# BRANCHING OPERATIONS IN ASSEMBLY

## LAB # 03

**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**

Monday, October 16, 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<br>[Marks 10] | Objectives of lab are partially covered<br>[Marks 5] | Objectives of lab are not shown<br>[Marks 0] | |
| **2. MIPS instructions with Comments and proper indentations.** | All the instructions are well written with comments explaining the code and properly indented<br>[Marks 20] | Some instructions are missing are poorly commented code<br>[Marks 10] | The instructions are not properly written<br>[Marks 0] | |
| **3. Simulation run without error and warnings** | The code is running in the simulator without any error and warnings<br>[Marks 10] | The code is running but with some warnings or errors.<br>[Marks 5] | The code is written but not running due to errors<br>[Marks 0] | |
| **4. Procedure** | All the instructions are written with proper procedure<br>[Marks 20] | Some steps are missing<br>[Marks 10] | steps are totally missing<br>[Marks 0] | |
| **5. OUTPUT** | Proper output of the code written in assembly<br>[Marks 20] | Some of the outputs are missing<br>[Marks 10] | No or wrong output<br>[Marks 0] | |
| **6. Conclusion** | Conclusion about the lab is shown and written<br>[Marks 20] | Conclusion about the lab is partially shown<br>[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: ________________________ | | | | |

# Branching Operations

## Objectives:

- Learn about branching
- Masking a bit and then manipulating it
- Shifting logical operators

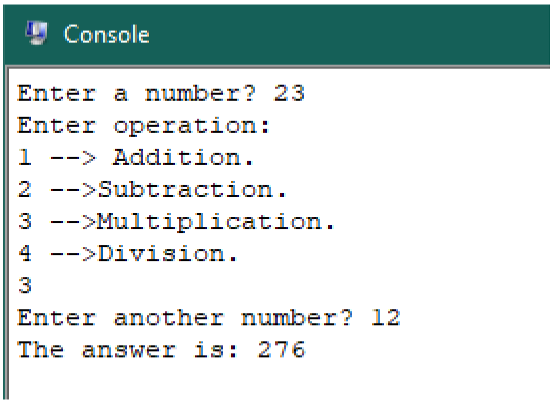
## Tasks:

**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. (use branching i.e beq and j).

**Code:**

```
.text
.globl main

main:
    # Take a number from user in $t0
    li $v0, 4
    la $a0, prompt1
    syscall

    li $v0, 5
    syscall
    move $t0, $v0

    # Take Operator from user in $t1
    li $v0, 4
    la $a0, operationsList
    syscall

    li $v0, 5
    syscall
    move $t1, $v0

    # Take another number from user $t2
    li $v0, 4
    la $a0, prompt2
    syscall

    li $v0, 5
    syscall
    move $t2, $v0


    # Making the calculator logic
    li $t3, 1
    li $t4, 2
    li $t5, 3
    li $t6, 4

    beq $t1, $t3, Addition
    beq $t1, $t4, Subtraction
    beq $t1, $t5, Multiplication
    beq $t1, $t6, Division

    Addition:
        add $t7, $t0,$t2
        j Display_Answer

    Subtraction:
        sub $t7, $t0,$t2
        j Display_Answer

    Multiplication:
        mul $t7, $t0,$t2
        j Display_Answer

    Division:
        div $t7, $t0,$t2
        j Display_Answer

    Display_Answer:
        li $v0, 4
        la $a0, answer
        syscall

        li $v0, 1 # print The expression (int)
        move $a0, $t7
        syscall

        j End

    End:
        li $v0, 10      # Exit the program
        syscall


.data
      prompt1: .asciiz "Enter a number? "
    operationsList: .asciiz "Enter operation:\n1 --> Addition.\n2 --
>Subtraction.\n3 -->Multiplication.\n4 -->Division.\n"
      prompt2: .asciiz "Enter another number? "
      answer: .asciiz "The answer is: "
```

**Output:**



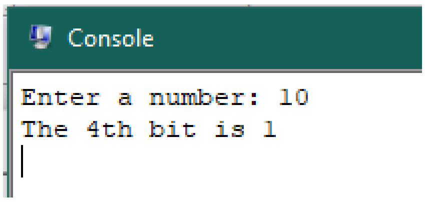
```
Enter a number? 23
Enter operation:
1 --> Addition.
2 -->Subtraction.
3 -->Multiplication.
4 -->Division.
3
Enter another number? 12
The answer is: 276
```

**Task 2**: 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).

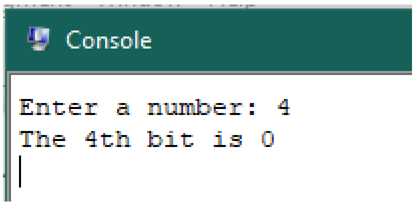
**Code:**

```
.text
.globl main

main:
    # Display the prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read user input
    li $v0, 5
    syscall
    move $t0, $v0  # Store the input in $t0

    # Check the 4th bit:
        #   To check whether the 4th bit is 0 or 1, we use a bitwise
AND operation. We load the value 0x08 into $t1, which is the binary
representation of 0000 1000. This value has all bits set to 0 except
the 4th bit.
        #   We then perform an AND operation between the user input
$t0 and the mask $t1. If the 4th bit of $t0 is 1, the result of the
AND operation will be non-zero. If it's 0, the result will be 0.
    li $t1, 0x08
    and $t0, $t0, $t1
    bnez $t0, bit_is_1

    # If the 4th bit is 0
    li $v0, 4
    la $a0, result0
    syscall
    j exit

bit_is_1:
    # If the 4th bit is 1
    li $v0, 4
    la $a0, result1
    syscall

exit:
    # Exit the program
    li $v0, 10
    syscall


.data
    prompt: .asciiz "Enter a number: "
    result0: .asciiz "The 4th bit is 0\n"
    result1: .asciiz "The 4th bit is 1\n"
```

**Output:**



```
Enter a number: 10
The 4th bit is 1
```



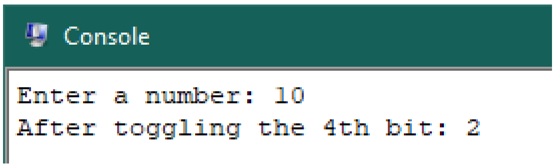
```
Enter a number: 4
The 4th bit is 0
```

**Task 3**: 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

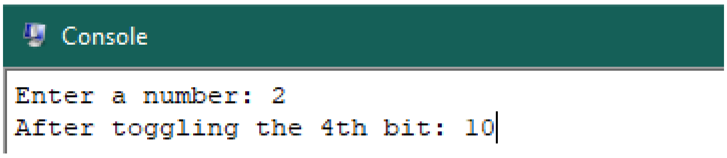
**Code:**

```
.text
.globl main

main:
    # Display the prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read user input
    li $v0, 5
    syscall
    move $t0, $v0  # Store the input in $t0

    # Toggle the 4th bit
    li $t1, 0x08   # Binary: 0000 1000
    xor $t0, $t0, $t1

    # Display the result
    li $v0, 4
    la $a0, result
    syscall
    li $v0, 1
    move $a0, $t0
    syscall

    # Exit the program
    li $v0, 10
    syscall

.data
    prompt: .asciiz "Enter a number: "
    result: .asciiz "After toggling the 4th bit: "
```

**Output:**



```
Enter a number: 10
After toggling the 4th bit: 2
```



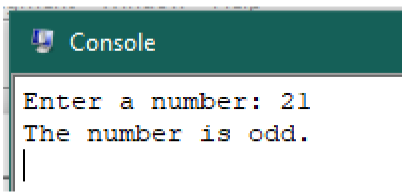
```
Enter a number: 2
After toggling the 4th bit: 10
```

**Task 4**: Write a program to check a number entered by user is even or odd.

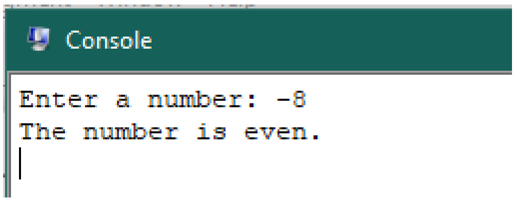
**Code:**

```
.text
.globl main

main:
    # Display the prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read user input
    li $v0, 5
    syscall
    move $t0, $v0  # Store the input in $t0

    # Check the last bit:
    li $t1, 0x01
    and $t0, $t0, $t1
    bnez $t0, bit_is_1

    # If the last bit is 0 it is even
    li $v0, 4
    la $a0, result0
    syscall
    j exit

bit_is_1:
    # If the last bit is 1 it is odd
    li $v0, 4
    la $a0, result1
    syscall

exit:
    # Exit the program
    li $v0, 10
    syscall


.data
    prompt: .asciiz "Enter a number: "
    result0: .asciiz "The number is even.\n"
    result1: .asciiz "The number is odd.\n"
```

**Output:**



```
Enter a number: 21
The number is odd.
```



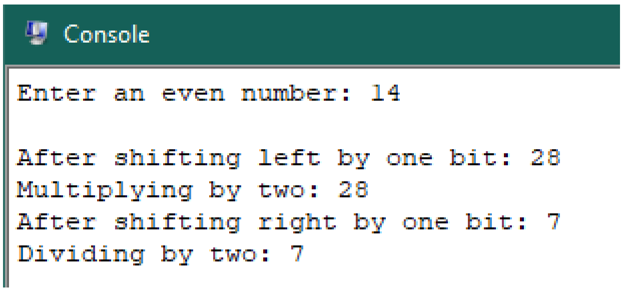
```
Enter a number: -8
The number is even.
```

**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:**

```
.text
.globl main

main:
    # Display the prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read user input
    li $v0, 5
    syscall
    move $t0, $v0  # Store the input in $t0

    li $t1, 2 # store 2 in t1

    # shifting to left by 1 bit
    sll $t2, $t0, 1
    li $v0, 4
    la $a0, shiftedLeft
    syscall

    li $v0, 1
    move $a0, $t2
    syscall

    # multiplied by 2
    mul $t3, $t0, $t1

    li $v0, 4
    la $a0, multAnswer
    syscall

    li $v0, 1 # print The expression (int)
    move $a0, $t3
    syscall

    # shifted to right by 1 bit
    srl $t4, $t0, 1

    li $v0, 4
    la $a0, shiftedRight
    syscall

    li $v0, 1
    move $a0, $t4
    syscall

    # division by 2
    div $t5, $t0, $t1

    li $v0, 4
    la $a0, divAnswer
    syscall

    li $v0, 1 # print The expression (int)
    move $a0, $t5
    syscall

    # Exit the program
    li $v0, 10
    syscall

.data
    prompt: .asciiz "Enter an even number: "
    shiftedLeft: .asciiz "\nAfter shifting left by one bit: "
    multAnswer: .asciiz "\nMultiplying by two: "
    shiftedRight: .asciiz "\nAfter shifting right by one bit: "
    divAnswer: .asciiz "\nDividing by two: "
```

**Output:**



```
Enter an even number: 14

After shifting left by one bit: 28
Multiplying by two: 28
After shifting right by one bit: 7
Dividing by two: 7
```

## Reference:

To view my codes, please refer to my GitHub Account.

## Conclusion:

In this lab I have learn about conditional and unconditional branching. I also learned how can I access nth bit of a number. I also learn how to change, update, read the nth bit. At last I also learned the left shifting and right shifting logical operators and their connection with multiplication and division.

The End.