

# Implementing pseudo-branches

- Most pseudo-branches are implemented using slt. For example, a branch-if-less-than instruction blt $a0, $a1, Label is translated into the following.

```
slt     $at, $a0, $a1      // $at = 1 if $a0 < $a1
bne     $at, $0, Label     // Branch if $at != 0
```

- This supports immediate branches, which are also pseudo-instructions. For example, blti $a0, 5, Label is translated into two instructions.

```
slti    $at, $a0, 5        // $at = 1 if $a0 < 5
bne     $at, $0, Label     // Branch if $a0 < 5
```

- All of the pseudo-branches need a register to save the result of slt, even though it's not needed afterwards.
  - MIPS assemblers use register $1, or $at, for temporary storage.
  - You should be careful in using $at in your own programs, as it may be overwritten by assembler-generated code.

# Translating an if-then statement

- We can use branch instructions to translate if-then statements into MIPS assembly code.

```
v0 = a0;
if (v0 < 0)
    v0 = -v0;
v1 = v0 + v0;
```

```
    move $v0 $a0
    bge  $v0, $0, Label
    sub  $v0, 0, $v0
Label:  add $v1, $v0, $v0
```

- Sometimes it's easier to *invert* the original condition.
  - In this case, we changed "continue if v0 < 0" to "skip if v0 >= 0".
  - This saves a few instructions in the resulting assembly code.

# Translating an if-then-else statement

- If there is an else clause, it is the target of the conditional branch
  - And the then clause needs a jump over the else clause

```
// increase the magnitude of v0 by one
if (v0 < 0)
    v0 --;

else
    v0 ++;
v1 = v0;
```

```
        bge  $v0, $0, E
        sub  $v0, $v0, 1
        j    L

E:      add  $v0, $v0, 1
L:      move $v1, $v0
```

  - Drawing the control-flow graph can help you out.

# Loops

- What does this code do?

```
label: sub   $a0,$a0,1
       bne   $a0,$zero,label
```

- Another example: for (i = 0; I < 4; i++) *stuff*

```
   add   $t0,$0,$0        # i = 0
loop:    //  stuff  goes here
   addi  $t0,$t0,1        # i++
   slt   $t1,$t0,4        # $t1 = i < 4
   bne   $t1,$zero,loop   # repeat if i<4
```

# Example: for (i=0; i<10; i++) s[i] = i;

```
# assume: $s0=addr(s), and let $t1=i
  move    $t1,$zero       # i = 0
loop:
  sll     $t0,$t1,2       # t0 = i*4
  addu    $t0,$s0,$t0     # t0 = addr(s[i])
  sw      $t1,0($t0)      # s[i] = i
  addu    $t1,$t1,1       # i++
  slti    $t0,$t1,10      # if (i<10) $t0=1
  bnez    $t0,loop        # loop if (i<10)
```

# Example: Count Characters in String

- Assume: $a0 points to a string of ASCII characters with 0x00 indicating the end
- Set $v0 = # of characters in string (exclude 0x00)

```
   li    $v0, 0            # length = 0
loop:
   lb    $t0, 0($a0)       # load char
   beq   $t0, $zero, done  # done if == 0x00
   addi  $v0, $v0, 1       # length++
   addi  $a0, $a0, 1       # next char
   j     loop              # repeat
done:  # $v0 = string length here
```