Lab 5: MIPS Branch Instructions and Flow Control

Objectives:

After completing this lab, you will get:

- Get familiar with MIPS Jump and Branch instructions.
- Learn how to translate high-level flow control constructs (if-then-else, for loop, while loop) to MIPS code.

1. MIPS Jump and Branch Instructions

Like all processors, MIPS has instructions for implementing unconditional and conditional jumps. The MIPS Jump and Branch instructions are shown in Table 4.1.

Instruction	Meaning	Format			
j label	jump to label	$op^6 = 2$	imm^{26}		
beq rs, rt, label	branch if (rs == rt)	$op^6 = 4$	rs^5	rt^5	imm^{16}
bne rs, rt, label	branch if (rs != rt)	$op^6 = 5$	rs^5	rt^5	imm^{16}
blez rs, label	branch if (rs<=0)	$op^6 = 6$	rs^5	0	imm^{16}
bgtz rs, label	branch if (rs > 0)	$op^6 = 7$	rs^5	0	imm^{16}
bltz rs, label	branch if (rs < 0)	$op^6 = 1$	rs^5	0	imm^{16}
bgez rs, label	branch if (rs>=0)	$op^6 = 1$	rs^5	1	imm^{16}

Table 1: MIPS Jump and Branch Instructions.

For unconditional jump, the instruction **j label** is used where label is the address of the target instruction as shown below:

```
j   label          # jump to label
. . .
label:
```

There are two MIPS conditional branch instructions that branch based on the condition whether two registers are equal or not as follows:

```
beq Rs, Rt, label    # branch to label if (Rs== Rt)
bne Rs, Rt, label    # branch to label if (Rs != Rt)
```

Four additional MIPS instructions are provided based on comparing the content of a register with **0** as follows:

```
bltz Rs, label          # branch to label if (Rs < 0)
bgtz Rs, label          # branch to label if (Rs > 0)
 blez Rs, label         # branch to label if (Rs <= 0)
bgez Rs, label          # branch to label if (Rs >= 0)
```

MIPS also provides four set on less than instructions as follows:

```
slt rd, rs, rt          # if (rs < rt) rd = 1 else rd = 0
sltu rd, rs, rt         # unsigned <
slti rt, rs, im16       # if (rs < im16) rt = 1 else rt = 0
sltiu rt, rs, im16      # unsigned <
```

For example, assume that **$s0 = 1** and **$s1 = -1 = 0xffffffff**, then the following two instructions produce different results as shown below:

```
slt     $t0, $s0, $s1       # results in $t0 = 0

sltu    $t0, $s0, $s1       # results in $t0 = 1
```

2. Translating High-Level Flow Control Constructs

We can translate any high-level flow construct into assembly language using the jump, branch and set-less-than instructions. For example, let us consider the following if statement:

if (a == b) c = d + e; else c = d – e;

Let us assume that variables **a**, **b**, **c**, **d**, **e** are stored in registers **$s0** thru **$s4** respectively. The following assembly code implements this IF statement:

```
bne  $s0, $s1, else
addu $s2, $s3, $s4
 j        next
else:
subu $s2, $s3, $s4
next:     . . .
```

Exercise1:

Complete, write and run the precedent program with different values of a, b, d and e. Determine the value of c ($s2) in each case.

```
.data
messg: .asciiz "The value of c is \n"
a: .word 10
b: .word 7
d: .word 20
e: .word 15
.text
     lw $s0 a
     lw $s1 b
     lw $s3 d
     lw $s4 e
     bne  $s0, $s1, else
     addu $s2, $s3, $s4
    j   next
else: subu $s2, $s3, $s4
next:     li    $v0, 4
          la    $a0, messg
          syscall
          move  $a0, $s2
          li    $v0, 1
         syscall
         li $v0,10
         syscall
```

Exercise2:

Given that: $t0 = 1 and $t1 = -1 = 0xffffffff. Write a program to determine the values of #$t2 and #$t3

slt $t2, $t0, $t1 #$t2 = …….

sltu $t3, $t0, $t1 # $t3 = ……

exercise2.asm

```
.data
messg: .asciiz "The values of regitres are : \n"
.text
     li $t0 , 1
     li $t1 , -1
     slt   $t2, $t0, $t1
     sltu  $t3, $t0,$t1

next:     li    $v0, 4
          la    $a0, messg
          syscall
          move  $a0, $t2
          li    $v0, 1
         syscall
         move   $a0, $t3
          li    $v0, 1
         syscall
         li $v0,10
         syscall
```

Exercise3:

Consider the following WHILE loop:

i = 0;
while ((i<n) {
X+=2;
i++;
}
Where A is an array of integers (4 bytes per element). Translate WHILE loop:

$s0 = i, $s1 = n and $s2 = X. Write and run a MIPS program to determine the final value of X.

Edit | Execute

exercise3.asm

```
1  .data
2  n: .word 10
3  messg: .asciiz " The final value of X is : \n"
4  .text
5       li $s0, 0                          # $s0 = i = 0
6       lw $sl, n                          # $s1 = n
7       li $s2, 0                            # $s2 = X = 0
8  loop:   beq     $s0, $sl, next            # (i == n)?
9          addi    $s3, $s3, 2               #X=X+2
10         addiu    $s0, $s0,1                # i++
11         j       loop                        # jump backwards to loop
12
13 next:     li    $v0, 4
14           la    $a0, messg
15           syscall
16           move  $a0, $s3
17           li    $v0, 1
18          syscall
19          li $v0,10
20          syscall
21
```

Exercise4:

Translate the IF statement to assembly language. Write *and run* a program with different signed values of $t1, $t2, $t3.

if (($t1 > $t2) || ($t2 > $t3)) {$t4 = 9;} else {s4=2;}