ARITHMETIC AND LOGICAL OPERATIONS IN QTSPIM(ASSEMBLY LANGUAGE)

LAB # 01

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:

5th 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	

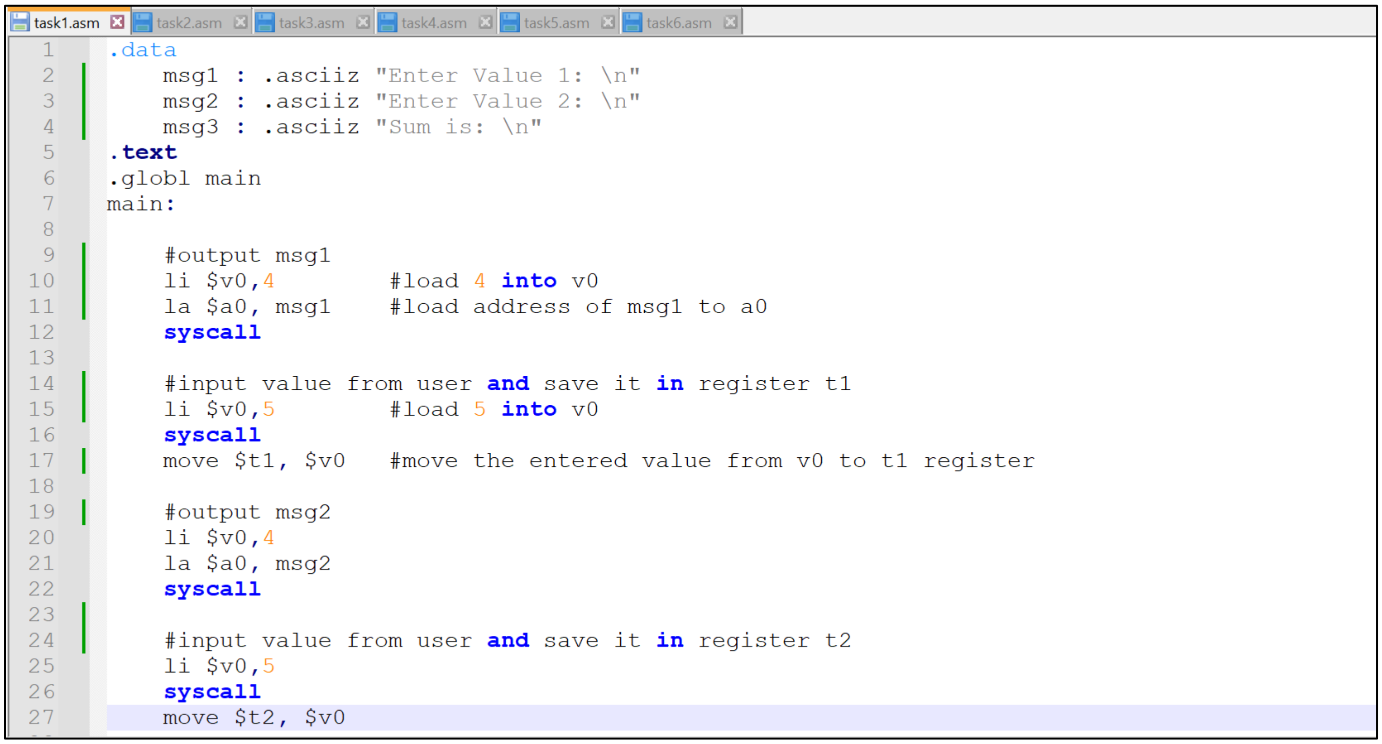
Total Marks Obtained:__________

Instructor Signature: ______________________

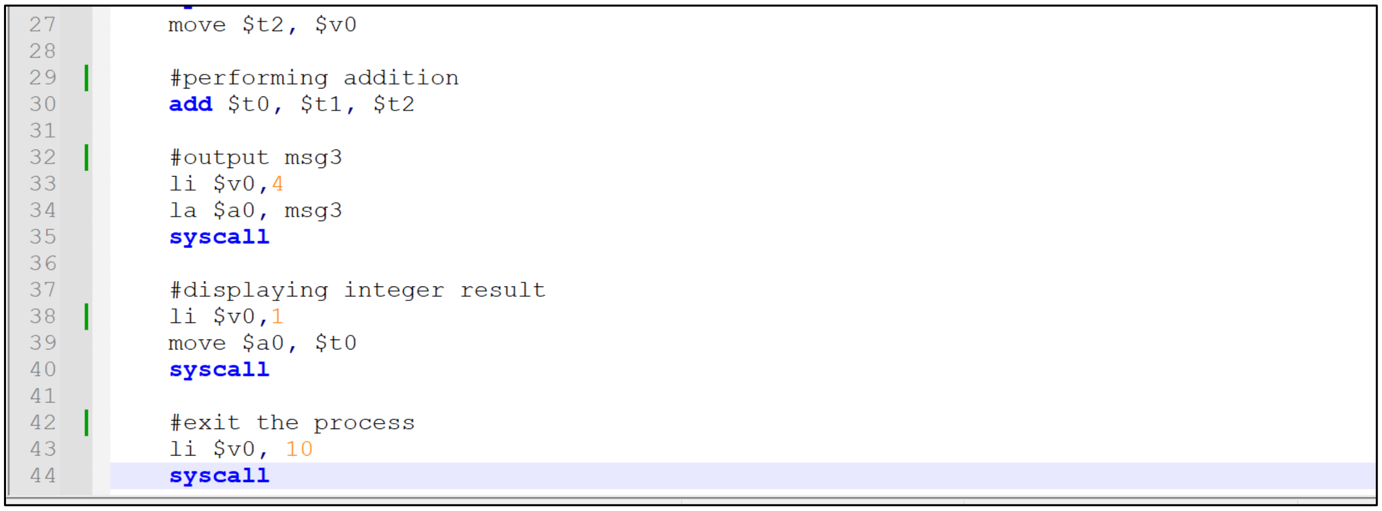
Task 1:

Write an assembly language program which takes two numbers from user and add them and show the result on console.

Code:

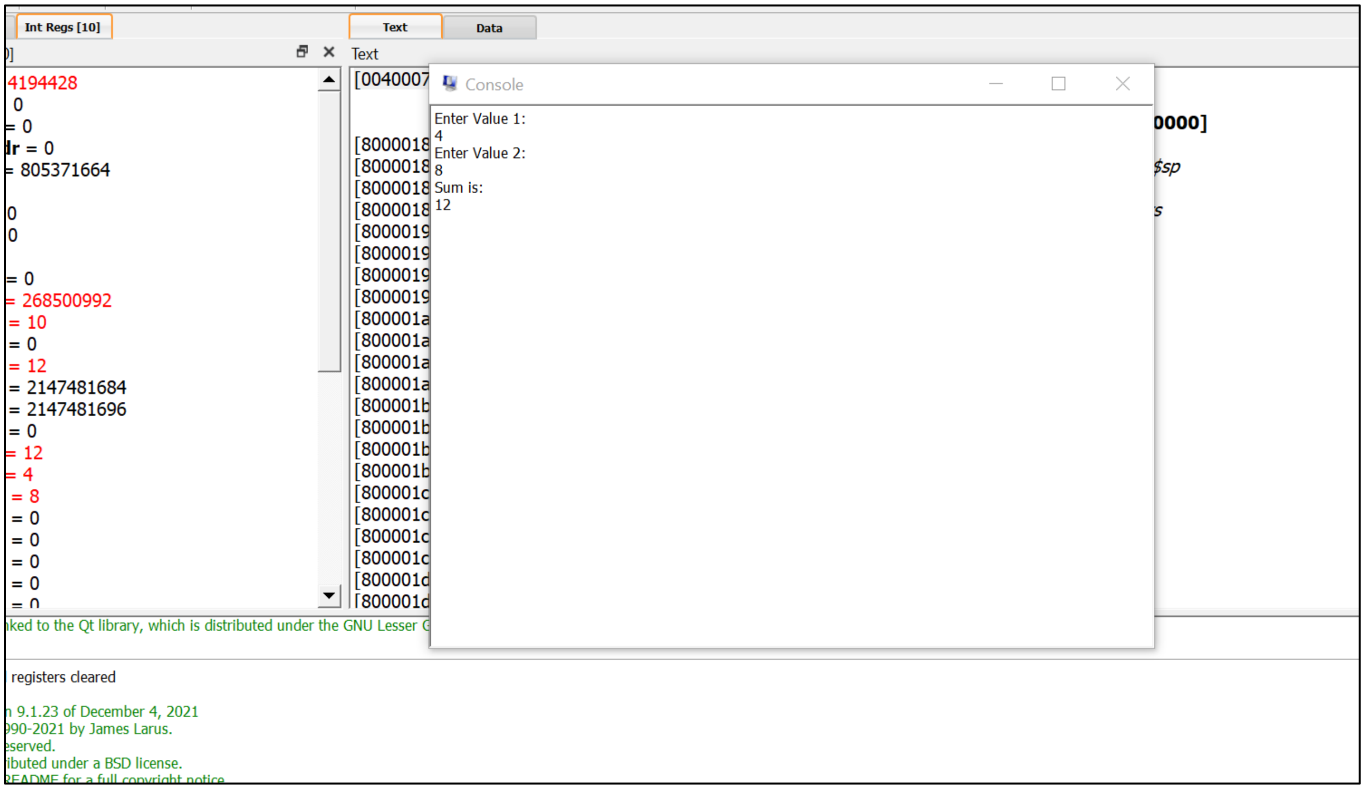


```
.data
    msg1 : .asciiz "Enter Value 1: \n"
    msg2 : .asciiz "Enter Value 2: \n"
    msg3 : .asciiz "Sum is: \n"
.text
.globl main
main:

    #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
    la $a0, msg2
    syscall

    #input value from user and save it in register t2
    li $v0,5
    syscall
    move $t2, $v0
```



```
    move $t2, $v0

    #performing addition
    add $t0, $t1, $t2

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

    #displaying integer result
    li $v0,1
    move $a0, $t0
    syscall

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

Output:



```
Int Regs [10]                                   Text    Data
]                                               Text
4194428                                         [0040007
 0                                              [8000018
= 0                                             [8000018
dr = 0                                          [8000018
= 805371664                                     [8000018
                                                [8000019
0                                               [8000019
0                                               [8000019
                                                [8000019
= 0                                             [800001a
= 268500992                                     [800001a
= 10                                            [800001a
= 0                                             [800001a
= 12                                            [800001b
= 2147481684                                    [800001b
= 2147481696                                    [800001b
= 0                                             [800001c
= 12                                            [800001c
= 4                                             [800001c
= 8                                             [800001c
= 0                                             [800001d
= 0                                             [800001d
= 0
= 0
= 0
```

```
Console
Enter Value 1:
4
Enter Value 2:
8
Sum is:
12
```

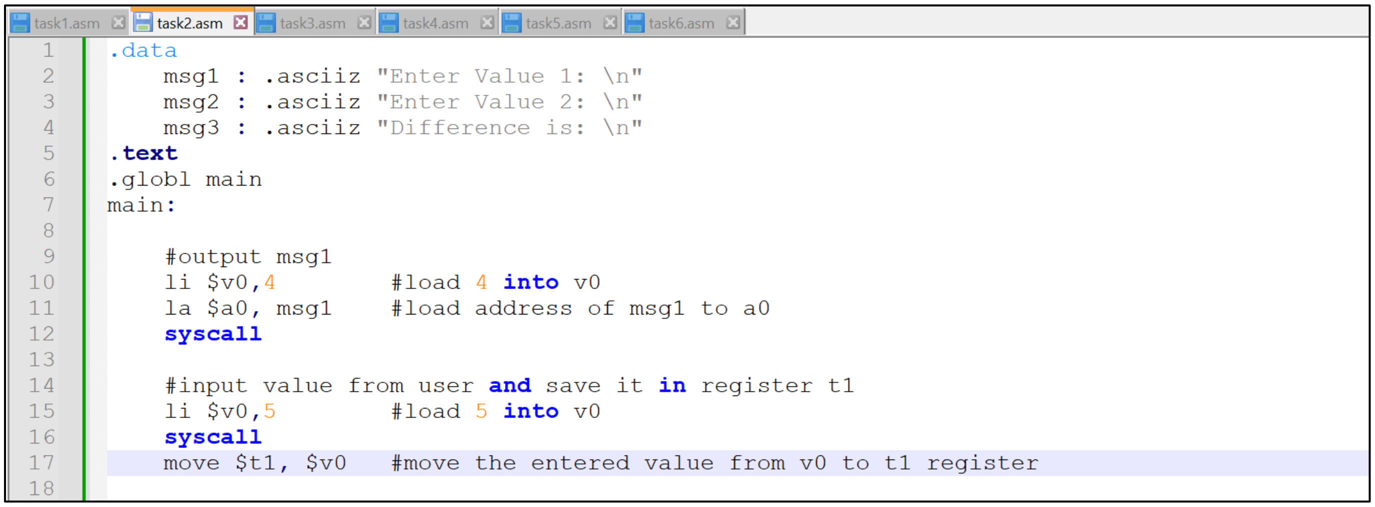
```
0000]
$sp
s
```

```
nked to the Qt library, which is distributed under the GNU Lesser G

registers cleared

n 9.1.23 of December 4, 2021
990-2021 by James Larus.
eserved.
ibuted under a BSD license.
README for a full copyright notice
```

Task 2:

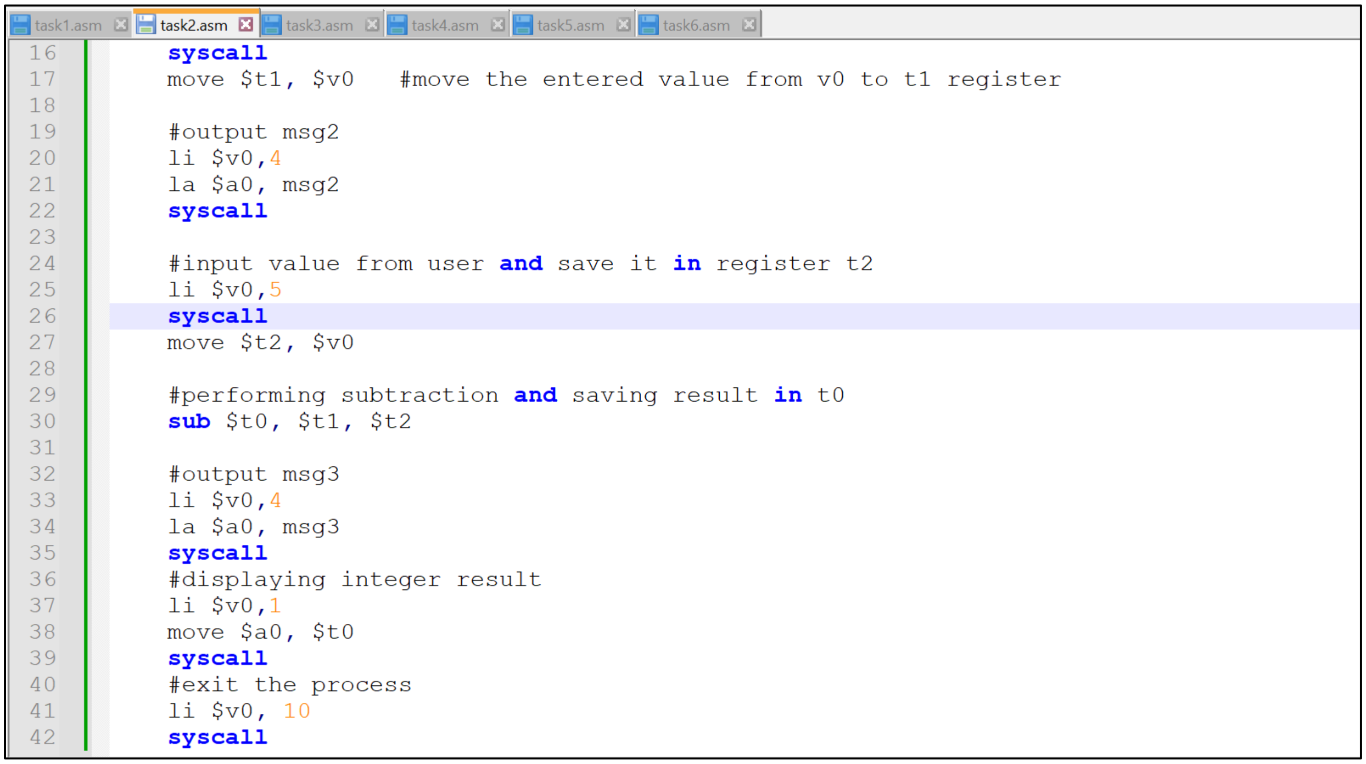
Write an assembly language program which takes two numbers from user and subtract them and show the result on console.

Code:



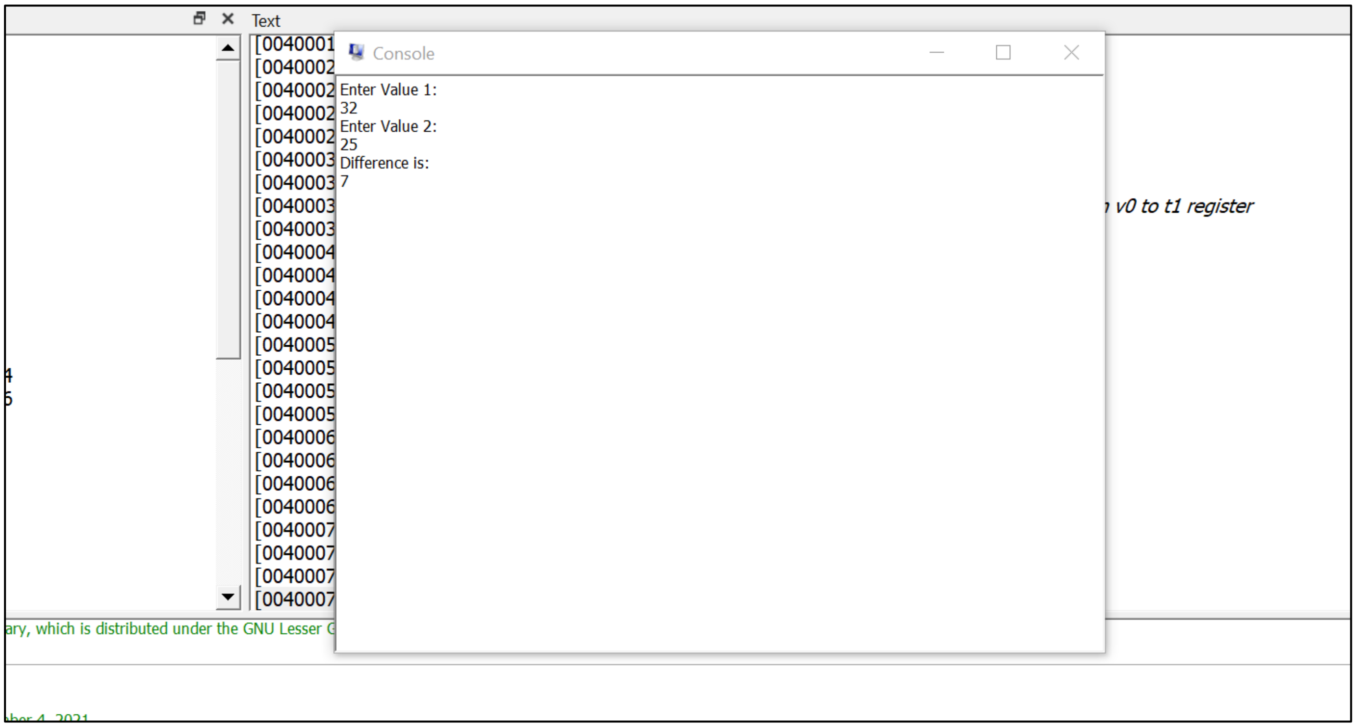
task1.asm task2.asm task3.asm task4.asm task5.asm task6.asm

```
.data
    msg1 : .asciiz "Enter Value 1: \n"
    msg2 : .asciiz "Enter Value 2: \n"
    msg3 : .asciiz "Difference is: \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

```



```
    syscall
    move $t1, $v0   #move the entered value from v0 to t1 register

    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

    #input value from user and save it in register t2
    li $v0,5
    syscall
    move $t2, $v0

    #performing subtraction and saving result in t0
    sub $t0, $t1, $t2

    #output msg3
    li $v0,4
    la $a0, msg3
    syscall
    #displaying integer result
    li $v0,1
    move $a0, $t0
    syscall
    #exit the process
    li $v0, 10
    syscall
```

Output:

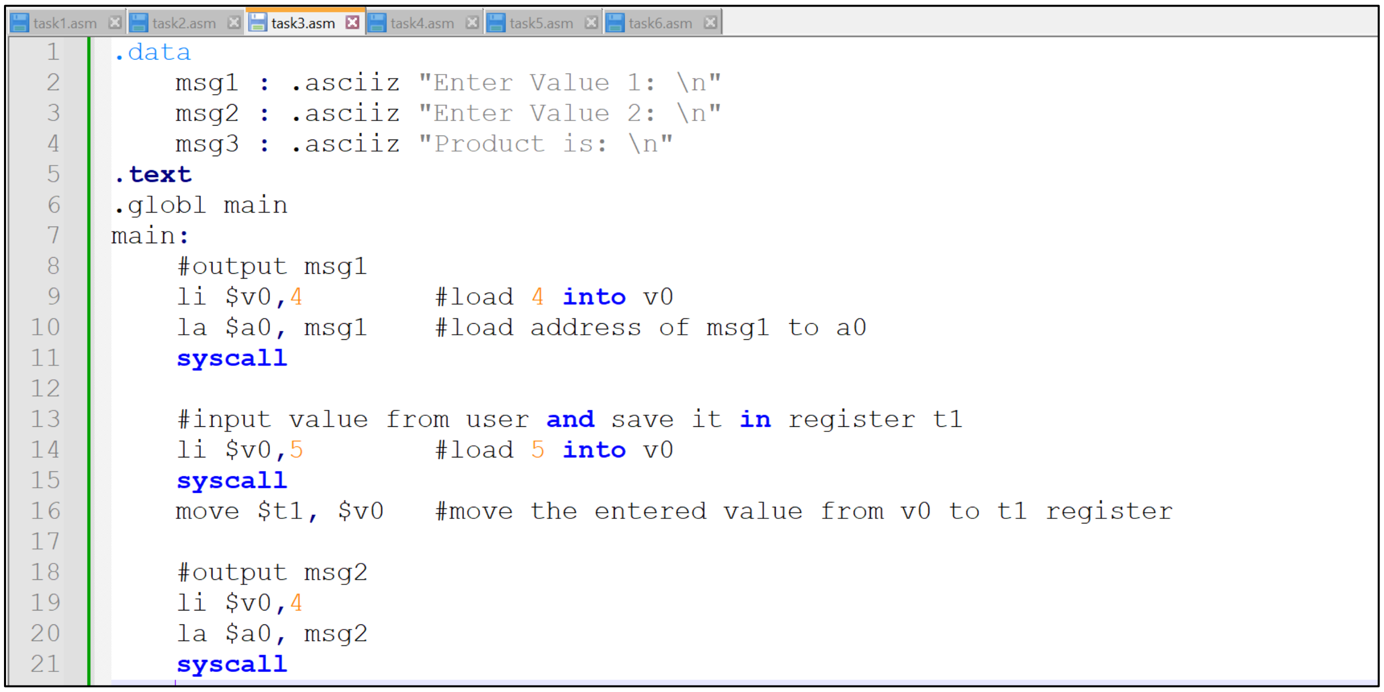


```
Enter Value 1:
32
Enter Value 2:
25
Difference is:
7
```

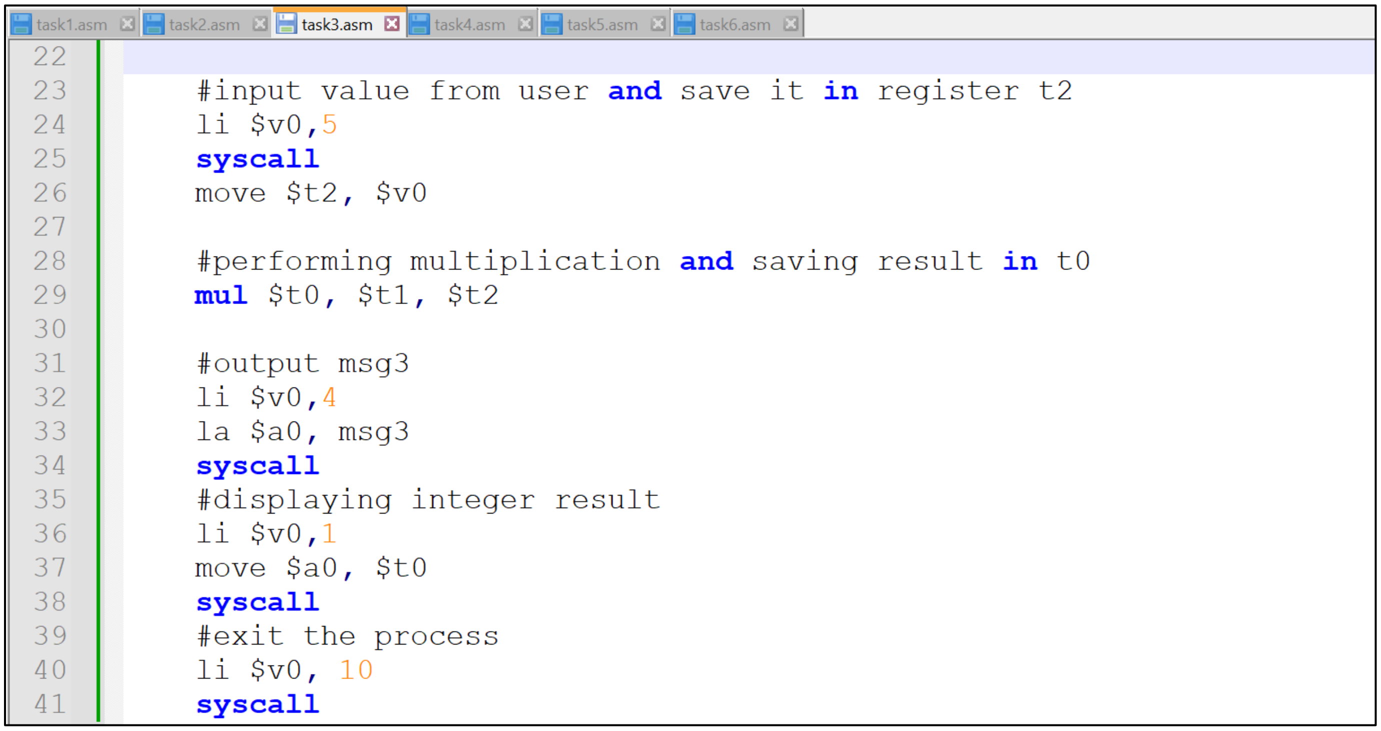
Task 3:

Write an assembly language program which takes two numbers from user and multiply them and show the result on console.

Code:

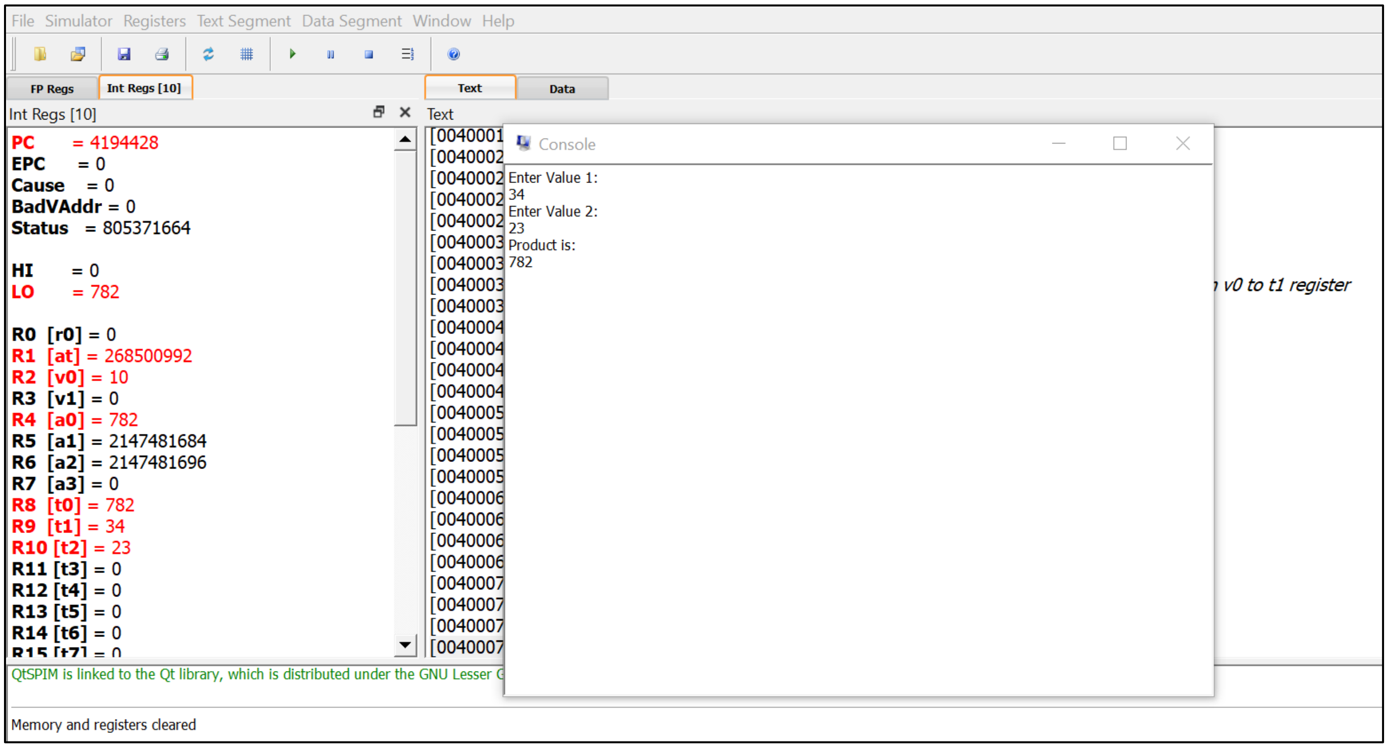


```
.data
    msg1 : .asciiz "Enter Value 1: \n"
    msg2 : .asciiz "Enter Value 2: \n"
    msg3 : .asciiz "Product is: \n"
.text
.globl main
main:
    #output msg1
    li $v0,4        #load 4 into v0
    la $a0, msg1    #load address of msg1 to a0
    syscall

    #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
    la $a0, msg2
    syscall
```



```

    #input value from user and save it in register t2
    li $v0,5
    syscall
    move $t2, $v0

    #performing multiplication and saving result in t0
    mul $t0, $t1, $t2

    #output msg3
    li $v0,4
    la $a0, msg3
    syscall
    #displaying integer result
    li $v0,1
    move $a0, $t0
    syscall
    #exit the process
    li $v0, 10
    syscall
```

Output:


File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text
Data
Console

```
Int Regs [10]
PC      = 4194428
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 805371664

HI      = 0
LO      = 782

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 782
R5  [a1] = 2147481684
R6  [a2] = 2147481696
R7  [a3] = 0
R8  [t0] = 782
R9  [t1] = 34
R10 [t2] = 23
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
```

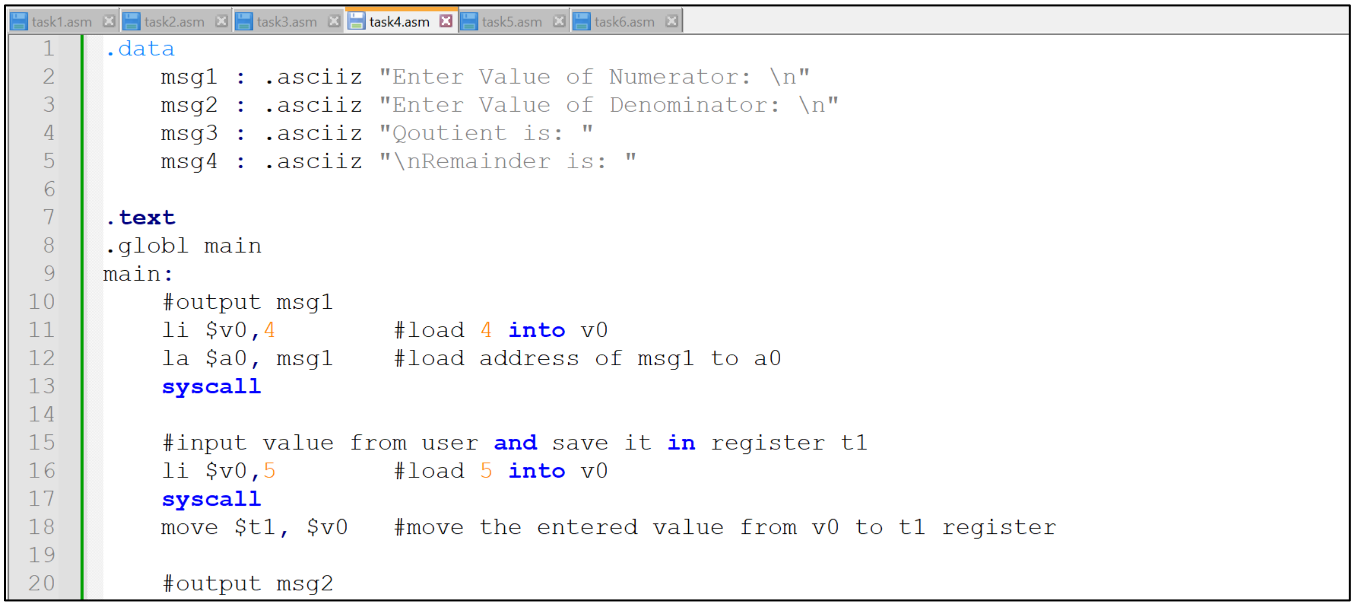
```
Enter Value 1:
34
Enter Value 2:
23
Product is:
782
```

```
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser G
Memory and registers cleared
```

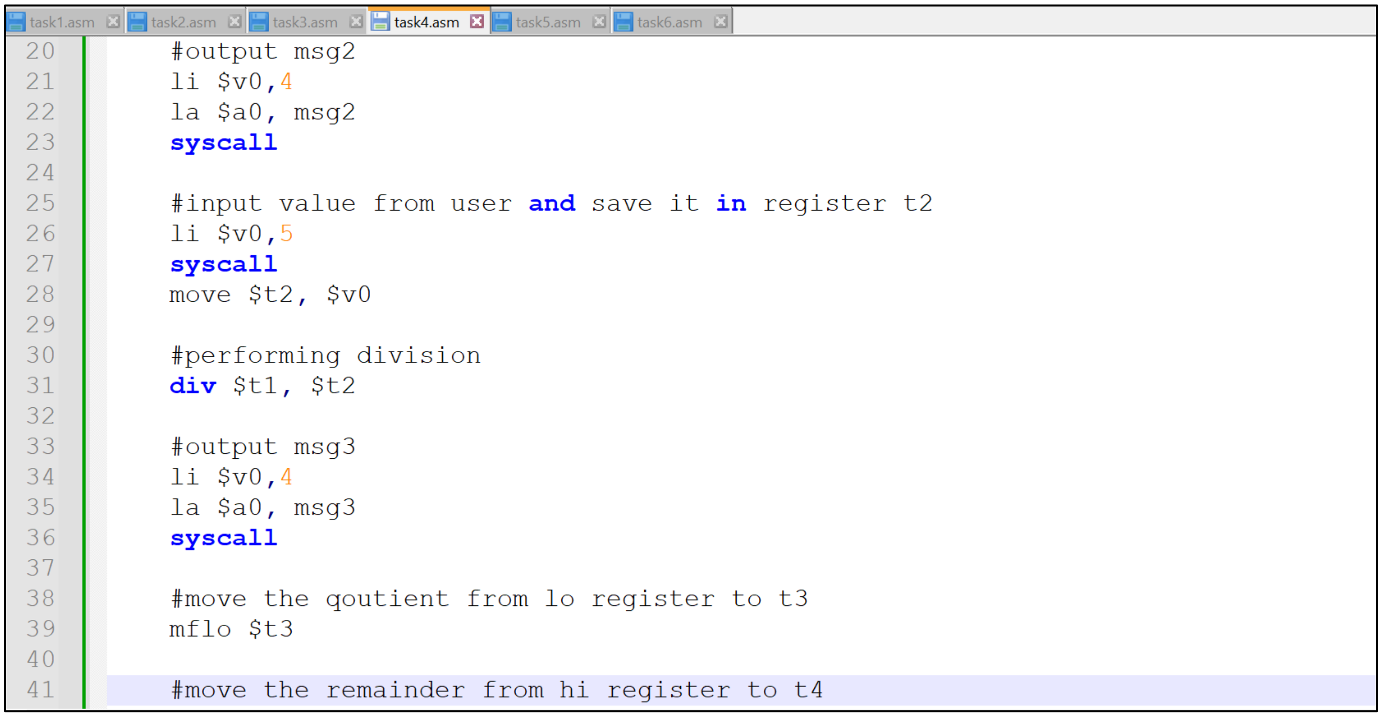
Task 4:

Write an assembly language program which takes two numbers from user and divide them and show the result on console.

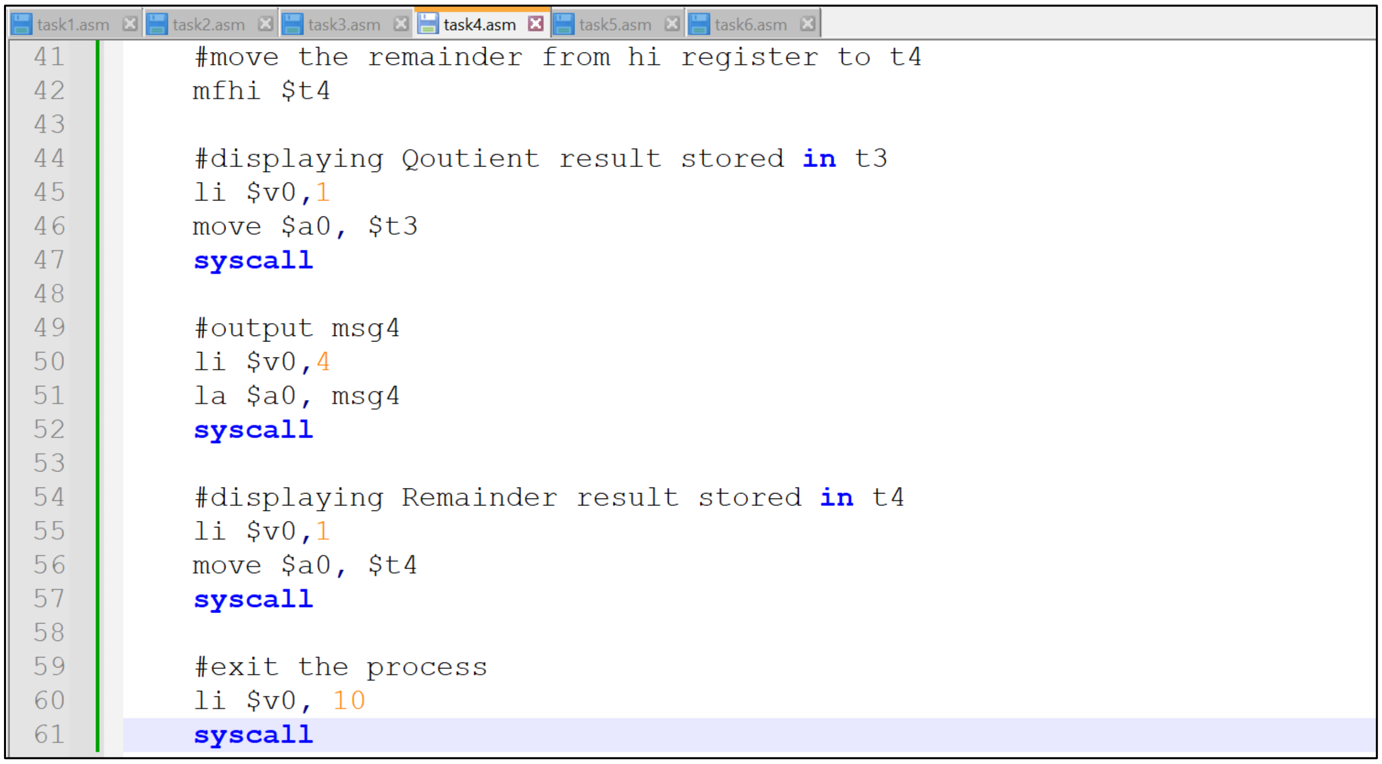
Code:



```
.data
    msg1 : .asciiz "Enter Value of Numerator: \n"
    msg2 : .asciiz "Enter Value of Denominator: \n"
    msg3 : .asciiz "Qoutient is: "
    msg4 : .asciiz "\nRemainder is: "

.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
```

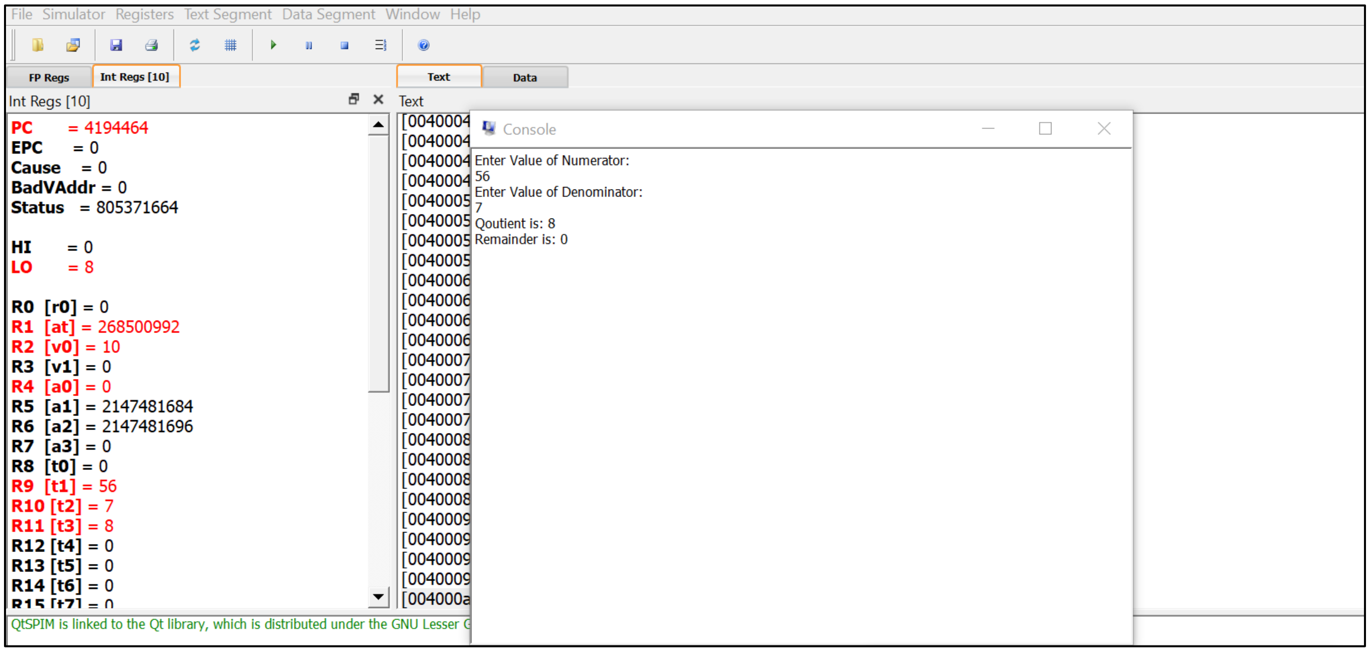

task1.asm task2.asm task3.asm task4.asm task5.asm task6.asm

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

	#input value from user and save it in register t2
	li $v0,5
	syscall
	move $t2, $v0

	#performing division
	div $t1, $t2

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

	#move the qoutient from lo register to t3
	mflo $t3

	#move the remainder from hi register to t4
```


task1.asm task2.asm task3.asm task4.asm task5.asm task6.asm

```
	#move the remainder from hi register to t4
	mfhi $t4

	#displaying Qoutient result stored in t3
	li $v0,1
	move $a0, $t3
	syscall

	#output msg4
	li $v0,4
	la $a0, msg4
	syscall

	#displaying Remainder result stored in t4
	li $v0,1
	move $a0, $t4
	syscall

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

Output:


File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text
Data
Int Regs [10]
Text
Console

```
PC      = 4194464
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 805371664

HI      = 0
LO      = 8

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 2147481684
R6  [a2] = 2147481696
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 56
R10 [t2] = 7
R11 [t3] = 8
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
```

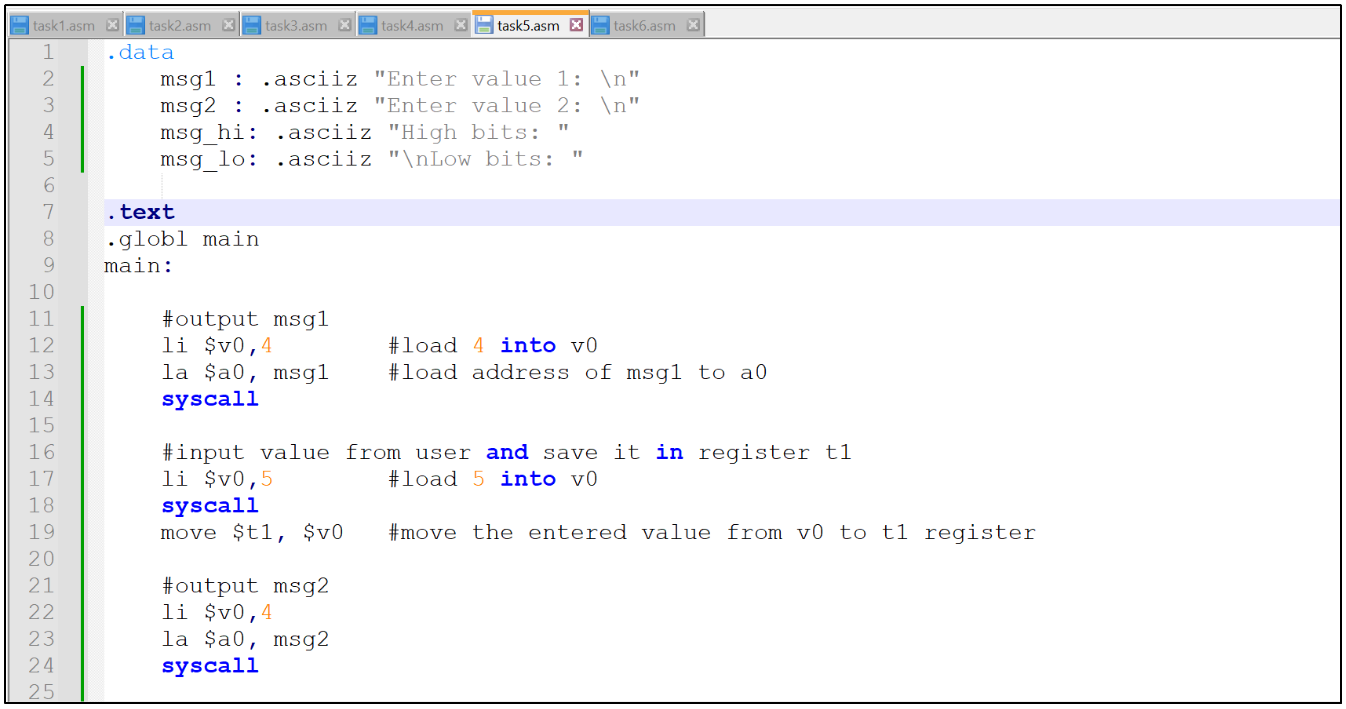
```
Enter Value of Numerator:
56
Enter Value of Denominator:
7
Qoutient is: 8
Remainder is: 0
```

QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser G

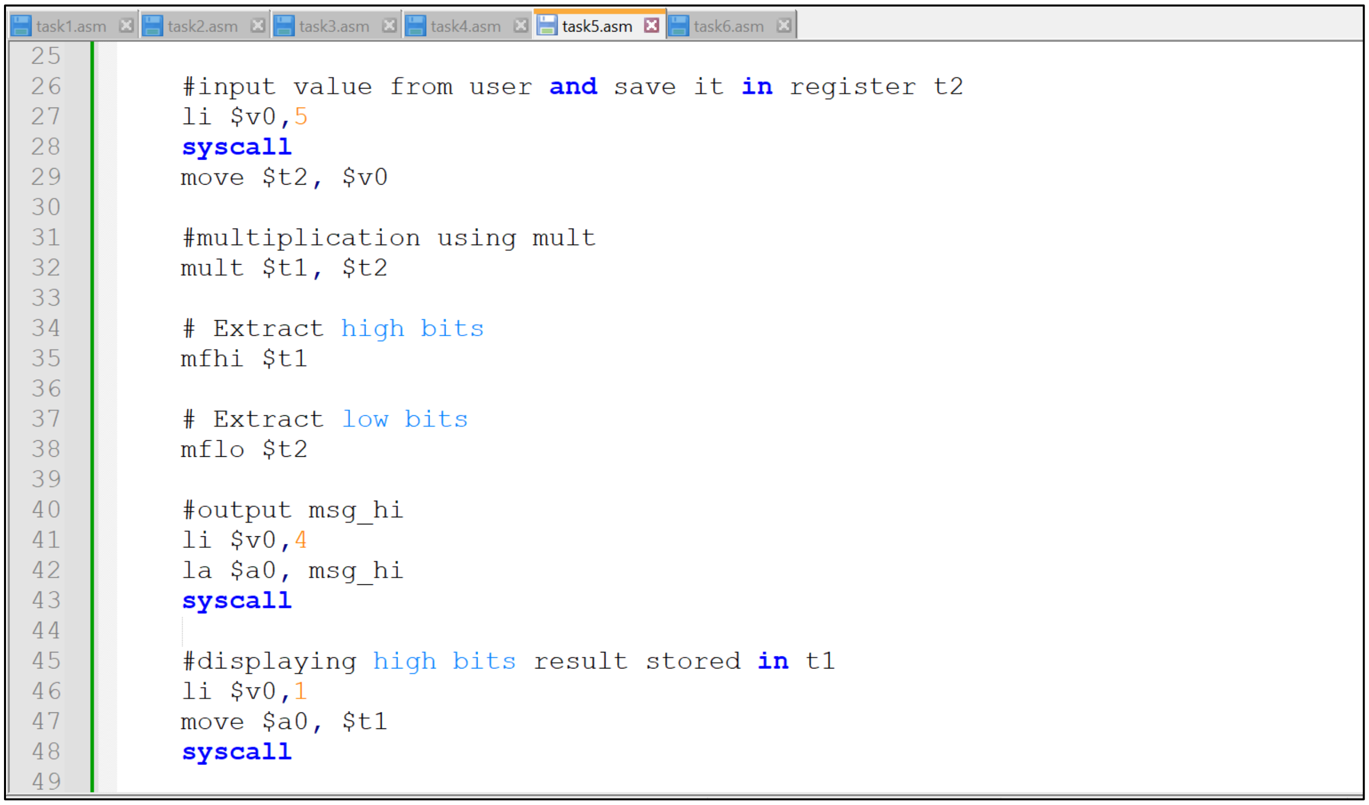
Task 5:

Write assembly program to multiply two numbers using MULT and extract the bit from high and low registers to general purpose registers.

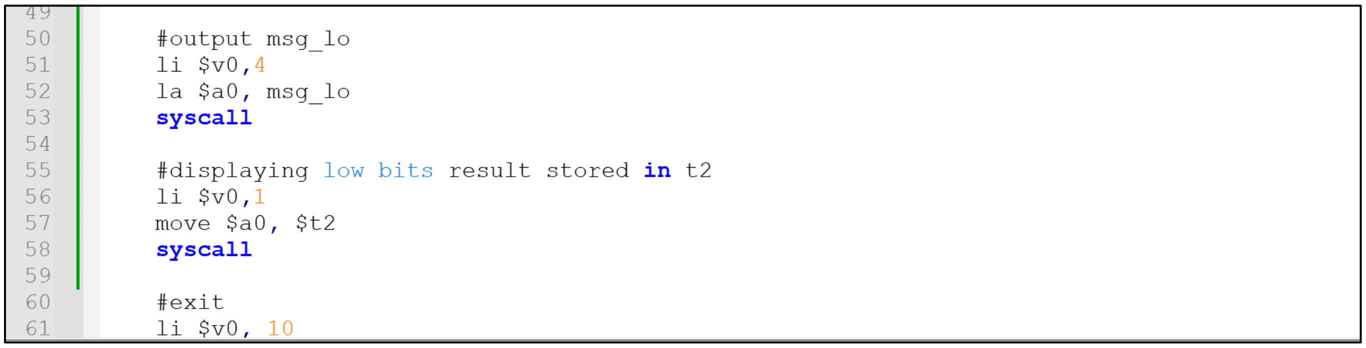
Code:



```
.data
    msg1 : .asciiz "Enter value 1: \n"
    msg2 : .asciiz "Enter value 2: \n"
    msg_hi: .asciiz "High bits: "
    msg_lo: .asciiz "\nLow bits: "

.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
    la $a0, msg2
    syscall

```

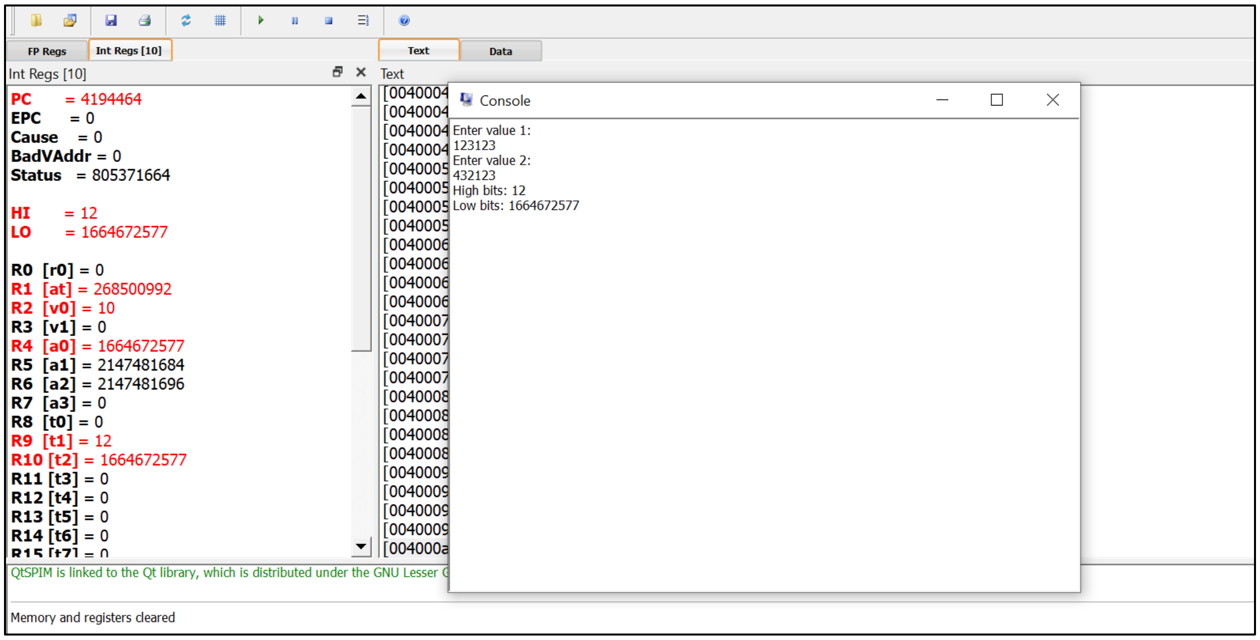


```
    #input value from user and save it in register t2
    li $v0,5
    syscall
    move $t2, $v0

    #multiplication using mult
    mult $t1, $t2

    # Extract high bits
    mfhi $t1

    # Extract low bits
    mflo $t2

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

    #displaying high bits result stored in t1
    li $v0,1
    move $a0, $t1
    syscall

```



```

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

    #displaying low bits result stored in t2
    li $v0,1
    move $a0, $t2
    syscall

    #exit
    li $v0, 10
```

Output:



```
PC      = 4194464
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 805371664

HI      = 12
LO      = 1664672577

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1664672577
R5  [a1] = 2147481684
R6  [a2] = 2147481696
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 12
R10 [t2] = 1664672577
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
```

```
Enter value 1:
123123
Enter value 2:
432123
High bits: 12
Low bits: 1664672577
```

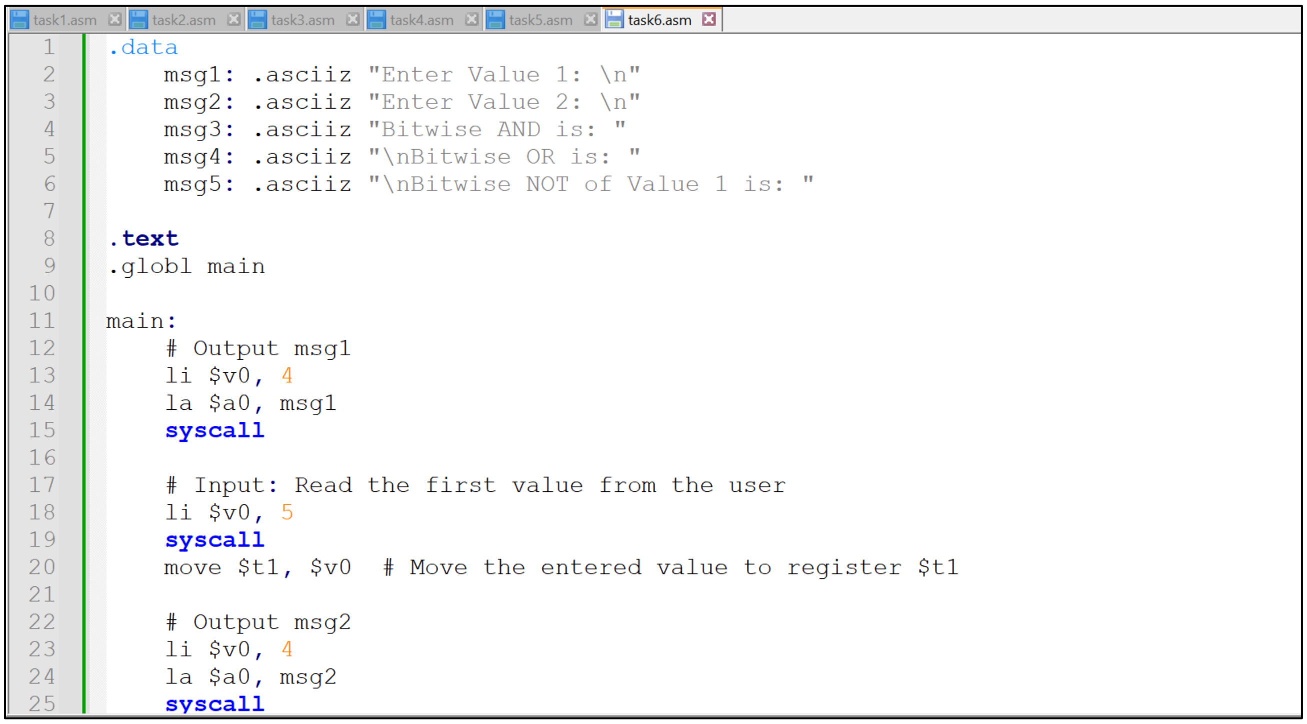
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser G

Memory and registers cleared

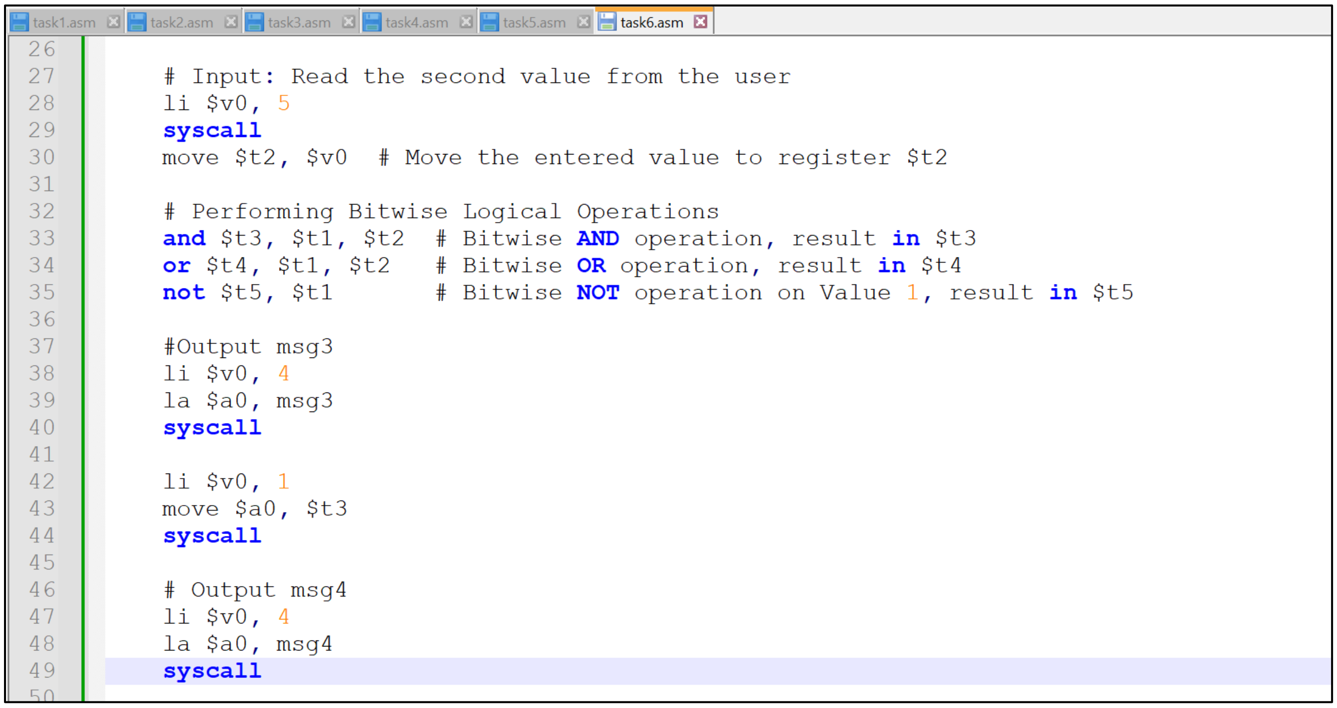
Task 6:

Write program to perform AND, OR , NOT operations in MIPS.

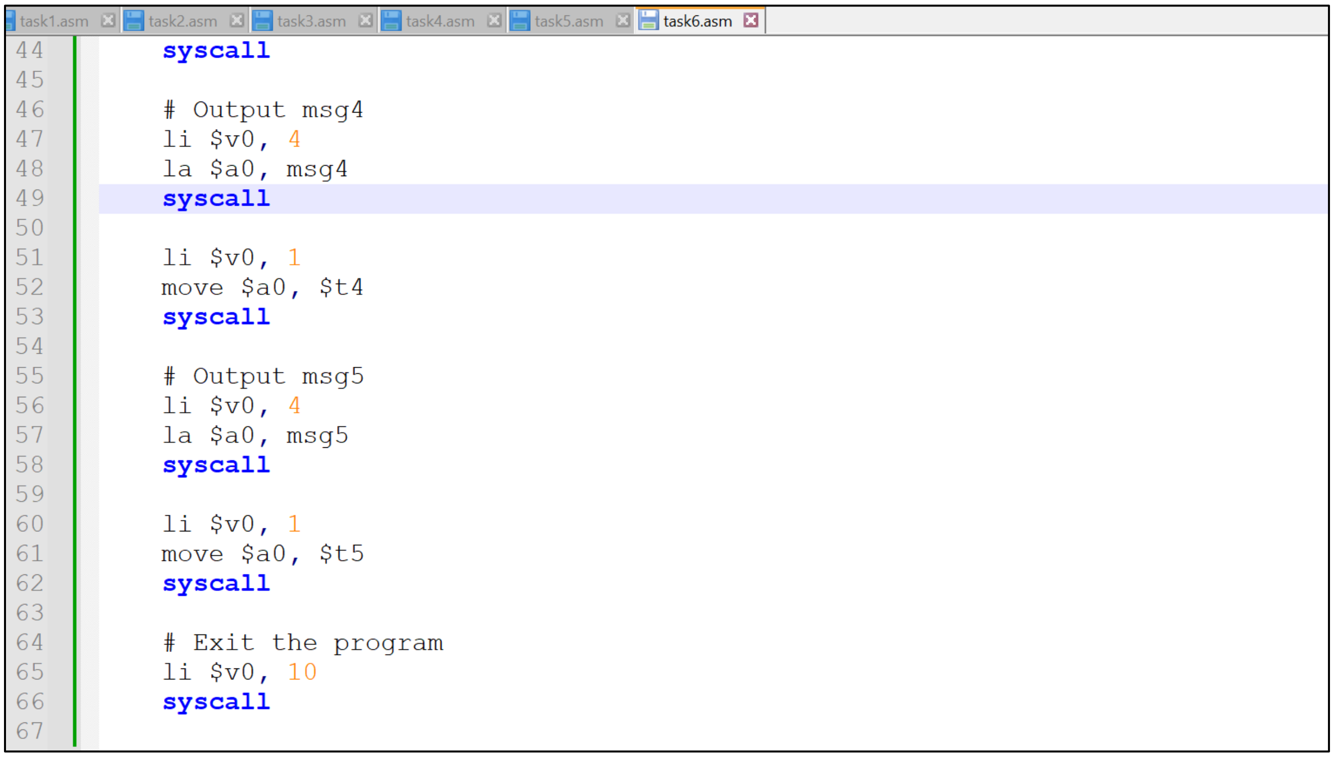
Code:



```
.data
    msg1: .asciiz "Enter Value 1: \n"
    msg2: .asciiz "Enter Value 2: \n"
    msg3: .asciiz "Bitwise AND is: "
    msg4: .asciiz "\nBitwise OR is: "
    msg5: .asciiz "\nBitwise NOT of Value 1 is: "

.text
.globl main

main:
    # Output msg1
    li $v0, 4
    la $a0, msg1
    syscall

    # Input: Read the first value from the user
    li $v0, 5
    syscall
    move $t1, $v0  # Move the entered value to register $t1

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

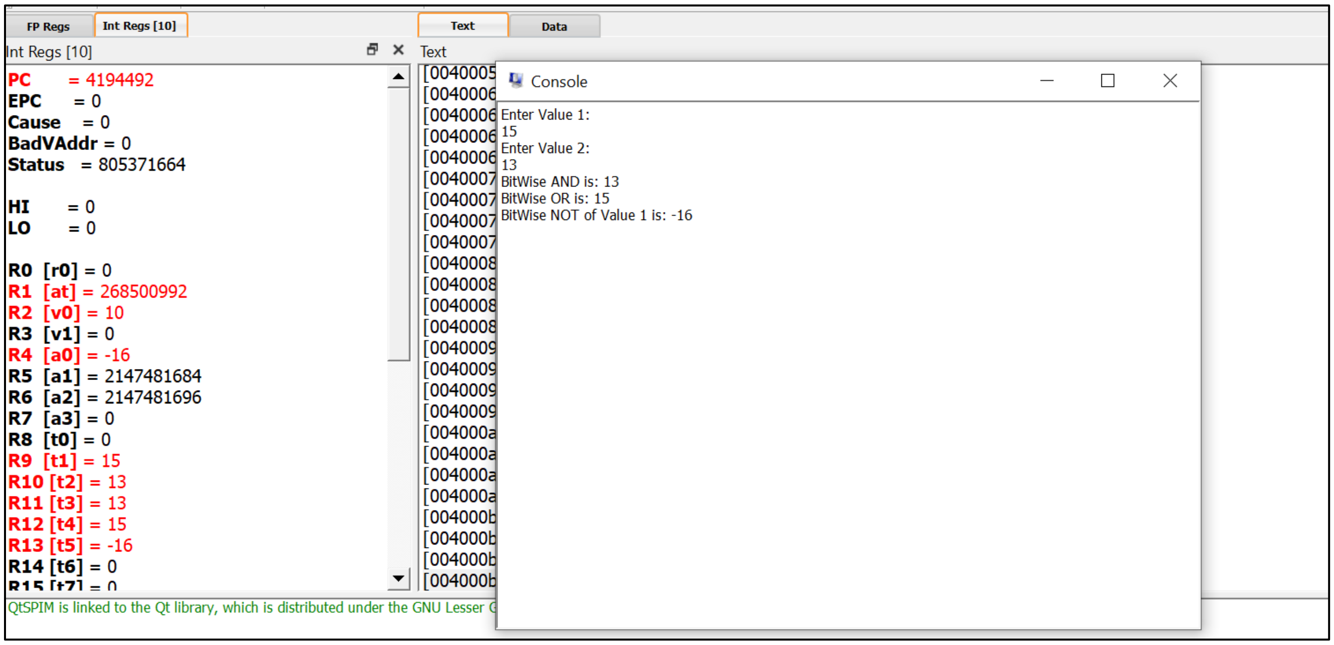


```

    # Input: Read the second value from the user
    li $v0, 5
    syscall
    move $t2, $v0  # Move the entered value to register $t2

    # Performing Bitwise Logical Operations
    and $t3, $t1, $t2  # Bitwise AND operation, result in $t3
    or $t4, $t1, $t2   # Bitwise OR operation, result in $t4
    not $t5, $t1       # Bitwise NOT operation on Value 1, result in $t5

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

    li $v0, 1
    move $a0, $t3
    syscall

    # Output msg4
    li $v0, 4
    la $a0, msg4
    syscall
```



```
    syscall

    # Output msg4
    li $v0, 4
    la $a0, msg4
    syscall

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

    # Output msg5
    li $v0, 4
    la $a0, msg5
    syscall

    li $v0, 1
    move $a0, $t5
    syscall

    # Exit the program
    li $v0, 10
    syscall

```

Output:



```
PC      = 4194492
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 805371664

HI      = 0
LO      = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = -16
R5  [a1] = 2147481684
R6  [a2] = 2147481696
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 15
R10 [t2] = 13
R11 [t3] = 13
R12 [t4] = 15
R13 [t5] = -16
R14 [t6] = 0
R15 [t7] = 0
```

```
[0040005
[0040006
[0040006
[0040006
[0040006
[0040007
[0040007
[0040007
[0040007
[0040008
[0040008
[0040008
[0040008
[0040009
[0040009
[0040009
[0040009
[004000a
[004000a
[004000a
[004000a
[004000b
[004000b
[004000b
[004000b
```

```
Enter Value 1:
15
Enter Value 2:
13
BitWise AND is: 13
BitWise OR is: 15
BitWise NOT of Value 1 is: -16
```

QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser G

Conclusion:

In this lab, I explored basic arithmetic operations (addition, subtraction, multiplication, and division) and logical operations (AND, OR, NOT) in MIPS assembly language. The programs were designed to take input from the user, perform the specified operations, and display the results on the console.