Lab# 3 LOOP & BRANCH INSTRUCTIONS

Instructor: I Putu Danu Raharja.

Objectives:
Learn to implement loops and conditional expressions in assembly language programs.

Method:

Translate an algorithm from pseudo-code into assembly language.

Preparation:

Read the chapter 2 of lecture textbook.

3.1 DEVELOP THE ALGORITHM IN PSEUDOCODE

Obviously most of you have been familiar to develop algorithms using Java construct such as the following:

```
if(condition){
        this block of code executed if condition is true
} else {
        this block of code executed if condition is false
}
```

The key to making MIPS assembly language programming easy is to initially develop the algorithm using a high-level pseudo-code notation with which we are already familiar. Then in the final phase we translate these high-level pseudo-code expressions into MIPS assembly language. In other words, in the final phase we are performing the similar function that a compiler performs, which is to translate high-level code into the equivalent assembly language.

3.2 CONDITIONAL AND UNCONDITIONAL BRANCH INSTRUCTIONS

Instructions	Description
bgez rs, L	**if** (rs ≥ 0) go to L;
bgtz rs, L	**if** (rs > 0) go to L;
blez rs, L	**if** (rs ≤ 0) go to L;
bltz rs, L	**if** (rs < 0) go to L;
bne rs, rt, L	**if** (rs != rt) go to L;
beq rs, rt, L	**if** (rs == rt) go to L;
slt rd, rs, rt	**if** (rs < rt) rd=1; **else** rd=0; rs and rt are *signed* integers.

Instructions		Description
sltu	rd, rs, rt	Same as **slt** except rs and rt are *unsigned* integers.
slti	rt, rs, immediate	**if** (rs < *signed* immediate) rd=1; **else** rd=0;
sltiu	rt, rs, immediate	**if** (rs < *unsigned* immediate) rd=1; **else** rd=0;
j	L	go to L

3.3 Examples

A. Example 1:

Write a MIPS assembly language program that calculates the sum of all positive integers less than or equal to N and displays the result in the monitor. Assume that N is stored in the register $t0.

Algorithm		Assembly Language		
	$t0 ← N;		li	$t0, N
	$t1 ← 1;		li	$t1, 1
	$a0 ← 0;		add	$a0, $zero, $zero
loop:	if ($t1 > $t0) go to print;	loop:	sltu	$t2, $t0, $t1
	$a0 ← $a0 + $t1;		bgtz	$t2, print
	$t1 ← $t1 + 1;		addu	$a0, $a0, $t1
	go to loop;		addi	$t1, $t1, 1
print:	display $a0;		j	loop
	exit;	print:		

B. Example 2:

Write a MIPS assembly language program that displays all the first N Fibonacci numbers.

Algorithm		Assembly Language
	$t0 ← N – 1;	
	$t1 ← 1;	
	$a0 ← 1;	
	display $a0;	
loop:	display $a0;	
	$t0 ← $t0 – 1;	
	if ($t0 == 0) stop;	
	$a0 ← $a0 + $t1;	
	$t1 ← $a0 – $t1;	
	go to loop;	
stop:		

3.4 LAB EXERCISES:

1. Write the complete code of example 1 and 2. Try running the program with both the run command and the step command.

2. What is the hexadecimal representation of the instruction **bgtz $t2, print**?

3. Write a complete MIPS program to display all ODD positive integers less than 1000.

4. Write a complete MIPS program to display the following pattern using *loops*.

   Run-time example:

   1

   2 3

   4 5 6