DATA TRANSFER
IN MIPS

LAB # 04

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

Thursday, October 26, 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] [Marks 0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained: __________ Instructor Signature: ____________________				

Data Transfer in MIPS

Objectives:

- How to write/store data in memory
- How to read/load data from memory

Tasks:

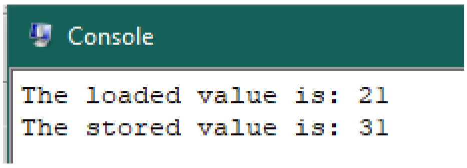
Task 1: Load a value from memory and add 10 to it. Store the result back in memory and show the result on console. (*hint: use MIPS instructions lw and sw*)

Code:

```
.text
.globl main

main:
    # display message
    li $v0, 4
    la $a0, before
    syscall

    # load the word
    lw $t0, num

    # display the number
    li $v0, 1
    move $a0, $t0
    syscall

    addi $t1, $t0, 10

    # store the value
    sw $t1, num

    # display the message
    li $v0, 4
    la $a0, after
    syscall

    # display the stored value
    li $v0, 1
    move $a0, $t1
    syscall

    j end

    end:
        li $v0, 10      # Exit the program
```

```
         syscall


.data
    num: .word 21
    before: .asciiz "The loaded value is: "
    after: .asciiz "\nThe stored value is: "
```

Output:



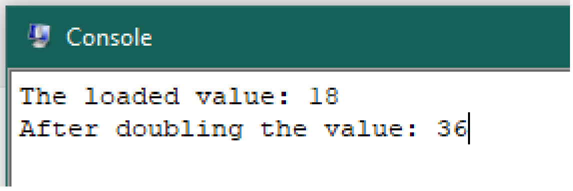
```
The loaded value is: 21
The stored value is: 31
```

Task 2: Load a value from memory and double it. Store the result back in memory also show on the console. (*use sll, sw and lw*)

Code:

```
.text
.globl main

main:
    # display message
    li $v0, 4
    la $a0, before
    syscall

    # load the value
    lw $t0, value

    # display the number
    li $v0, 1
    move $a0, $t0
    syscall

    # Double the value using sll (shift left logical)
    sll $t1, $t0, 1  # Shift left by 1 bit to double the value
    sw $t1, value

    # display the message
    li $v0, 4
    la $a0, after
    syscall

    # display the number
    li $v0, 1
    move $a0, $t1
    syscall

    j end
    end:
        li $v0, 10      # Exit the program
        syscall

.data
    value: .word 18
    before: .asciiz "The loaded value: "
    after: .asciiz "\nAfter doubling the value: "
```

Output:



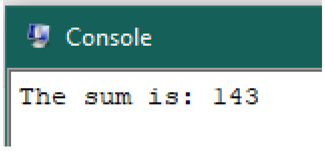
```
The loaded value: 18
After doubling the value: 36
```

Task 3: Load an address of a label into a register and jump to that address and perform addition in that address. *.(use jr(jump register))*

Code:

```
.text
.globl main

main:
    la $t0, additionLabel # load label to register
    jr $t0 # jump to the label

    additionLabel:
        li $t1, 75
        li $t2, 68
        add $t3, $t1, $t2

        # print the message
        li $v0, 4
        la $a0, result
        syscall

        # print the integer
        li $v0, 1
        move $a0, $t3
        syscall

        j end

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

.data
    result: .asciiz "The sum is: "
```

Output:



```
The sum is: 143
```

Task 4: Write assembly program to find the Fibonacci series.

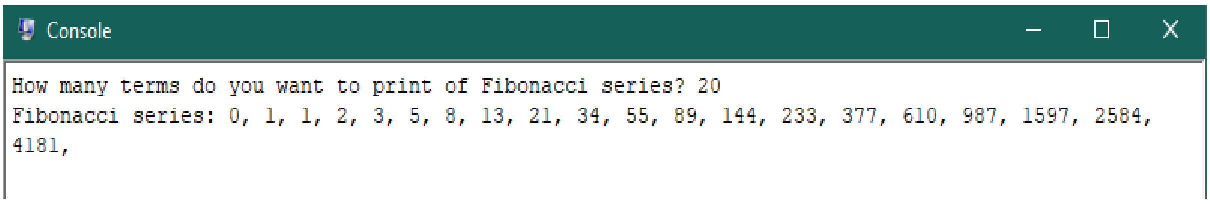
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Users will be asked to enter a number, for instance 9. Then assembly will print the first 9 numbers of Fibonacci series.

Code:

```
.text
.globl main

main:
    # display prompt and read 'n'
    li $v0, 4
    la $a0, prompt
    syscall

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

    # init first and second
    li $t1, 0
    li $t2, 1

    # print first and second with commas
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $t1
    syscall

    li $v0, 4
    la $a0, comma
    syscall

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

    li $v0, 4
    la $a0, comma
    syscall
    # init loop 'i'
    li $t3, 2

    loop:
        add $t4, $t1, $t2 # next = first + second

        # print the sequence with commas
        li $v0, 1
        move $a0, $t4
        syscall

        li $v0, 4
        la $a0, comma
        syscall

        # update i, first and second
        move $t1, $t2
        move $t2, $t4
        addi $t3, $t3, 1

        # exit loop if we have generated n terms
        beq $t3, $t0, end

        j loop


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

.data
    prompt: .asciiz "How many terms do you want to print of 
Fibonacci series? "
    result: .asciiz "Fibonacci series: "
    comma: .asciiz ", "
```

Output:



```
How many terms do you want to print of Fibonacci series? 20
Fibonacci series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181,
```

Reference:

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

Conclusion:

In this lab I have learnt how can we access the data in memory. Now I am able to read / load data from memory to a register and also can write / store from register to memory. I have learnt how to transfer data among registers and memory in MIPS.

The End.