Outline of Lecture

1. **Instructions for Making Decisions**

Conditional Instructions

- In order to write meaningful programs, all computer languages must contain ***decision making*** statements (e.g., `if` statements).

- The MIPS assembly language contains two decision making instructions which are ***branch equal*** and ***branch not equal*** (they are also called ***conditional branches***):

  ➜ `beq register1, register2, L1:` This instruction means go to the statement labeled L1 if the value in `register1` ***equals*** the value in `register2`.

  ➜ `bne register1, register2, L1:` This instruction means go to the statement labeled L1 if the value in `register1` ***does not equal*** the value in `register2`.

Example

In the following C code segment, `f`, `g`, `h`, `i`, and `j` are variables:

```
          if (i == j) goto L1;
          f = g + h;
      L1: f = f - i;
```

Assuming that the 5 variables correspond to 5 registers `$16` through `$20`, *what is the compiled MIPS code?*

Answer

The compiled program is

```
beq $s3, $s4, L1        # goto L1 if i equals j

add $s0, $s1, $s28      # f=g+h (skipped if i=j)

L1: sub $s0, $s0, $s3   # f=f-i(always executed)
```

Since instructions are also stored in memory, like data, they have memory addresses. Thus, the label `L1` corresponds to the memory address of the sub instruction.

- MIPS also have an ***unconditional branch*** instruction termed `jump` and is abbreviated as `j` (e.g., `j Exit`).
- Decision instructions in MIPS can be used to choose between 2 alternatives or it can be used for iterating a computation in a ***loop***.

Example

Given the following loop is C:

```
Loop: g = g + A[i];
      i = i + j;
      if (i !=h) goto Loop;
```

Assume A is an array of 100 elements and starts at address `Astart`. The variables g, h, `i`, and `j` are associated to the registers $s1, $s2, $s3, and $s4. The base address of the array A is stored in $5.

What is the MIPS assembly code for the above C loop?

Answer

```
Loop: add $t1, $s3, $s3   # Temp reg $t1=2*i
      add $t1, $t1, $t1   # Temp reg $t1 = 4*i
      add $t1, $t1, $s5   # $t1=address of A[i]
      lw $t0, 0($t1)      # Temp reg $t0=A[i]
      add $s1, $s1, $t0   # g = g + A[i]
      add $s3, $s3, $s4   # i = i + j
      bne $s3, $s2, Loop  # goto Loop if i ≠ h
```

- In writing computer programs, it is often useful to see if a variable is ***less*** than the other.

- The MIPS assembly language has such an instruction called `set on less than` and abbreviated `slt`.

  ➜ `slt register1, register2, register3`: This instruction means that `register1` ***is set to 1*** if the value in `register2` is ***less*** than the value in `register3` (e.g., `slt $t0, $s3, $s4`).

With `beq`, `bne`, and `slt`, and using the fixed value `0` in `register $0`, we can accomplish all relative conditions (e.g., `branch less than`, `branch greater than`, etc.).

- The MIPS assembly language has also un unconditional jump instruction which is useful for `if-then-else` statements, which is `jump register` (`jr`):
  - ➜ `jr:` This instruction means that an uncondi-`tional jump` to the address specified by a register.

Further Reading

Chapter 3 and Appendix. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface*. Morgan Kaufman Publishers, 1998.**