CS 21 Laboratory 4: Conditional and Unconditional Branches

University of the Philippines - Diliman
College of Engineering
Department of Computer Science

1 Conditional Execution

In Assembly Language, If-then-else statements are carried out using conditional and unconditional branches. The following activity will demonstrate an actual implementation.

Create an assembly language program file. In the data segment, create two strings, one containing the word "EQUAL" and the other "UNEQUAL". Label them "equal" and "unequal", respectively.

Using the load immediate instruction,load the value of 1 into t0 and t1.

Follow the **li** instructions with the instructions in Code Block 1.

Code Block 1 If-then-else in Assembly Language

```
beq $t0, $t1, then # branch to then label if equal
else: la $a0, unequal  # load the addr of equal into $a0.
li $v0, 4 # 4 is the print_string syscall.
syscall # do the syscall.
b after # unconditional branch to after
then: la $a0, equal  # load the addr of equal into $a0.
li $v0, 4 # 4 is the print_string syscall.
syscall # do the syscall.
after:
```

Run the program. What is the output? Why is the unconditional branch necessary? Play around with the values of t0 and t1.

2 Loops

Create another assembly language program. In the data segment, create a string "BANZAI" label it "banzai".

Get a POSITIVE integer input from the user via the console(you do remember how to do this, right?). Store the integer in register t3. Load the integer 0 into register t0.

Now append the instructions in Code Block 2.

How is the output related to the input? What would happen if the input is 0? What kind of loop is this?

Code Block 2 Looping in Assembly Language

```
start: la $a0, banzai  # load the addr of equal into $a0.
li $v0, 4 # 4 is the print_string syscall.
syscall # do the syscall.
addi $t0, $t0, 1 # add 1 to t0
bne $t0, $t3, start # branch to start if not equal
```

3 Machine Exercises

Work on the following machine exercises independently. Upon finishing one, call my attention for checking.

3.1 Multiplier

Create a program that takes two integer inputs from the user and then outputs the product of those two integers. Use of the **mult** instruction(and its derivatives) not allowed. Filename should be "cs21lab4a.asm". Don't forget the required header comments.

3.2 LCM Finder

Create a program that takes two integer inputs from the user and then outputs the Least Common Multiple(LCM) of those two integers. Use of the **mult** instruction(and its derivatives) not allowed. Filename should be "cs21lab4b.asm". Don't forget the required header comments.

3.3 Lowercase again

Redo the machine exercise in Laboratory Session 2 with the following new specifications:

1. instead of being restricted to 6 letter words, the user would now have the option of choosing how many letters the input would have. The user would now have to type the number of letters in the input first, before typing the actual input word.
2. there should only be **one** store instruction in your entire program

Filename should be "cs21lab4c.asm". Don't forget the required header comments.

4 Learning Checklist

In this session you should have learned...

- how to create and use conditional statements in Assembly Language
- how to create and use loops in Assembly Language