**BRANCHING OPERATIONS**

**IN ASSEMBLY**

**LANGUAGE**

**LAB # 0****2**

**Fall 2023**

**CSE-304L**

**Computer Organization & Architecture Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Bilal Habib**

Friday, October 13, 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**ASSESSMENT RUBRICS COA LABS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB REPORT ASSESSMENT** | | | | |
| **Criteria** | **Excellent** | **Average** | **Nill** | **Marks Obtained** |
| 1. **Objectives of Lab** | All objectives of lab are properly covered  [Marks 10] | Objectives of lab are partially covered  [Marks 5] | Objectives of lab are not shown  [Marks 0] |  |
| 1. **MIPS instructions with**   **Comments and proper indentations.** | All the instructions are well written with comments explaining the code and properly indented  [Marks 20] | Some instructions are missing are poorly commented code  [Marks 10] | The instructions are not properly written  [Marks 0] |  |
| 1. **Simulation run without error and warnings** | The code is running in the simulator without any error and warnings  [Marks 10] | The code is running but with some warnings or errors.  [Marks 5] | The code is written but not running due to errors  [Marks 0] |  |
| 1. **Procedure** | All the instructions are written with proper procedure  [Marks 20] | Some steps are missing  [Marks 10] | steps are totally missing  [Marks 0] |  |
| 1. **OUTPUT** | Proper output of the code written in assembly  [Marks 20] | Some of the outputs are missing  [Marks 10] | No or wrong output  [Marks 0] |  |
| 1. **Conclusion** | Conclusion about the lab is shown and written  [Marks 20] | Conclusion about the lab is partially shown  [Marks 10] | Conclusion about the lab is not shown[Marks0]  [Marks 0] |  |
| 1. **Cheating** |  |  | Any kind of cheating will lead to 0 Marks |  |
| Total Marks Obtained: \_\_\_\_\_\_\_\_\_\_  Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |

**Branching Operations**

Objectives:

* How to create a branch
* How to jump to other branches
* Create reusable branches

Tasks:

**Task 1**: Enter a number 5432 from user and then display the last digit in the console. (hint: use mfhi ).

**Code:**

.text

.globl main

main:

li $v0, 4 #print string

la $a0, prompt

syscall

li $v0, 5 # enter an int

syscall

move $t0, $v0

li $t1, 10

div $t0, $t1

mfhi $t0

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 10 # Exit the program

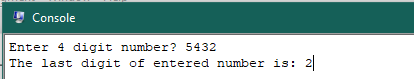
syscall

.data

prompt: .asciiz "Enter 4 digit number? "

result: .asciiz "The last digit of entered number is: "

**Output:**

****

**Task 2**: Check whether a number input by user is negative or equal to zero or greater then zero using branching ( Use bgt or ble ).

**Code:**

.text

.globl main

main:

li $v0, 4 #print string

la $a0, prompt

syscall

li $v0, 5 # enter an int

syscall

move $t0, $v0

li $t1, 0 # store a zero in t1

# if a number is positive

bgt $t0,$t1, isPositive

# if a number is zero

beq $t0, $t1, isZero

# if a number is negative

blt $t0, $t1, isNegative

# Some branchings

isPositive:

li $v0, 4 #print string

la $a0, positive

syscall

j end # Jump to end

isZero:

li $v0, 4 #print string

la $a0, zero

syscall

j end

isNegative:

li $v0, 4 #print string

la $a0, negative

syscall

j end

end:

li $v0, 10 # Exit the program

syscall

.data

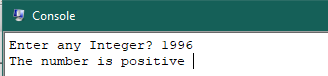
prompt: .asciiz "Enter any Integer? "

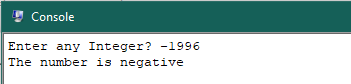
positive: .asciiz "The number is positive "

zero: .asciiz "The number is zero "

negative: .asciiz "The number is negative "

**Output:**

****

****

**Task 3**: Check using branch whether the number input by user are equal or not ( Use beq ).

**Code:**

.text

.globl main

main:

li $v0, 4 #print string

la $a0, prompt1

syscall

li $v0, 5 # enter an int

syscall

move $t0, $v0

li $v0, 4 #print string

la $a0, prompt2

syscall

li $v0, 5 # enter an int

syscall

move $t1, $v0

# if a both number are equal

beq $t0,$t1, isEqual

# if both number are not equal

bne $t0, $t1, isNotEqual

# ------------ Some branchings: -----------------------

isEqual:

li $v0, 4 #print string

la $a0, equal

syscall

j end # Jump to end

isNotEqual:

li $v0, 4 #print string

la $a0, notEqual

syscall

j end

end:

li $v0, 10 # Exit the program

syscall

.data

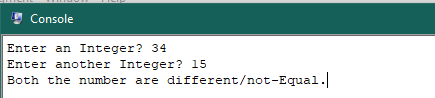
prompt1: .asciiz "Enter an Integer? "

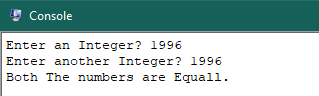
prompt2: .asciiz "Enter another Integer? "

equal: .asciiz "Both The numbers are Equall."

notEqual: .asciiz "Both the number are different/not-Equal."

**Output:**

****

****

**Task 4**: Write the assembly of the below C++ code.

Int age; …..

**Code:**

.text

.globl main

main:

li $v0, 4 #print string

la $a0, prompt

syscall

li $v0, 5 # enter an int

syscall

move $t0, $v0

li $t1, 18 # load 18 to a register

# if user >= 18

bge $t0, $t1, isEighteen

# if user is not 18

blt $t0, $t1, isNotEighteen

# -------------- Some branchings -----------------

isEighteen:

li $v0, 4 #print a string

la $a0, msg1

syscall

j end

isNotEighteen:

li $v0, 4 #print a string

la $a0, msg2

syscall

j end

end:

li $v0, 10 # Exit the program

syscall

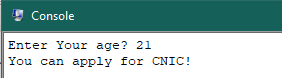
.data

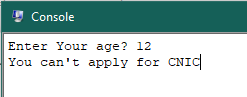
prompt: .asciiz "Enter Your age? "

msg1: .asciiz "You can apply for CNIC!"

msg2: .asciiz "You can't apply for CNIC"

**Output:**

****

****

**Task 5**: Write a program which take a limit from user and compute the sum of numbers from 0 to the limit ( Use bqe, add, addi, and J (jump)).

**Code:**

.text

.globl main

main:

li $v0, 4 #print string

la $a0, prompt

syscall

li $v0, 5 # enter an int num

syscall

move $t0, $v0

li $t1, 1 # store 1

li $t2, 2 # store 2

# if num > 0

bgt $t0, $zero, countSum

# else num <= 0

ble $t0, $zero, noSum

# -------------- Some branchings -----------------

countSum: # n(n+1)/2

add $t3, $t0, $t1 # n+1

mul $t4, $t0, $t3 # n(n+1)

div $t4, $t2 # n(n+1)/2

mflo $t5 # save qoutient

li $v0, 1

move $a0, $t5

syscall

j end

noSum:

li $v0, 4

la $a0, errorMessage

syscall

j end

end:

li $v0, 10 # Exit the program

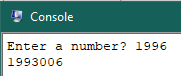
syscall

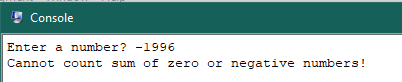
.data

prompt: .asciiz "Enter a number? "

errorMessage: .asciiz "Cannot count sum of zero or negative numbers!"

**Output:**

****

****

Reference:

To view my codes, please refer to my [GitHub Account.](https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/computer_organization_and_architechure_lab/lab_reports/)

Conclusion:

In this lab I have learnt how to create branches and use them in code. Mostly these branches are like If Else in high level languages. This lab give me insight of how to create a branch jump to another branch exit a program and much more.

The End.