Computer Organization & Architecture Lab

Lab Report # 03

Submitted By: AWAIS SADDIQUI

Registration No: 21PWCSE1993

Section: "A"

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

Student Signature: _____

Submitted to:

Dr. Bilal Habib

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	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	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	Conclusion about the lab is not shown[Marks0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained: _____ Instructor Signature: _____				

Branching Operation

Task 01:

Take the 1st number from the user. Then take a number to do the operation. Then finally take a 2nd number from a user.

Code:

a)

Task1.asm U

Task1.asm

```
.data
    input1 : .asciiz "Enter a Ist number: "
    operation : .asciiz "To perform operation (1: Add, 2: Subtract, 3: Multiply,
    input2 : .asciiz "Enter 2nd number : "
    result : .asciiz "Result :) "

.text
.globl main
main:

    li $v0, 4
    la $a0, input1
    syscall

    li $v0, 5
    syscall
    move $s0, $v0

    li $v0, 4
    la $a0, operation
    syscall

    li $v0, 5
    syscall
    move $s1, $v0

    li $v0, 4
    la $a0, input2
    syscall

    li $v0, 5
    syscall
    move $s2, $v0
```

b)

```
        beq $s1, 1, addition
        beq $s1, 2, subtraction
        beq $s1, 3, multiplication
        beq $s1, 4, divide
        j return
addition:
        add $s3, $s0, $s2
        j finalAnswer

subtraction:
        sub $s3, $s2, $s0
        j finalAnswer

multiplication:
        mul $s3, $s0, $s2
        j finalAnswer

divide:
        div $s2, $s0
        mflo $s3
        j finalAnswer

finalAnswer:
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 1
        move $a0, $s3
        syscall

return:
        li $v0 , 10
        syscall
```

Output:

```
Console
Enter a Ist number: 15
To perform operation (1: Add, 2: Subtract, 3: Multiply, 4: Divide :3
Enter 2nd number : 5
Result :) 75
```

Task 02:

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:

Task2.asm

```
.data
    prompt: .asciiz "Enter a number: "
    bit_position_prompt: .asciiz "Enter the bit position to check (0-based index)
    result_0: .asciiz "The bit at the specified position is 0."
    result_1: .asciiz "The bit at the specified position is 1."

.text
main:
    li $v0, 4
    la $a0, prompt
    syscall

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

    li $v0, 4
    la $a0, bit_position_prompt
    syscall

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

    srl $t0, $t0, $t1

    andi $t0, $t0, 1

    beqz $t0, bit_is_0
    j bit_is_1

bit_is_0:
    li $v0, 4
    la $a0, result_0
    syscall
    j done
```

```
bit_is_1:
    li $v0, 4
    la $a0, result_1
    syscall

done:
    # Exit
    li $v0, 10
    syscall
```

Output:

```
Console
Enter a number: 5
Enter the bit position to check (0-based index): 4
The bit at the specified position is 0.
```

Task 03:

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:

Task3.asm

```
.data
prompt: .asciiz "Enter a number: "
bit_position_prompt: .asciiz "Enter the bit position to toggle (0-based index): "
result_msg: .asciiz "Toggled number: "

.text
main:
    li $v0, 4
    la $a0, prompt
    syscall

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

    li $v0, 4
    la $a0, bit_position_prompt
    syscall

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

    li $t2, 1
    sll $t2, $t2, $t1

    xor $t0, $t0, $t2

    li $v0, 4
    la $a0, result_msg
    syscall

    li $v0, 1
    move $a0, $t0
    syscall
```

Output:

```
Console
Enter a number: 5
Enter the bit position to toggle (0-based index): 4
Toggled number: 21
```

Task 05:

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:

Task5.asm

```
.data
    prompt: .asciiz "Enter a number: "
    even_msg: .asciiz "The number is even."
    odd_msg: .asciiz "The number is odd."

.text
main:
    li $v0, 4
    la $a0, prompt
    syscall

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

    andi $t1, $t0, 1  # Use AND with 1 to check the least significant bit (LSB)

    beqz $t1, number_is_even
    j number_is_odd

number_is_even:
    li $v0, 4
    la $a0, even_msg
    syscall
    j done

number_is_odd:
    li $v0, 4
    la $a0, odd_msg
    syscall

done:
    li $v0, 10
    syscall
```

Output:

Console

```
Enter a number: 7
The number is odd.
```

Console

```
Enter a number: 6
The number is even.
```