Module 4

MIPS Assembly Language;
Control Structures

Module Four

- This week, we are going to talk about :

- Program process flow
- Program Control Structures
  - IF – THEN – ELSE
  - LOOPS
  - SUBROUTINES
- Related data storage methods

Process Flow

- Top - Down
- Sequential
  - Each instruction is processed one after the one before.
- Branch
  - Conditional - Decision based on value comparison

    Equal - Not Equal

    Negative - Zero - Positive

    IBM Fortran; special hardware
  - Always

    Jump

    GoTo

Conditional Branch

- Decision making instructions
  - alter the control flow,
  - select the "next" instruction to be executed

- **`Branch on equal compare - Beq`**

  If the values in the two registers are the same

- **`Branch on unequal compare - Bne`**

  If the values in the two registers are different

- Example: if (i==j) h = i + j;

```
                 bne $s0, $s1, Label
                 add $s3, $s0, $s1
          Label:       ....
```

The set-on-less-than Instruction

- New instruction:

```
slt $at, $s1, $s2
```

```
if  $s1 < $s2   then
   $at = 1
              else
   $at = 0
```

- Can now build general control structures
- The assembler needs a register to do this, **$at**

Branch less than

- MIPS conditional branch instructions:

```
bne   beq   slt
```

- Put these together to create all the other conditional branches.

```
blt $t4, $t5, Finish  # if $t4 < $t5; go to Finish

slt $at, $t4, $t5     # if $t4 < $t5; $at = 1
bne $at, $zero, Finish
```

- Branch less than or equal to:

```
ble $t2, $t3, More  # if $t2 <= $t3; go to More

slt $at, $t3, $t2     # if $t3 < $t2; $at = 1
beq $at, $zero, More
```

Branch greater than

- The branch greater than:

```
bgt $t6, $t7, Again  # if $t6 > $t7; go to Again

slt $at, $t7, $t6      # if $t7 < $t6; $at = 1
bne $at, $zero, Again
```

- Branch greater than or equal to:

```
bge $t2, $t3, More  # if $t2 => $t3; go to More

slt $at, $t2, $t3      # if $t2 < $t3; $at = 1
beq $at, $zero, Finish
```

Control Structures

- IF - THEN
- LOOPS
  - FOR
  - WHILE
  - UNTIL
- Important: Create with the fewest number of instructions

IF - THEN

- IF - THEN

```
if  (i =! j)    then    k = k + 2
```

- MIPS

```
    BEQ   $t0, $t1, next     #  equal;  skip
    ADDI  $s0, $s0, 2        #  not equal;  add
next:
```

IF - ELSE IF

- IF - ELSEIF - ELSE

```
if      ( i == A )  then  k = k + 2
elseif ( i == B )  then  k = m + 4
else     k = p
```

- MIPS

```
    BEQ  $t0, $t1, A        #   goto case A
    BEQ  $t0, $t2, B        #   goto case B
    ADD  $s0, $zero, $s4    #  else
    J   C                    #  goto finished
A: ADDI $s0, $s0, 2          #  case A
    J   C                    #  goto finished
B: ADDI  $s0, $s2, 4         #  case B
C:                           #  finished
```

Loops

- FOR - LOOP
  - Named from FORTRAN
  - Variation of UNTIL loop
  - Fixed number of iterations

```
FOR I = 1,5 DO 129
```

- WHILE - LOOP
  - Test at beginning

- UNTIL - LOOP
  - Test at end

WHILE LOOP

- Test for condition, process **while** condition remains **true**
- MIPS (**while** value $t0 is less than value $s5)

```
loop: BEQ  $t0,$s5, done   #  test condition
      ADD  $s0, $t4, $t5
      LW   $s4, 64 ($s1)
      ADD  $s2, $s2, $s4
      ADDI $t0, $t0,  1
      J        loop        #  repeat the loop
done:
```

UNTIL LOOP

- Process, test, continue **until** condition becomes **true**
- MIPS (**until** value $t0 equals value $s5)

```
loop: ADD  $s0, $t4, $t5
      LW   $s4, 64 ($s1)
      ADD  $s2, $s2, $s4
      ADDI $t0, $t0, 1
      BNE  $t0, $s5, loop  # test condition
done:
```

Summary

- MIPS Assembly Language and control structures
  - Process Flow
  - Set on less than
  - Control Structures

    IF – THEN – ELSE

    Loops

      WHILE
      UNTIL

Next: Subroutines