DATA TRANSFER IN MIPS

LAB # 04

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:

27th October 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

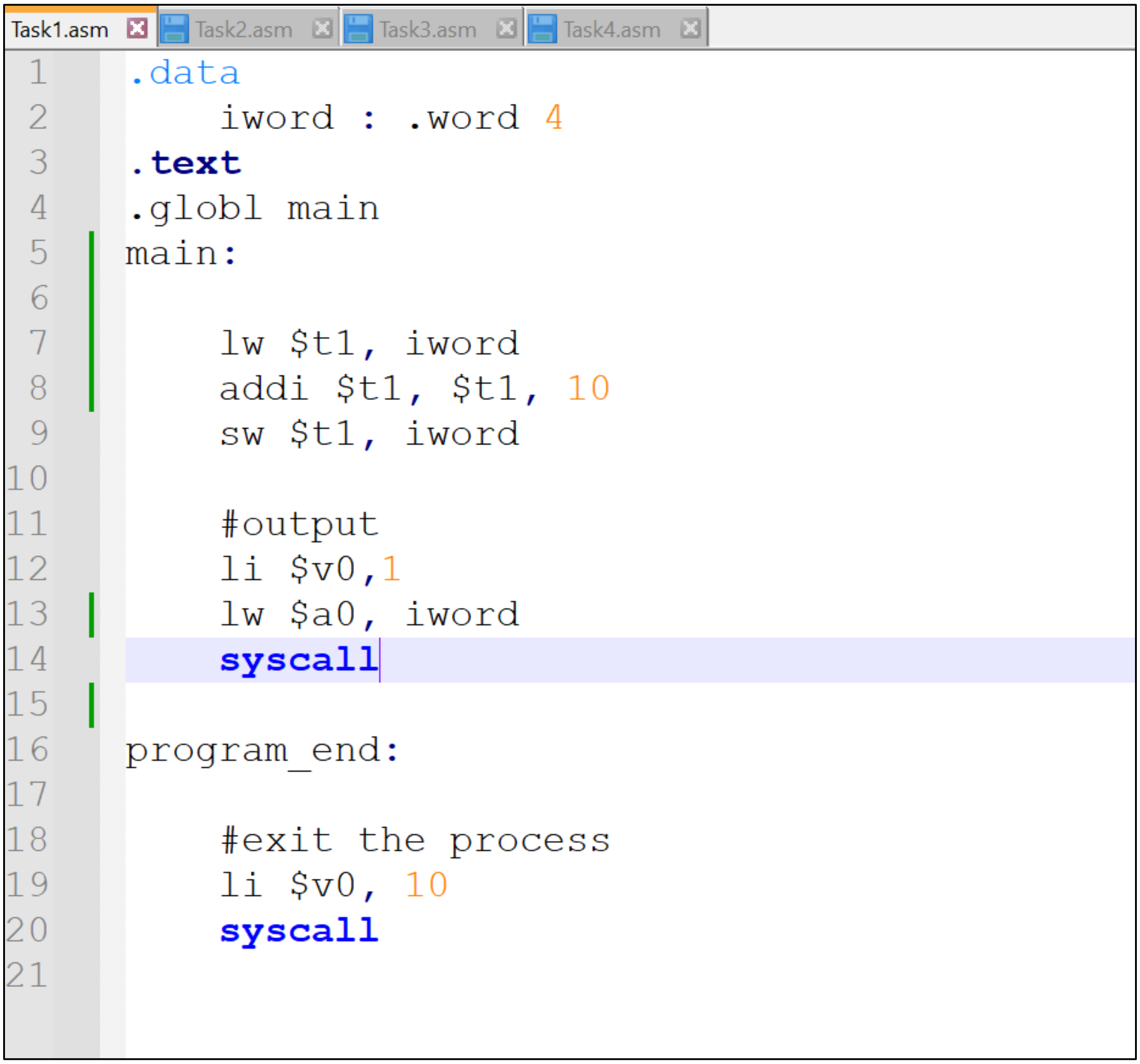
ASSESSMENT RUBRICS COA LABS

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:

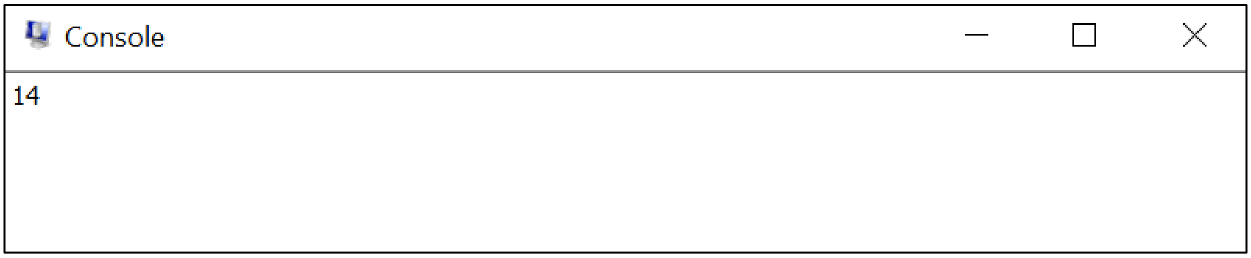
Load a value from memory and add 10 to it. Store the result back in memory and show the result on console. (*hint: use MIPS instructions lw and sw*)

Code:



```
.data
	iword : .word 4
.text
.globl main
main:

	lw $t1, iword
	addi $t1, $t1, 10
	sw $t1, iword

	#output
	li $v0,1
	lw $a0, iword
	syscall

program_end:

	#exit the process
	li $v0, 10
	syscall

```

Output:

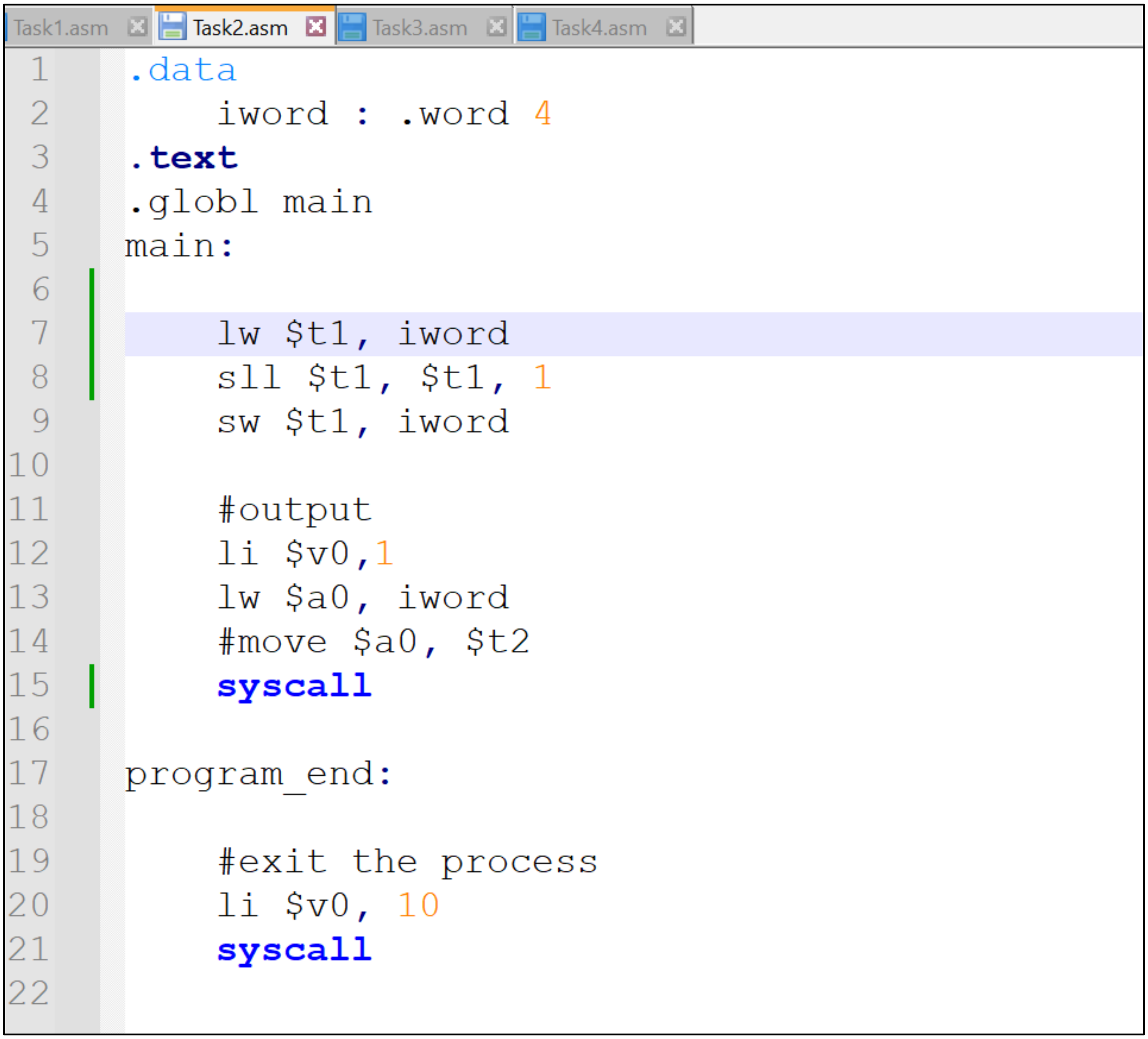


```
14
```

Task 2:

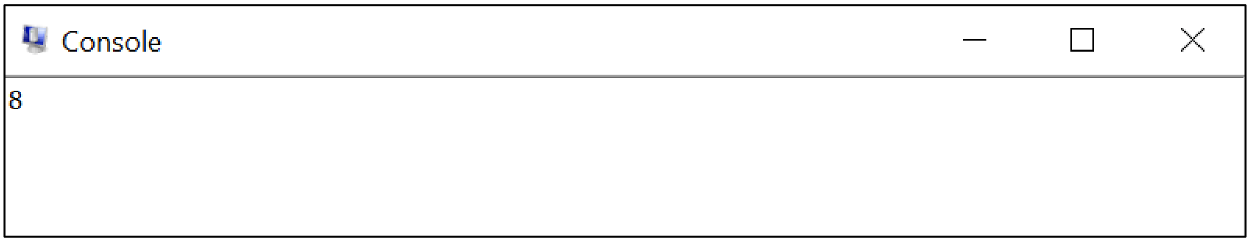
Load a value from memory and double it. Store the result back in memory also show on the console. (*use sll, sw and lw*)

Code:



```
.data
    iword : .word 4
.text
.globl main
main:

    lw $t1, iword
    sll $t1, $t1, 1
    sw $t1, iword

    #output
    li $v0,1
    lw $a0, iword
    #move $a0, $t2
    syscall

program_end:

    #exit the process
    li $v0, 10
    syscall

```

Output:

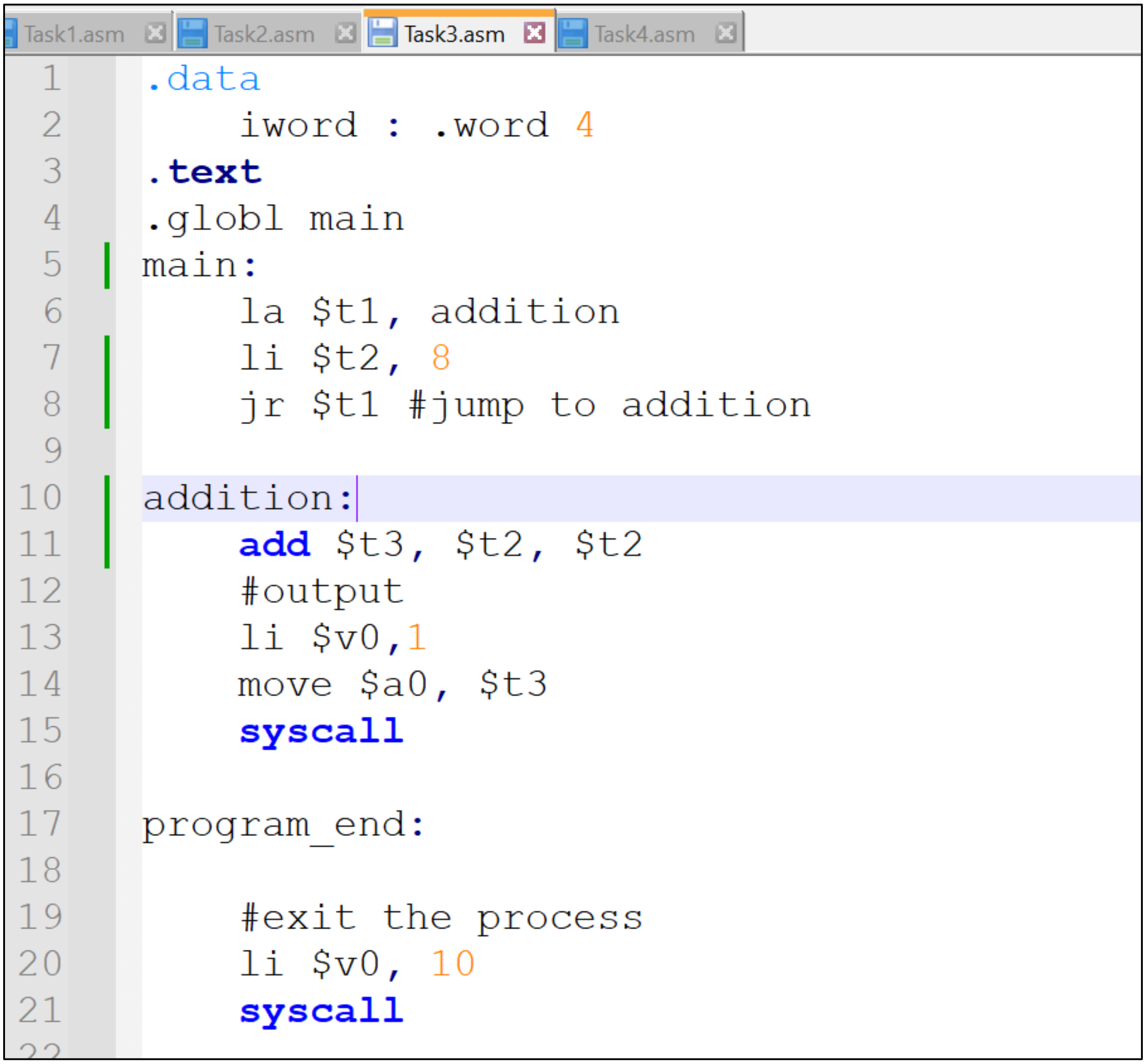


```
8
```

Task 3:

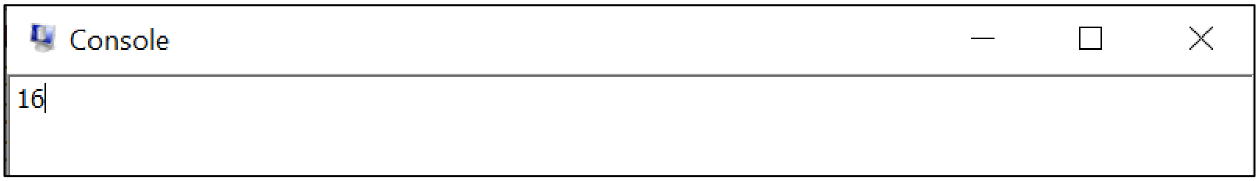
Load an address of a label into a register and jump to that address and perform addition in that address. .(use jr(jump register))

Code:



```
.data
	iword : .word 4
.text
.globl main
main:
	la $t1, addition
	li $t2, 8
	jr $t1 #jump to addition

addition:
	add $t3, $t2, $t2
	#output
	li $v0,1
	move $a0, $t3
	syscall

program_end:

	#exit the process
	li $v0, 10
	syscall
```

Output:



```
16
```

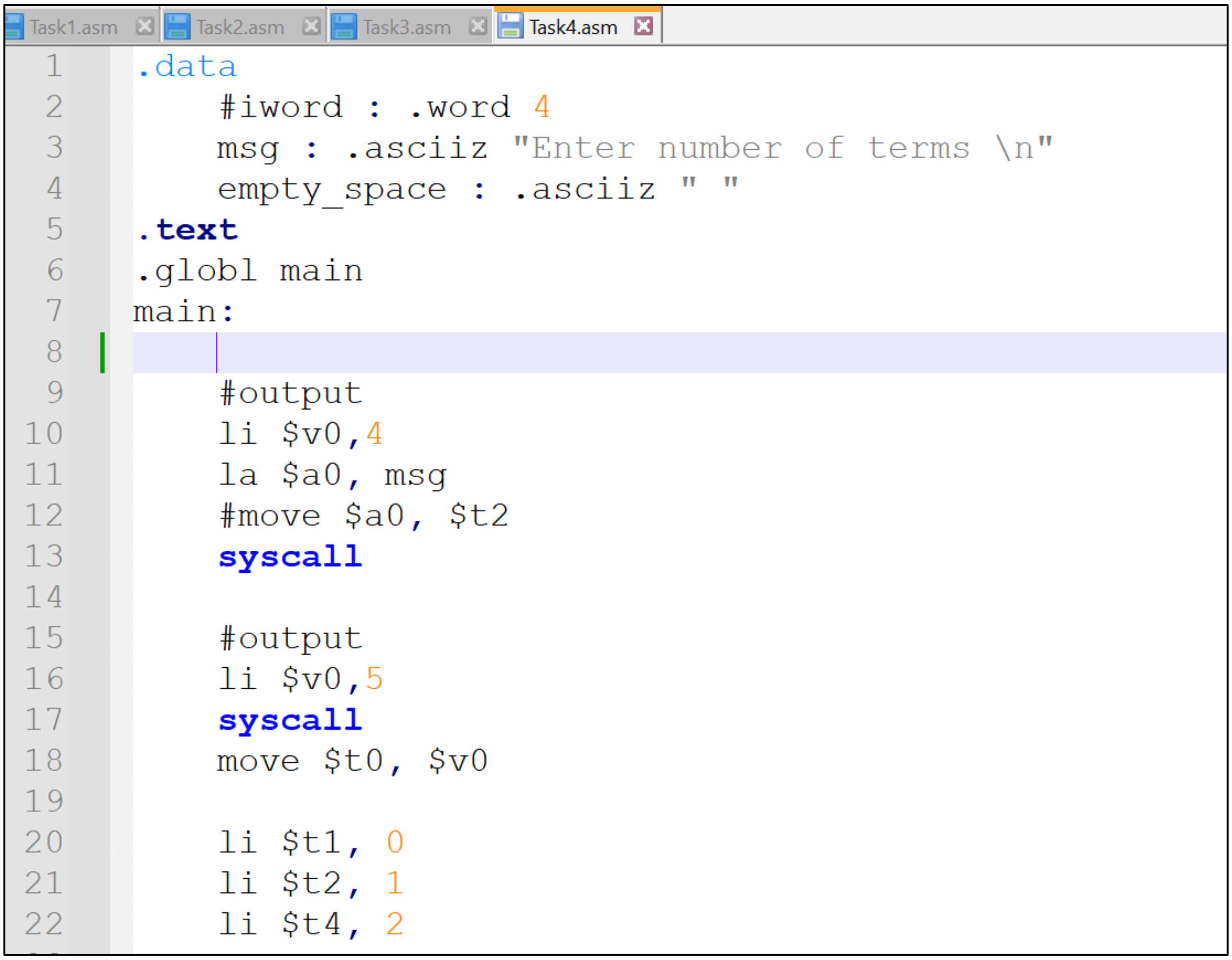
Task 4:

Write assembly program to find the Fibonacci series.

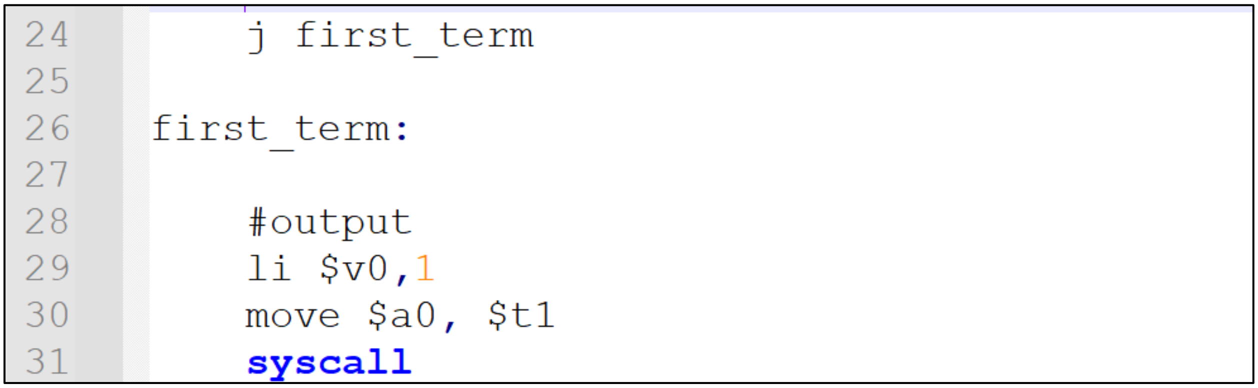
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

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

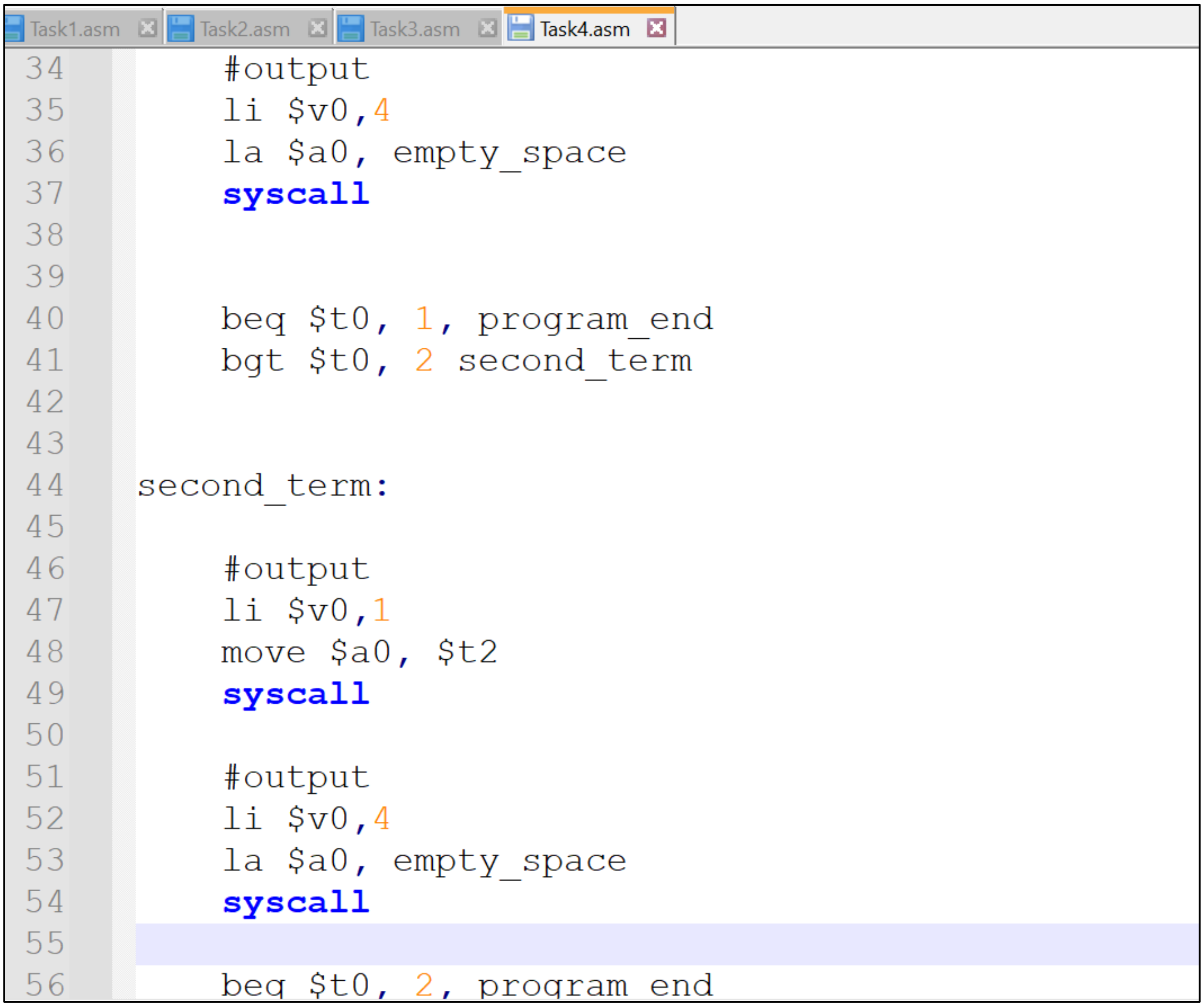
Code:



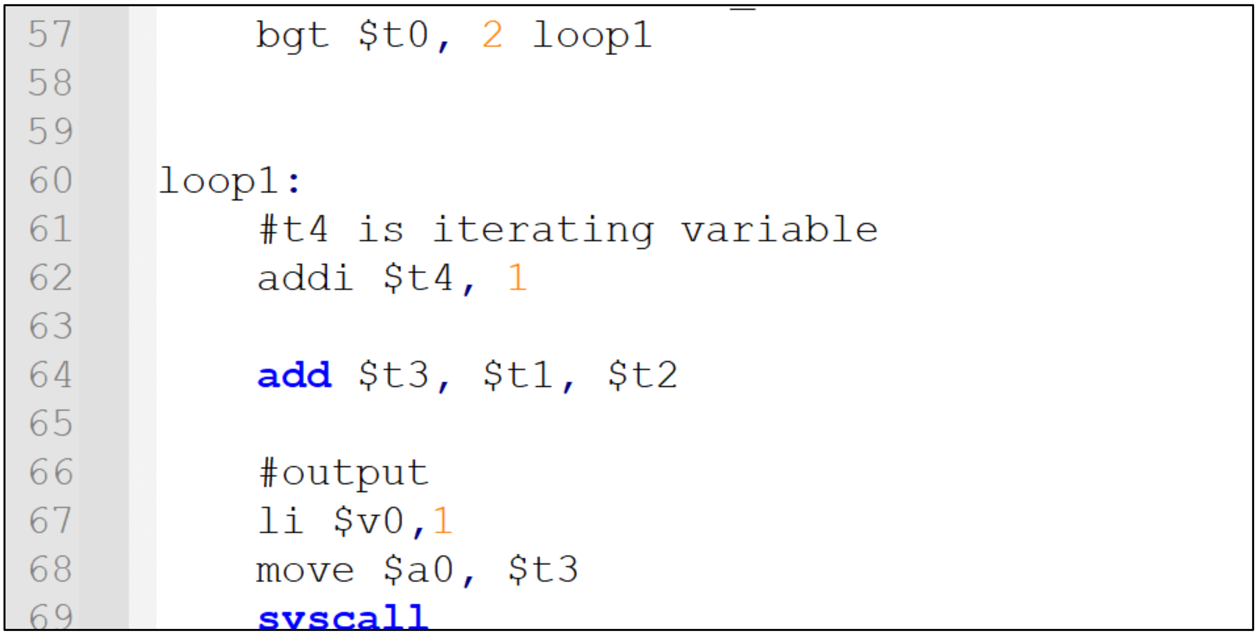
```
.data
    #iword : .word 4
    msg : .asciiz "Enter number of terms \n"
    empty_space : .asciiz " "
.text
.globl main
main:

    #output
    li $v0,4
    la $a0, msg
    #move $a0, $t2
    syscall

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

    li $t1, 0
    li $t2, 1
    li $t4, 2
```



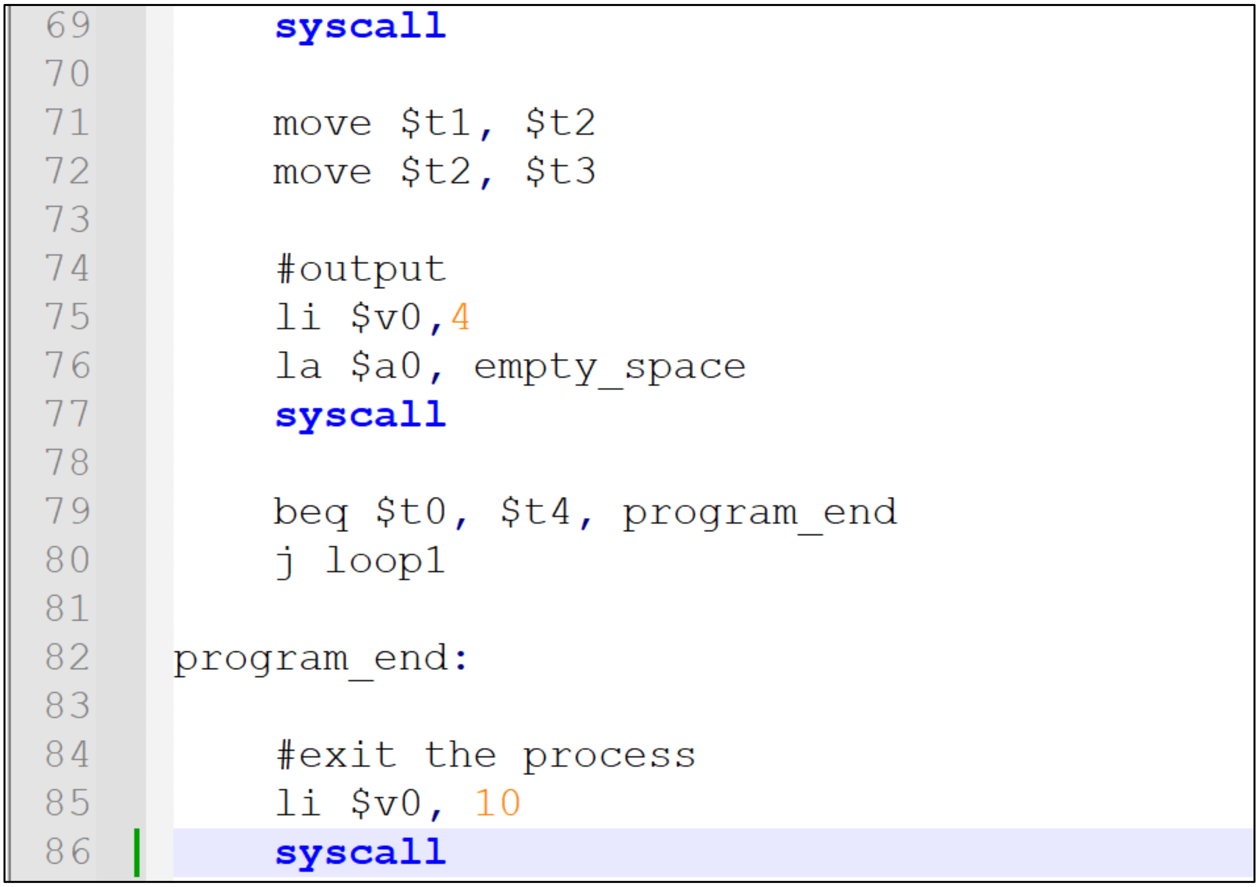
```
    j first_term

first_term:

    #output
    li $v0,1
    move $a0, $t1
    syscall
```



```
    #output
    li $v0,4
    la $a0, empty_space
    syscall


    beq $t0, 1, program_end
    bgt $t0, 2 second_term


second_term:

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

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

    beq $t0, 2, program_end
```

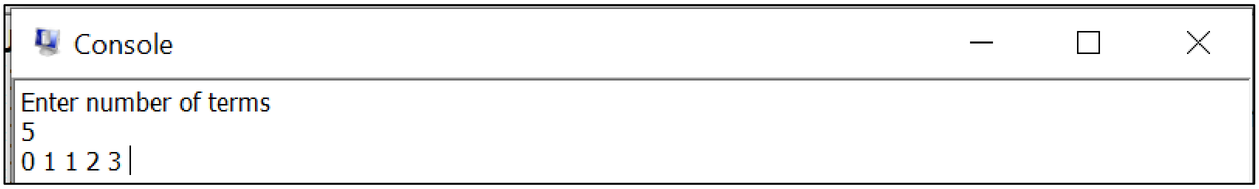


```
    bgt $t0, 2 loop1


loop1:
    #t4 is iterating variable
    addi $t4, 1

    add $t3, $t1, $t2

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

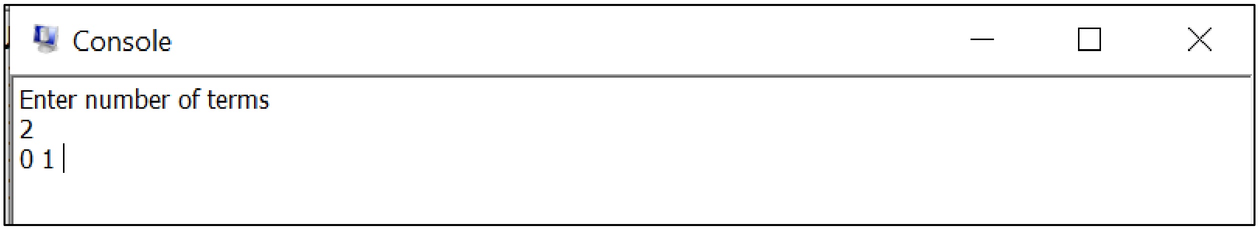


```
        syscall

        move $t1, $t2
        move $t2, $t3

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

        beq $t0, $t4, program_end
        j loop1

program_end:

        #exit the process
        li $v0, 10
        syscall
```

Output:



```
Enter number of terms
5
0 1 1 2 3
```



```
Enter number of terms
2
0 1
```

Conclusion:

In this lab, I learned about the storing and loading instructions(lw and sw) in MIPS Assembly.