More ways to compare, <, >

beq, bne only test for equality or inequality, respectively.
How about $s0 < $s1?

The slt (set on less than) instruction is for this purpose

```
slt   $t0, $s0, $s1        # $t0 = ($s0 < $s1)
bne   $t0, $zero, L1       # if ($t0 != 0) goto L1
```

What the slt instruction does:

```
if ($s0 < $s1)
        $t0 = 1;
else
        $t0 = 0;
```

Using sll (shift-left logical) to multiply by a power of 2

Above, we used

```
sll   $t0, $s3, 2
```

This shifts the contents of $s3 left by 2 bits and places the result in $t0. General form:

```
sll   [dest GPR], [src GPR], [constant]
```

Example: Suppose $s0 contains [24 zeros]_0000_0110$_2$ (6_{10})

The instruction: `sll $s1, $s0, 5`
After execution:
$s0 unchanged
$s1 [24 zeros]_1100_0000$_2$ (192_{10})

⟶ In decimal, $6 \times 2^5 = 192$

Notes:
- Bits disappear off the left end
- zeros shifted in from the right

MIPS pseudoinstructions

pseudo-instructions → line of assembly language text available as a convenience to the programmer

- not real MIPS instructions
- generates one or more real instructions automatically

Example: blt (branch on less than)
- produces an slt and a bne instruction

Warning: In ENCM 369, only real MIPS instructions are usually allowed

Exception: la (load address) — as in Lab #2
Very useful!

Procedures in MIPS assembly language

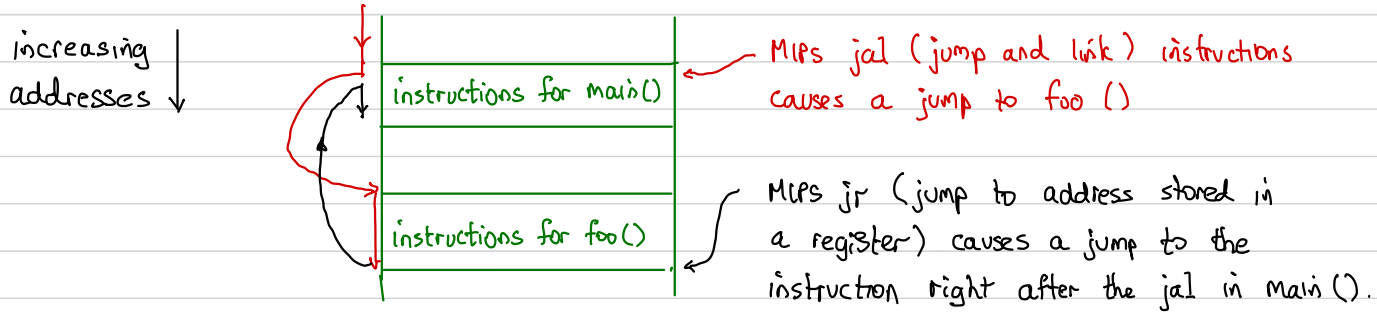
Very important - handles C-language functions

What's involved:

- jumping into and returning from procedures with jal, jr
- argument passing and return values
- handling register conflicts
- managing the stack to help with
  - register conflicts
  - allocation of local variables

Instruction flow

Suppose we have main() and foo(), where main() calls foo().



The return-address register : $ra (GPR 31)

From above,

main :

instructions

jal foo ←

add $s0, $v0, $zero

Call
- put address of add instruction into $ra
- put address of foo into the PC (program counter)

instructions

foo :

instructions

jr $ra ←

Return
- Copy address in $ra into the PC.

Summary of jump instructions : j, jal, jr

j label → goto label (no other actions)

jal label → goto label, but remembers address of the next instruction (where to come back) in $ra.

jr GPR → uses contents of GPR to decide where to jump (like jr $ra)

Important terms : caller and callee

Above program involves two procedures :

e.g., main (caller)
foo (callee)