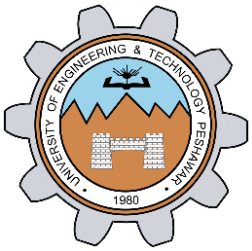
# BRANCHING AND SHIFTING OPERATIONS

**LAB # 03**




**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:

**19th 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:_____<br>Instructor Signature: _____ | | | | |

## Task 1:

Take the 1st number from user. Then take a number to do the operation. (1 corresponds to addition, 2 corresponds to subtraction, 3 for multiplication and 4 for division). Then finally take a 2nd number from a user.

## Code:

task1.asm task4.asm task5.asm

```
.data
    msg1 : .asciiz "Enter the number 1: \n"
    msg2 : .asciiz "Enter 1 for addition, 2 for subtraction, 3 for multiplication and 4 for
    msg3 : .asciiz "Enter the number 2: \n"
    msg4 : .asciiz "Result: \n"
.text
.globl main
main:

    #output msg1
    li $v0,4        #load 4 into v0
    la $a0, msg1    #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5        #load 5 into v0
    syscall
    move $t1, $v0   #move the entered value from v0 to t1 register


    #output msg2
    li $v0,4
```

task1.asm task4.asm task5.asm

```
    li $v0,4
    la $a0, msg2
    syscall

    #input value from user and save it in register t2
    li $v0,5        #load 5 into v0
    syscall
    move $t2, $v0   #move the entered value from v0 to t2 register


    #output msg3
    li $v0,4
    la $a0, msg3
    syscall

    #input value from user and save it in register t3
    li $v0,5        #load 5 into v0
    syscall
    move $t3, $v0   #move the entered value from v0 to t3 register

    beq $t2,1,addition
    beq $t2,2,subtraction
```

task1.asm | task4.asm | task5.asm

```
43      beq $t2,2,subtraction
44      beq $t2,3,multiplication
45      beq $t2,4,division
46
47      j end_program
48
49  addition:
50      add $t4, $t1,$t3
51
52      #output msg4
53      li $v0,4
54      la $a0, msg4
55      syscall
56
57      li $v0,1
58      move $a0, $t4
59      syscall
60      j end_program
61
62  subtraction:
63      sub $t4, $t1,$t3
64
```

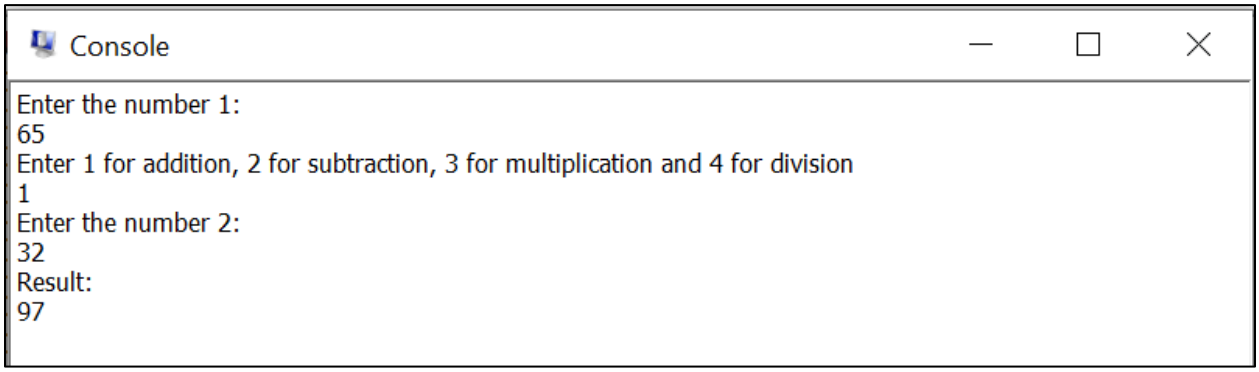
task1.asm | task4.asm | task5.asm

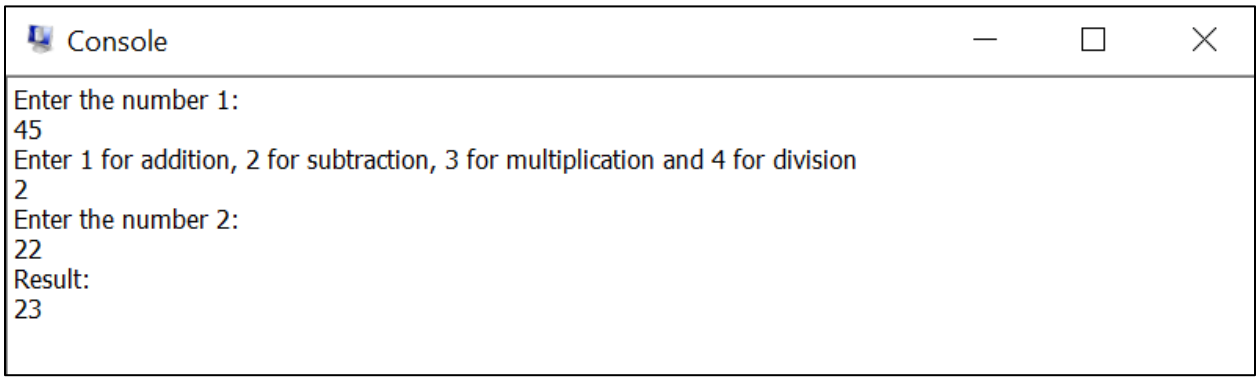
```
64
65      #output msg4
66      li $v0,4
67      la $a0, msg4
68      syscall
69
70      li $v0,1
71      move $a0, $t4
72      syscall
73
74      j end_program
75
76  multiplication:
77      mul $t4, $t1,$t3
78
79      #output msg4
80      li $v0,4
81      la $a0, msg4
82      syscall
83
84      li $v0,1
85      move $a0, $t4
```

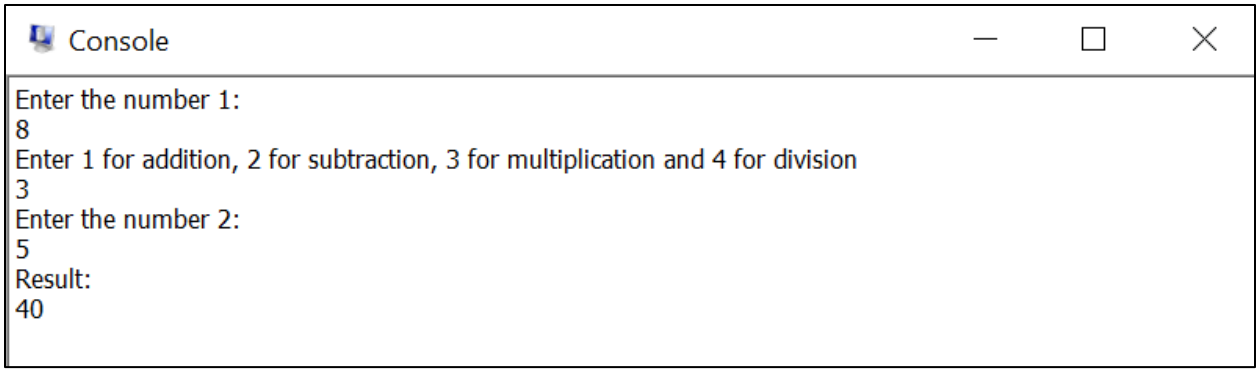
task1.asm | task4.asm | task5.asm

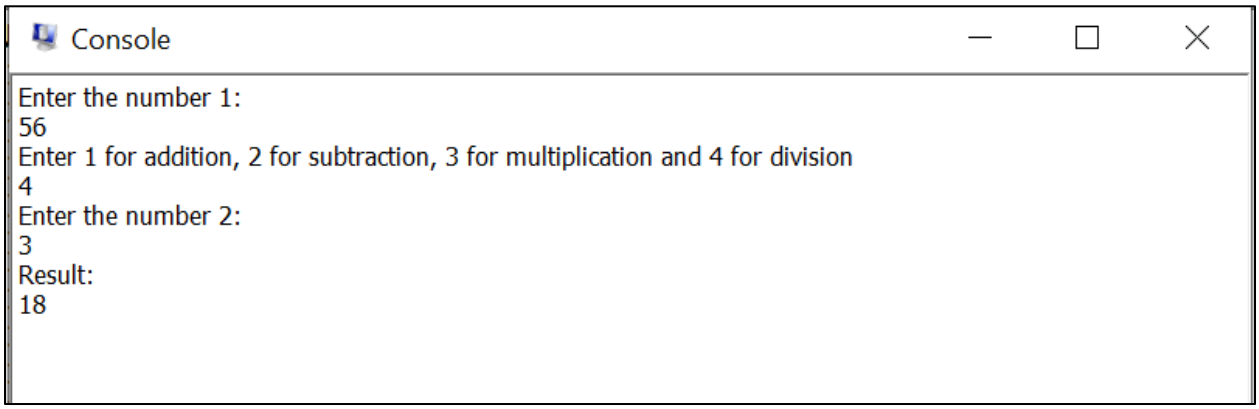
```
85      move $a0, $t4
86      syscall
87      j end_program
88
89  division:
90      div $t4, $t1,$t3
91
92      #output msg4
93      li $v0,4
94      la $a0, msg4
95      syscall
96
97      li $v0,1
98      move $a0, $t4
99      syscall
100     j end_program
101
102 end_program:
103
104     #exit the process
105     li $v0, 10
106     syscall
```

**Output:**




Console
Enter the number 1:
8
Enter 1 for addition, 2 for subtraction, 3 for multiplication and 4 for division
3
Enter the number 2:
5
Result:
40


Console
Enter the number 1:
56
Enter 1 for addition, 2 for subtraction, 3 for multiplication and 4 for division
4
Enter the number 2:
3
Result:
18

**Task 2 & 3:**

Write a program that's show the bit position of a number is 0 or 1. (Hint if number is 5 it is represented by 0101 show the 4th bit position is 0, similarly if the user enters 9 then the binary equivalent is 1001. In this case the 4th bit position is 1).

Now toggle the bit find in the previous task if the bit is 1 set it to 0 if it is 0 then set it to 1.

**Code:**

task1.asm | task4.asm | task5.asm | task2.asm

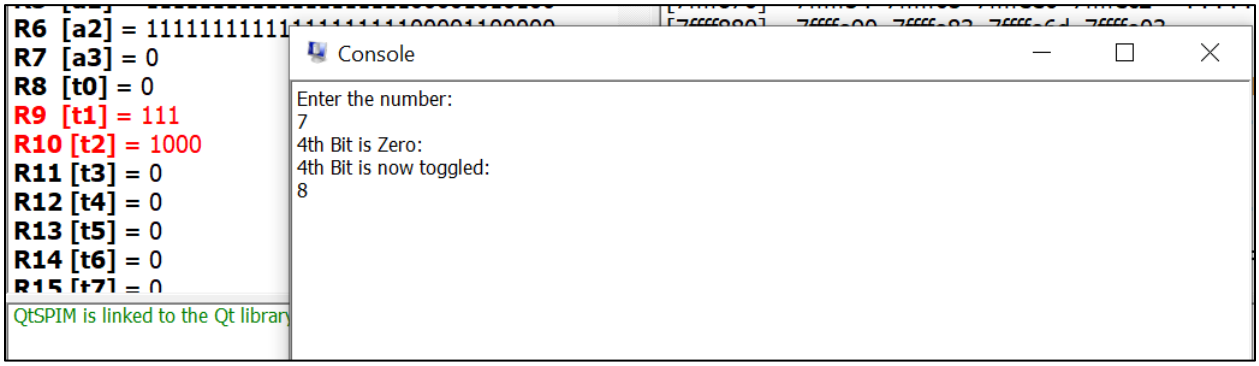
```
.data
    msg1 : .asciiz "Enter the number: \n"
    msg2 : .asciiz "4th Bit is One: \n"
    msg3 : .asciiz "4th Bit is Zero: \n"
    msg4 : .asciiz "4th Bit is now toggled: \n"
.text
.globl main
main:

    #output msg1
    li $v0,4          #load 4 into v0
    la $a0, msg1      #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5          #load 5 into v0
    syscall
    move $t1, $v0     #move the entered value from v0 to t1 register

    andi $t3, $t1, 8

    beq $t3, 8, One #1000 in binary
    beq $t3, 0, Zero #0000 in binary
```

```
23      beq $t3, 0, Zero #0000 in binary
24
25      j end_program
26
27  One:
28      #output msg2
29      li $v0,4        #load 4 into v0
30      la $a0, msg2    #load address of msg1 to a0
31      syscall
32
33      j one_to_zero
34
35  Zero:
36      #output msg3
37      li $v0,4        #load 4 into v0
38      la $a0, msg3    #load address of msg1 to a0
39      syscall
40
41      j zero_to_one
42
43  one_to_zero:
44      andi $t2, 7 #binary 0111
```

```
45
46      #output msg4
47      li $v0,4        #load 4 into v0
48      la $a0, msg4    #load address of msg1 to a0
49      syscall         #load 4 into v0
50
51      li $v0,1
52      move $a0, $t2
53      syscall
54      j end_program
55
56
57  zero_to_one:
58      or $t2, 8 #binary 1000
59
60      #output msg4
61      li $v0,4        #load 4 into v0
62      la $a0, msg4    #load address of msg1 to a0
63      syscall         #load 4 into v0
64
65      li $v0,1
```

```
65      li $v0,1
66      move $a0, $t2
67      syscall
68
69      j end_program
70
71
72  end_program:
73      #exit the process
74      li $v0, 10
75      syscall
```

**Output:**



**Task 4:**

Write a program to check a number entered by user is even or odd.

**Code:**

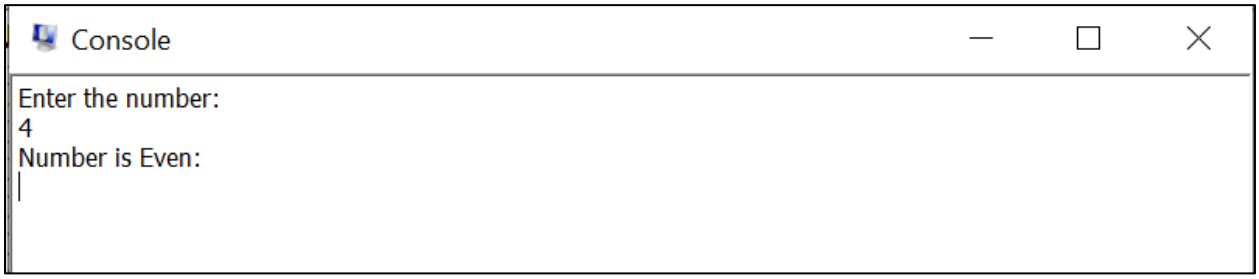
task1.asm | task4.asm | task5.asm

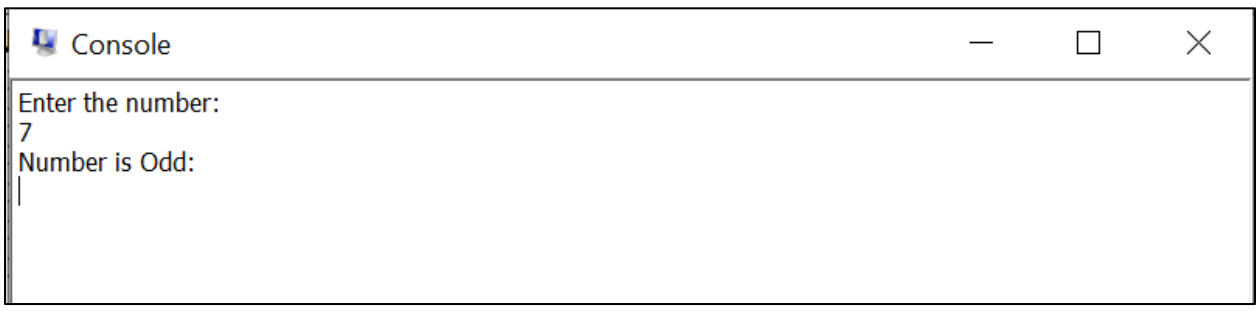
```
.data
    msg1 : .asciiz "Enter the number: \n"
    msg2 : .asciiz "Number is Even: \n"
    msg3 : .asciiz "Number is Odd: \n"

.text
.globl main
main:

    #output msg1
    li $v0,4          #load 4 into v0
    la $a0, msg1      #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5          #load 5 into v0
    syscall
    move $t1, $v0     #move the entered value from v0 to t1 register

    andi $t3, $t1, 1 #Binary 0001 to extract first bit

    beq $t3, 0, EvenNumber
    beq $t3, 1, OddNumber
```

task1.asm | task4.asm | task5.asm

```
23      beq $t3, 1, OddNumber
24
25      j end_program
26
27  EvenNumber:
28      #output msg2
29      li $v0,4          #load 4 into v0
30      la $a0, msg2      #load address of msg2 to a0
31      syscall
32
33      j end_program
34  OddNumber:
35
36      #output msg3
37      li $v0,4          #load 4 into v0
38      la $a0, msg3      #load address of msg3 to a0
39      syscall
40      j end_program
41  end_program:
42
43      #exit the process
44      li $v0, 10
45      syscall
```

**Output:**



**Task 5:**

Show that shifting left of an even number by 1 position is a multiplication by 2 and shifting right of an even number by 1 position is a division by 2. (Hint: Use sll and srl).

**Code:**

task1.asm task4.asm task5.asm

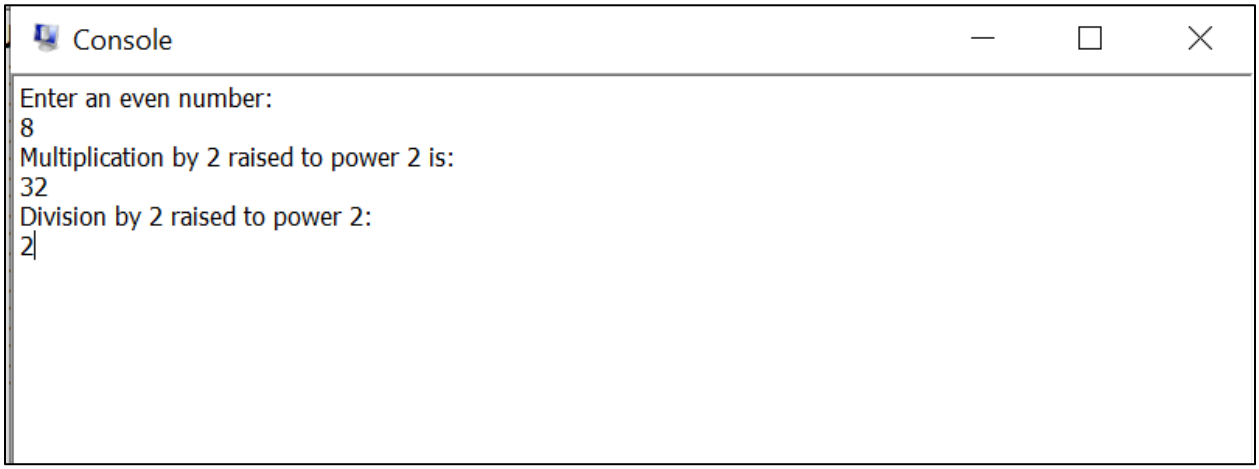
```
.data
    msg1 : .asciiz "Enter an even number: \n"
    msg2 : .asciiz "Multiplication by 2 raised to power 2 is: \n"
    msg3 : .asciiz "\nDivision by 2 raised to power 2: \n"

.text
.globl main
main:

    #output msg1
    li $v0,4          #load 4 into v0
    la $a0, msg1      #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5          #load 5 into v0
    syscall
    move $t1, $v0     #move the entered value from v0 to t1 register

    sll $t2, $t1, 2 #shift left by amount of 2
    srl $t3, $t1, 2 #shift right by amount of 2
```

task1.asm | task4.asm | task5.asm

```
21      srl $t3, $t1, 2 #shift right by amount of 2
22
23      #output msg2
24      li $v0,4        #load 4 into v0
25      la $a0, msg2    #load address of msg1 to a0
26      syscall
27
28      li $v0,1
29      move $a0, $t2
30      syscall
31
32      #output msg3
33      li $v0,4        #load 4 into v0
34      la $a0, msg3    #load address of msg1 to a0
35      syscall
36
37      li $v0,1
38      move $a0, $t3
39      syscall
40      j end_program
```

```
42  end_program:
43
44      #exit the process
45      li $v0, 10
46      syscall
```

**Output:**



**Conclusion:**

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