

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
}
```

MIPS control instructions

- ❑ In this lecture, we introduced some of MIPS's control-flow instructions

`j immediate`

// for unconditional jumps

`bne` and `beq $r1, $r2, label`

// for conditional branches

`slt` and `slti $r1, $r2, $r3`

// set if less than (w/ and w/o an immediate)

- ❑ And how to implement loops

- ❑ Today, we'll talk about

- MIPS's pseudo branches
- if/else
- case/switch

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
```

- ❑ Later this semester we'll see how supporting just **beq** and **bne** simplifies the processor design.

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.

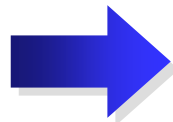
```
slti  $at, $a0, 5      // $at = 1 if $a0 < 5
bne   $at, $0, Label   // Branch if $a0 < 5
```

- ❑ 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;
```

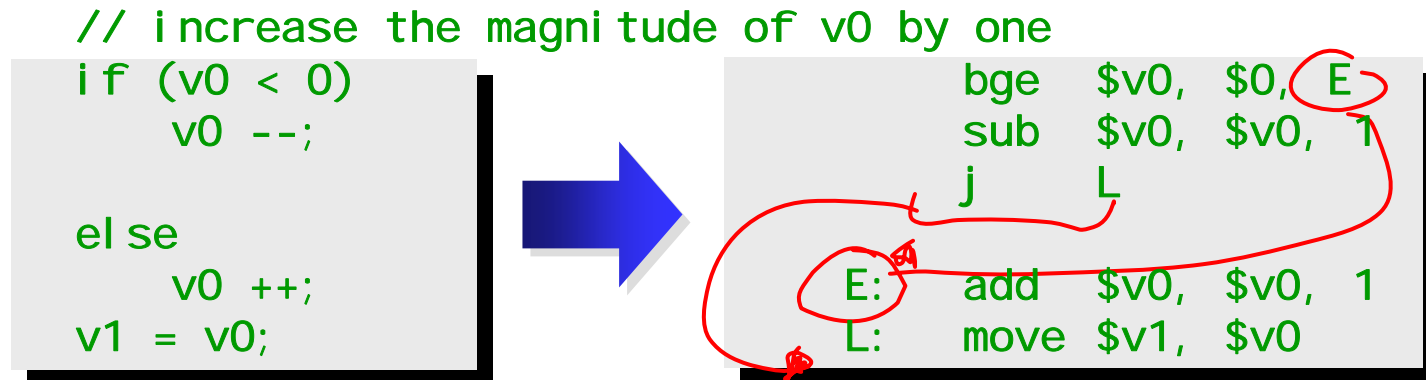


```
lw $t0, 0($a0)  
bge $t0, $zero, label  
sub $t0, $zero, $t0  
label: add $t1, $t0, $t0
```

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

Translating an if-then-else statements

- 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



- Dealing with else-if code is similar, but the target of the first branch will be another if statement.
 - Drawing the control-flow graph can help you out.