### **BRANCHING OPERATIONS**

LAB # 02



# Fall 2023 CSE-304L Computer Organization and Architecture Lab

Submitted by: Ali Asghar

Registration No.: 21PWCSE2059

Class Section: C

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

Submitted to:

Dr. Bilal Habib

Date:

14th October 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]	
2. 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]	
3. 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]	
4. Procedure	All the instructions are written with proper procedure [Marks 20]	Some steps are missing [Marks 10]	steps are totally missing [Marks 0]	
5. 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]	
6. 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]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained:				
Instructor Signature:				

Task 1:

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

Code:

```
📑 task1.asm 🗵 📑 task2.asm 🗵 블 task3.asm 🗵 블 task4.asm 🗵 🛗 task5.asm 🗷 🔡 client.c 🗵 🛗 practice.asm 🗵
     .data
          msq1 : .asciiz "Enter the number: \n"
          msg2 : .asciiz "Last Digit is: \n"
 3
 4
      .text
 5
     .globl main
 6
     main:
 7
          #output msg1
 9
          li $v0,4
                            #load 4 into v0
10
          la $a0, msg1
                            #load address of msg1 to a0
11
          syscall
12
13
          #input value from user and save it in register t1
14
          li $v0,5
                            #load 5 into v0
15
          syscall
16
          move $t1, $v0  #move the entered value from v0 to t1 register
17
18
          #save 10 in t2 to divide t1 value by it
19
          li $t2, 10
20
21
          #performing division
          div $t1, $t2
```

```
23
24
          #move value stored in HI reg to t3
25
         mfhi $t3
26
27
          #output msg2
         li $v0,4
28
29
         la $a0, msg2
30
          syscall
31
          #output last digit
32
33
         li $v0,1
         move $a0, $t3
34
35
         syscall
36
37
          #exit the process
         li $v0, 10
39
          syscall
```

```
Console

Enter the number: 5432
Last Digit is: 2
```

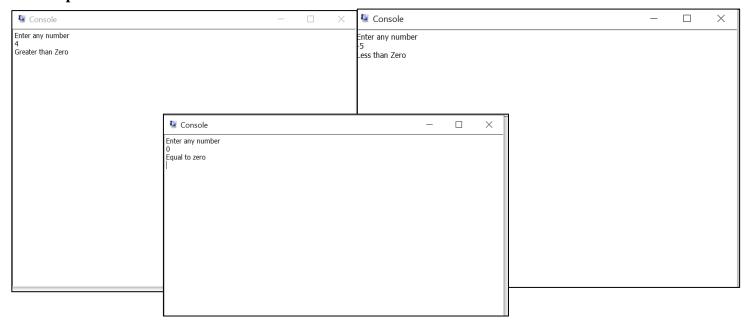
### Task 2:

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

```
ask1.asm 🗵 🔚 task2.asm 🗵 🔡 task3.asm 🗵 🔡 task4.asm 🗵 🔡 task5.asm 🗵 🔡 client.c 🗵 🔡 practice.asm 🗵
     .data
         input msg : .asciiz "Enter any number \n"
 3
         msg1 : .asciiz "Equal to zero \n"
 4
         msg2 : .asciiz "Less than Zero\n"
 5
         msg3 : .asciiz "Greater than Zero\n"
 6
     .text
 7
     .globl main
8
     main:
9
          #output input msg
         li $v0,4
12
         la $a0, input msg
13
         syscall
14
          #input value from user and save it in register t1
16
                           #load 5 into v0
          li $v0,5
          syscall
18
19
         move $t1, $v0  #move the entered value from v0 to t1 register
20
21
         beq $t1, $zero, eq to zero
         blt $t1, $zero, ls th zero
```

```
🗵 🔚 task2.asm 🗵 📙 task3.asm 🗵 🔡 task4.asm 🗵 🔡 task5.asm 🗵 🔡 client.c 🗵 🔡 practice.asm
19
          move $t1, $v0
                             #move the entered value from v0 to t1 register
20
          beq $t1, $zero, eq to zero
          blt $t1, $zero, ls th zero
23
          bgt $t1, $zero, gr th zero
2.4
     eq to zero:
26
          #output msg1
27
          li $v0,4
28
          la $a0, msg1
29
          syscall
          #j main
31
32
          #exit the process
33
          li $v0, 10
34
          syscall
36
     ls th zero:
37
          #output msg2
```

```
task1.asm 🗵 블 task2.asm 🗵 📙 task3.asm 🗵 🛗 task4.asm 🗵 🔡 task5.asm 🗵 🛗 client.c 🗵 🛗 practice.asm 🗵
37
           #output msq2
38
           li $v0,4
39
           la $a0, msg2
40
           syscall
41
42
           #j main
           #exit the process
43
           li $v0, 10
44
45
           syscall
46
47
      gr th zero:
           #output msq3
48
49
           li $v0,4
50
           la $a0, msq3
51
           syscall
52
53
           #j main
54
           #exit the process
           li $v0, 10
55
56
           syscall
57
58
```



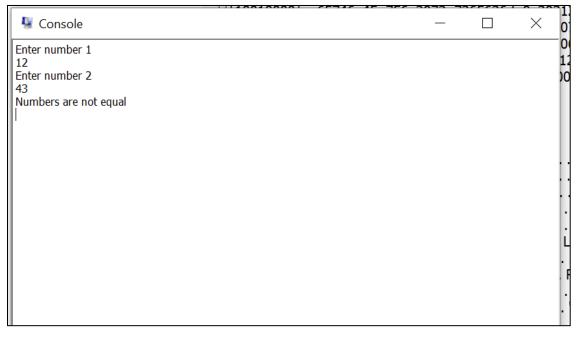
### Task 3:

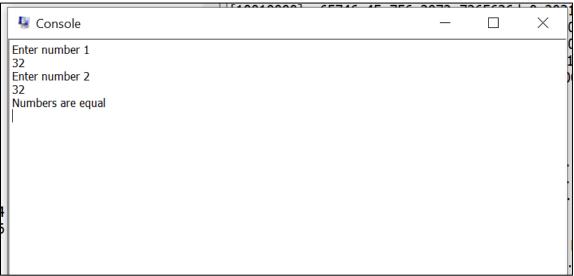
Check using branch whether the number input by user are equal or not ( Use beq).

```
task1.asm 🗵 📙 task2.asm 🗵 🛗 task3.asm 🗵 🔡 task4.asm 🗵 🛗 task5.asm 🗵 🛗 client.c 🗵 💾 practice.asm 🗵
     .data
 2
          input msg1 : .asciiz "Enter number 1 \n"
 3
          input msg2 : .asciiz "Enter number 2 \n"
          msg1 : .asciiz "Numbers are equal\n"
 4
 5
          msq2 : .asciiz "Numbers are not equal\n"
 6
      .text
 7
      .globl main
     main:
 9
10
          #output msg1
                             #load 4 into v0
11
          li $v0,4
12
          la $a0, input msg1 #load address of msg1 to a0
13
          syscall
14
          #input value from user and save it in register t1
15
          li $v0,5
                             #load 5 into v0
16
          syscall
17
          move $t1, $v0
18
                             #move the entered value from v0 to t1 reg
19
```

```
task1.asm 🗵 🔚 task2.asm 🗵 🔡 task3.asm 🗵 🔡 task4.asm 🗵 🔡 task5.asm 🗵 🔡 client.c 🗵 🔡 practice.asm 🗵
19
20
          #output msg2
                            #load 4 into v0
21
          li $v0,4
          la $a0, input msg2 #load address of msg1 to a0
22
23
          syscall
24
          #input value from user and save it in register t2
25
26
          li $v0,5
                            #load 5 into v0
          syscall
27
          move $t2, $v0
                            #move the entered value from v0 to t2 reg
28
29
          beg $t1, $t2, equal
30
31
32
          #output msg2
          li $v0,4
33
34
          la $a0, msg2
35
          syscall
36
37
          #exit the process
          li $v0, 10
38
39
          syscall
40
```

```
41
42
     equal:
          #output msg1
43
          li $v0,4
44
45
          la $a0, msg1
46
          syscall
47
          #exit the process
48
49
          li $v0, 10
50
          syscall
```





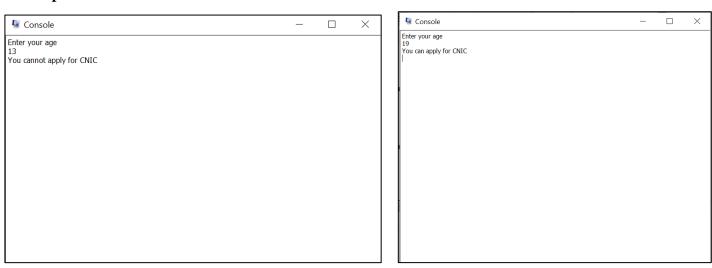
### Task 4:

Write the assembly of the below C++ code:

```
Int age;
Cout<<"enter your age"<<endl;
Cin>>age;
If(age > 18)
{
Cout<<"you can apply for CNIC"<<endl;
}
Else
{
Cout<<"you cannot apply for CNIC"<<endl;
}
```

```
🖁 task1.asm 🗵 🔡 task2.asm 🗵 🔡 task3.asm 🗵 🔡 task4.asm 🗵 🔡 task5.asm 🗵 🛗 client.c 🗵 🔡 practice.asm 🗵
 1 .data
 2
          input msg1 : .asciiz "Enter your age \n"
 4
          msg1 : .asciiz "You can apply for CNIC\n"
          msq2 : .asciiz "You cannot apply for CNIC\n"
 5
 6
     .text
     .globl main
     main:
 9
10
          #output msg1
          li $v0,4
                           #load 4 into v0
11
12
          la $a0, input msg1 #load address of msg1 to a0
13
          syscall
14
15
          #input value from user and save it in register t1
16
          li $v0,5
                      #load 5 into v0
          syscall
17
          move $t1, $v0 #move the entered value from v0 to t1 register
19
          li $t2,18
21
          bgt $t1, $t2, success
```

```
📑 task2.asm 🗵 📑 task3.asm 🗵 🔚 task4.asm 🗵 🔡 task5.asm 🗵 🔠 client.c 🗵 🛗 practice.asm 🗵
22
           bgt $t1, $t2, success
23
24
           #output msg2
25
           li $v0,4
           la $a0, msg2
26
27
           syscall
28
29
           #exit the process
           li $v0, 10
30
           syscall
31
32
33
34
      success:
35
           #output msg1
           li $v0,4
36
           la $a0, msg1
37
           syscall
38
39
           #exit the process
40
           li $v0, 10
41
           syscall
42
```



### 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)). Below is the C++ language code:

```
Int limit;
Int sum;
Cout<<"Enter a number"<<endl;
Cin>>limit;
for (int i = 1; i <= limit; ++i) {
    sum += i;
}
Cout<<"sum of numbers from 1 to <<li>limit<<"is"<<sum<<endl;</pre>
```

```
ask1.asm 🗵 🔚 task2.asm 🗵 🔡 task3.asm 🗵 🔚 task4.asm 🗵 🔚 task5.asm 🗵 🔡 client.c 🗵 🔡 practice.asm 🗵
     .data
2
          input msg1 : .asciiz "Enter the limit \n"
3
         msq1: .asciiz "Sum of numbers from 1 to "
4
5
         msq2 : .asciiz " is:\n"
6
     .text
7
     .globl main
     main:
9
10
          #output msg1
11
          li $v0,4
                            #load 4 into v0
12
          la $a0, input msg1 #load address of msg1 to a0
13
          syscall
14
15
          #input value from user and save it in register t1
16
          li $v0,5
                            #load 5 into v0
17
          syscall
18
         move $t1, $v0  #move the entered value from v0 to t1 regis
19
20
          li $t2,1
21
          li $t3,0
          j success
```

```
task1.asm 🗵 🔡 task2.asm 🗵 🔡 task3.asm 🗵 🔡 task4.asm 🗵 🔡 task5.asm 🗵 🔡 client.c 🗵 🔡 practice.asm 🗵
24
      success:
25
           add $t3, $t3, $t2
26
           bge $t2, $t1, success2
           addi $t2, 1
27
28
           j success
29
30
      success2:
           #output msg1
31
           li $v0,4
33
           la $a0, msg1
34
           syscall
36
           #output
37
           li $v0,1
           move $a0, $t1
39
           syscall
40
41
           #output msg2
42
           li $v0,4
           la $a0, msg2
43
44
           syscall
45
46
            #output
```

```
#output
li $v0,1
move $a0, $t3
syscall

#exit the process
li $v0, 10
syscall

syscall
```

```
Enter the limit
5
Sum of numbers from 1 to 5 is:
15
```

### **Conclusion:**

In this lab, I learned about the branching instructions (Control Structures) in MIPS Assembly.