**Basic Commands of Assembly Language**

**LAB # 03**

****

**Fall 2021**

**CSE304L Computer Organization & Architecture**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Prof: Ammad khalil**

November 29, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Task #01:**

**Source code:**

.data

str1: .asciiz "Enter First No: "

str2: .asciiz "Please! Enter Second No: "

str3: .asciiz "Sum of First and Second No is : "

.text

main:

li $v0,4 #when v0 contain 4 it will system call for string display

la $a0,str1 #la mean load address we load address of string 1

syscall #system call for displaying string 1

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t1,$v0

add $t2,$t0,$t1

li $v0,4

la $a0,str3

syscall

li $v0,1

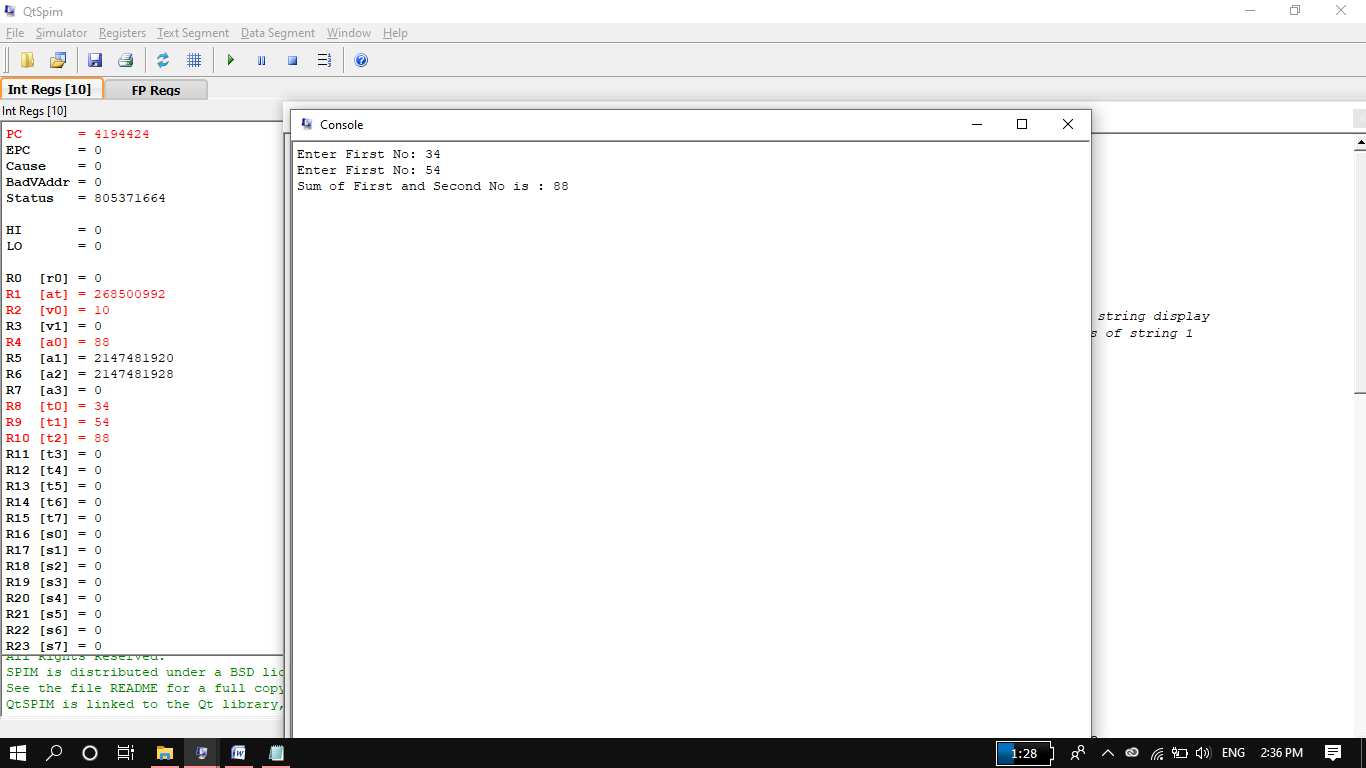
move $a0,$t2

syscall

li $v0,10

syscall

**Output:**



**Task#02:**

**Source Code:**

.data

str1: .asciiz "Please! Enter Num1: " #z for null character. if we remove z then it will print all strings till next z.

str2: .asciiz "Please! Enter Num2: "

str3: .asciiz "Answer is: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

bgt $t0,$t1 additionlabel

sub $t2,$t1,$t0

j exit

additionlabel: add $t2,$t1,$t0

exit:

li $v0,4

la $a0,str3

syscall

li $v0,1

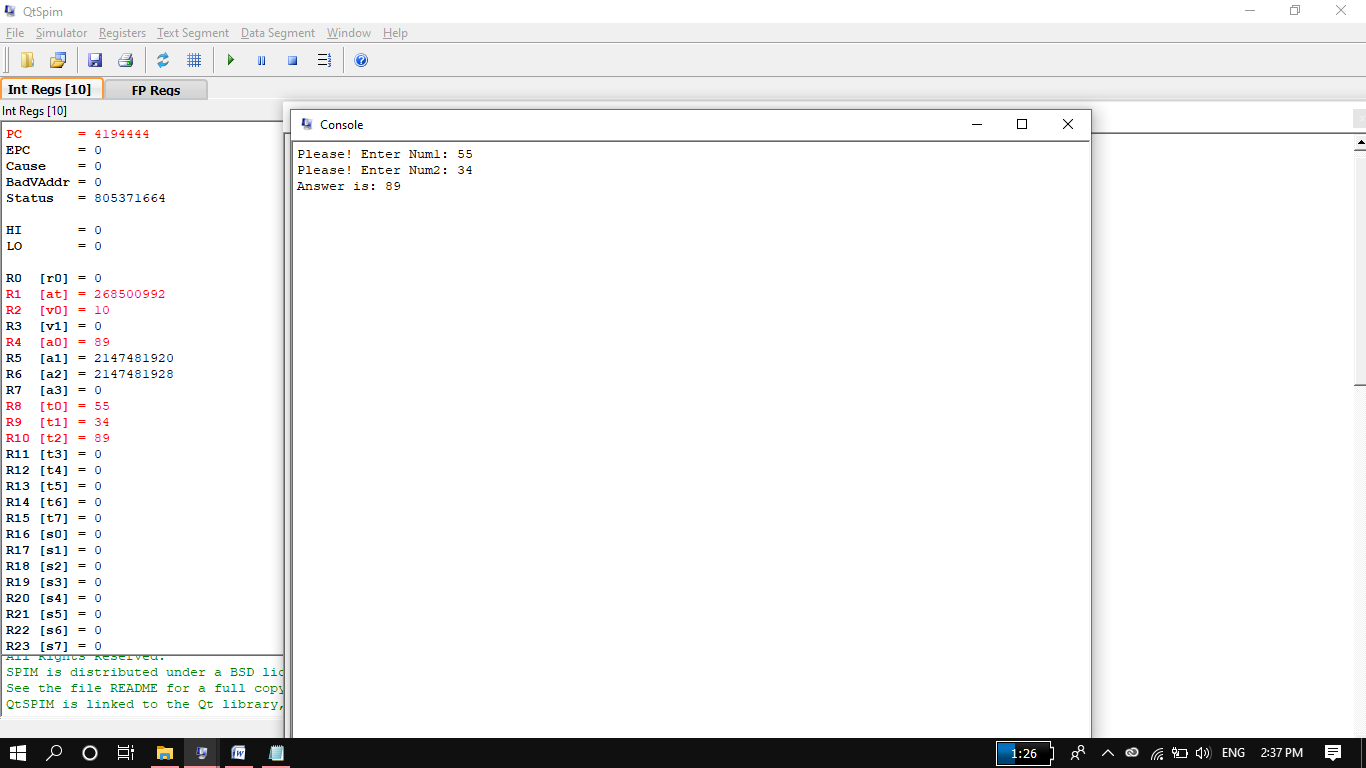
move $a0,$t2

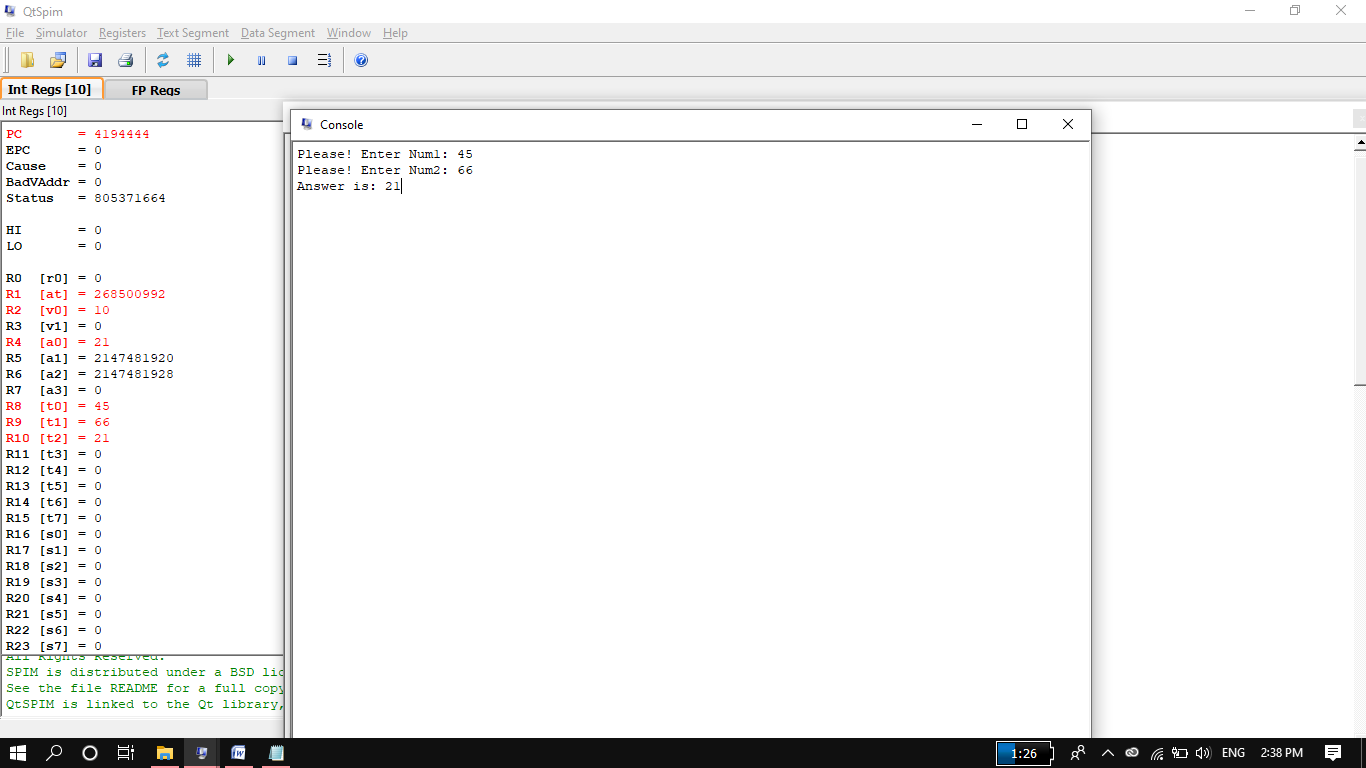
syscall

li $v0,10

syscall

**Output:**





**Task#03:**

**Source Code:**

.data

str1: .asciiz "Please! Enter Num1: " #z for null character. if we remove z then it will print all strings till next z.

str2: .asciiz "Please! Enter Num2: "

str3: .asciiz "Answer is: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

mul $t2,$t1,$t0 #this result store in common register $t2.by default here also #result store in low register

#if we want to store result in any other register it will give syntax error.

li $v0,4

la $a0,str3

syscall

li $v0,1

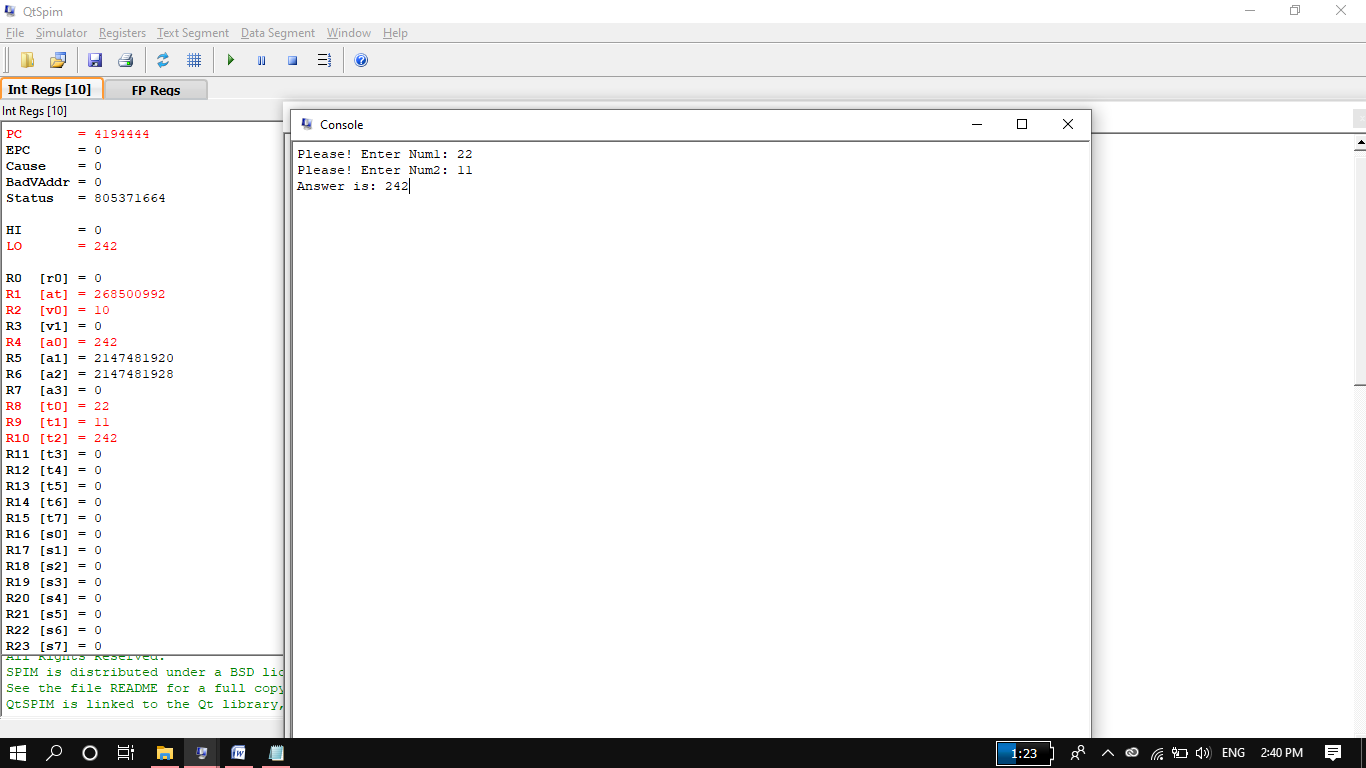
move $a0,$t2

syscall

li $v0,10

syscall

Output:



**Task 4:**

**Source Code:**

.data

str1: .asciiz "Please! Enter Num1: "

str2: .asciiz "Please! Enter Num2: "

str3: .asciiz "Answer is: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

mult $t0,$t1 #it will multiply t0 and t1 and store result in low register LH.

#if we want to store result in any other register it will give syntax error.

li $v0,4

la $a0,str3

syscall

li $v0,1 #request for displaying data in a0 register

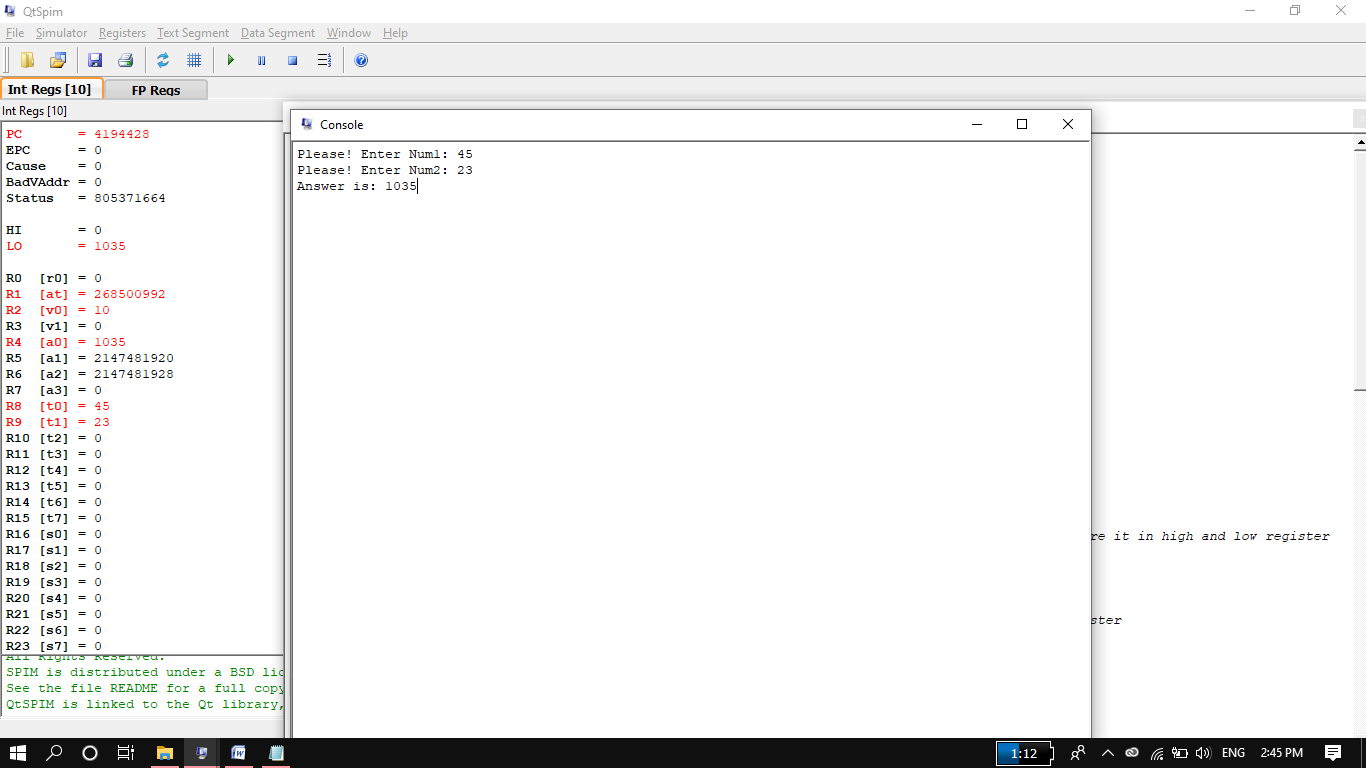
mflo $a0 #it move output of mult to a0 register

syscall

li $v0,10

syscall

**Output:**



**Task#05:**

**Source Code:**

.data

str1: .asciiz "Please! Enter Num1: "

str2: .asciiz "Please! Enter Num2: "

str3: .asciiz "Qoutient is: "

str4: .asciiz "\nRemender is: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

div $t0,$t1 #it will divide t0 and t1 and store reminder in high register and #Quotient in low register.

#we can also write it like div $t2,$t1,$t0. but in this cse only Quotient #stored in $t2.

li $v0,4

la $a0,str3

syscall

li $v0,1 #request for displaying data in a0 register

mflo $a0 #it move output of mult to a0 register

syscall

li $v0,4

la $a0,str4

syscall

li $v0,1

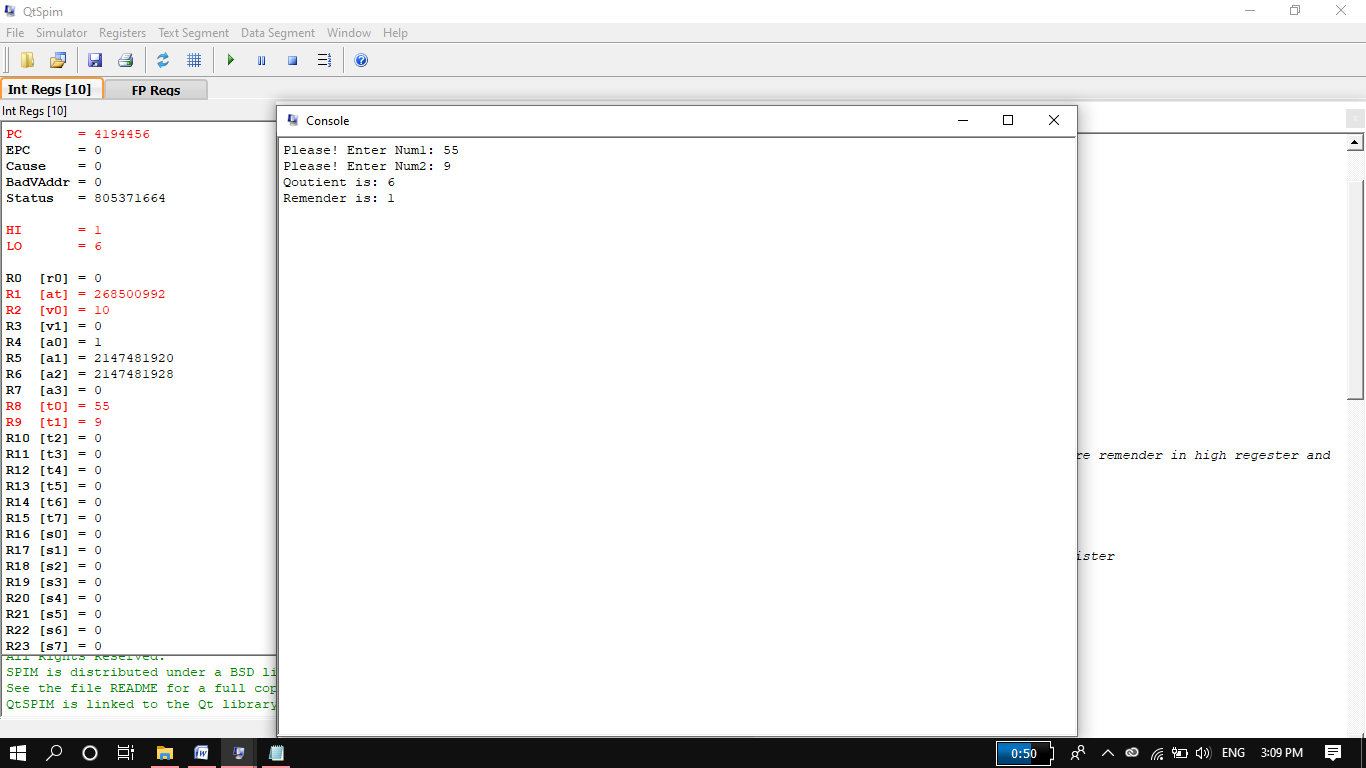
mfhi $a0

syscall

li $v0,10

syscall

Output:



**Task#06:**

**Source Code:**

.data

str1: .asciiz "Please! Enter Num1: " #z for null character. if we remove z then it will print all strings till next z.

str2: .asciiz "Please! Enter Num2: "

str3: .asciiz "Answer is: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

beq $t0,$t1 additionlabel

sub $t2,$t1,$t0

j exit

additionlabel: add $t2,$t1,$t0

exit:

li $v0,4

la $a0,str3

syscall

li $v0,1

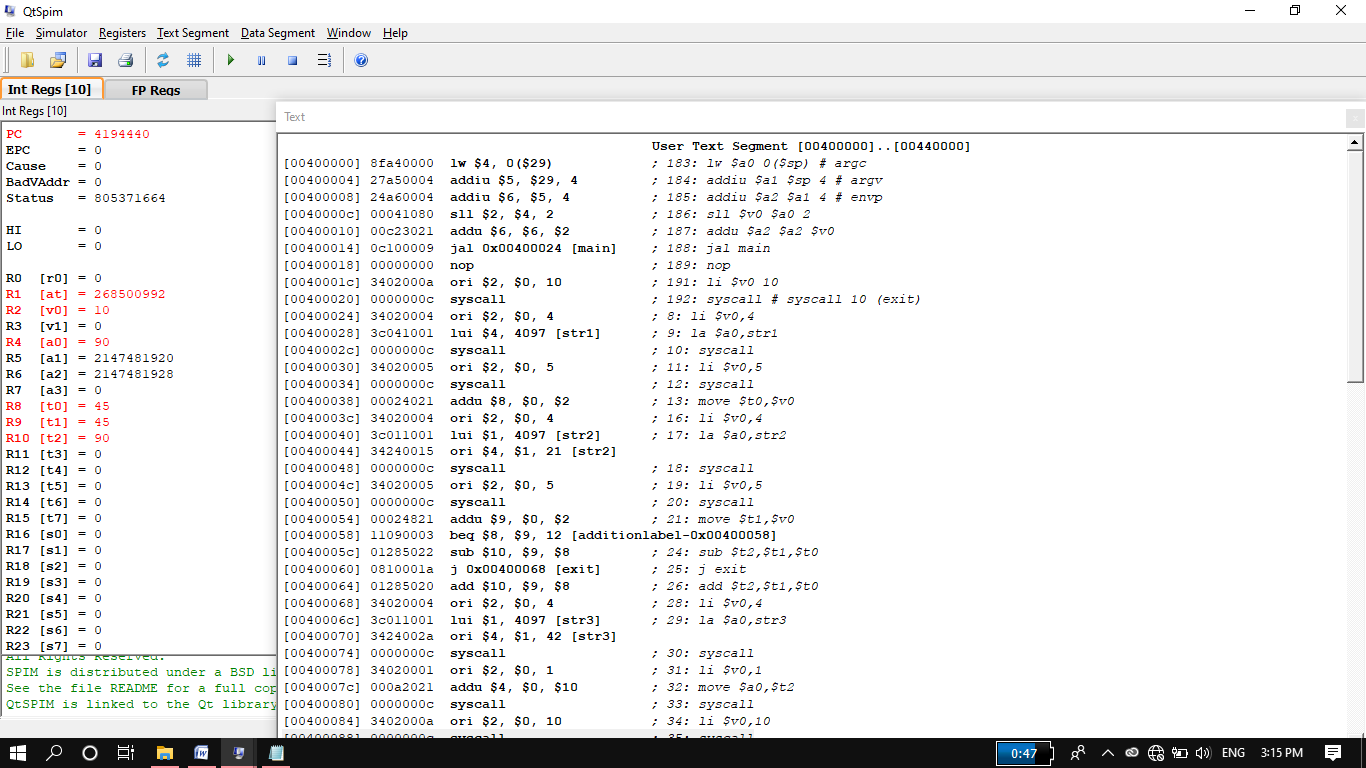
move $a0,$t2

syscall

li $v0,10

syscall

Output:



**Complete Task:**

**Source Code:**

.data

str1: .asciiz"Enter first NO:"

str2: .asciiz"Enter second NO:"

str3: .asciiz"AND:"

str4: .asciiz"\nOR:"

str5: .asciiz"\nNOR:"

str6: .asciiz"\nXOR:"

str7: .asciiz"\nAndi:"

str8: .asciiz"\nXori:"

str9: .asciiz"\nOri:"

str10: .asciiz"\nShift left:"

str11: .asciiz"\nShift Right:"

str12: .asciiz"\nShift Right arithm:"

str13: .asciiz"\nShift left by var:"

str14: .asciiz"\nShift Right by var:"

str15: .asciiz"\nShift Right arithm by var:"

.text

main:

li $v0,4 #system call for string print

la $a0,str1 #load address of string to a0 for print

syscall #print string

li $v0,5 #system call for user input.

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

li $v0,5

syscall

move $t1,$v0

li $v0,4

la $a0,str3

syscall

and $t2,$t1,$t0

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str4

syscall

or $t2,$t1,$t0

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str5

syscall

nor $t2,$t1,$t0

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str6

syscall

xor $t2,$t1,$t0

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str7

syscall

andi $t2,$t0,7

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str8

syscall

xori $t2,$t0,7

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str9

syscall

ori $t2,$t0,7

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str10

syscall

sll $t2,$t0,2

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str11

syscall

srl $t2,$t0,2

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str12

syscall

sra $t2,$t0,2

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str13

syscall

sllv $t2,$t0,$t1

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str14

syscall

srlv $t2,$t0,$t1

move $a0,$t2

li $v0,1

syscall

li $v0,4

la $a0,str15

syscall

srav $t2,$t0,$t1

move $a0,$t2

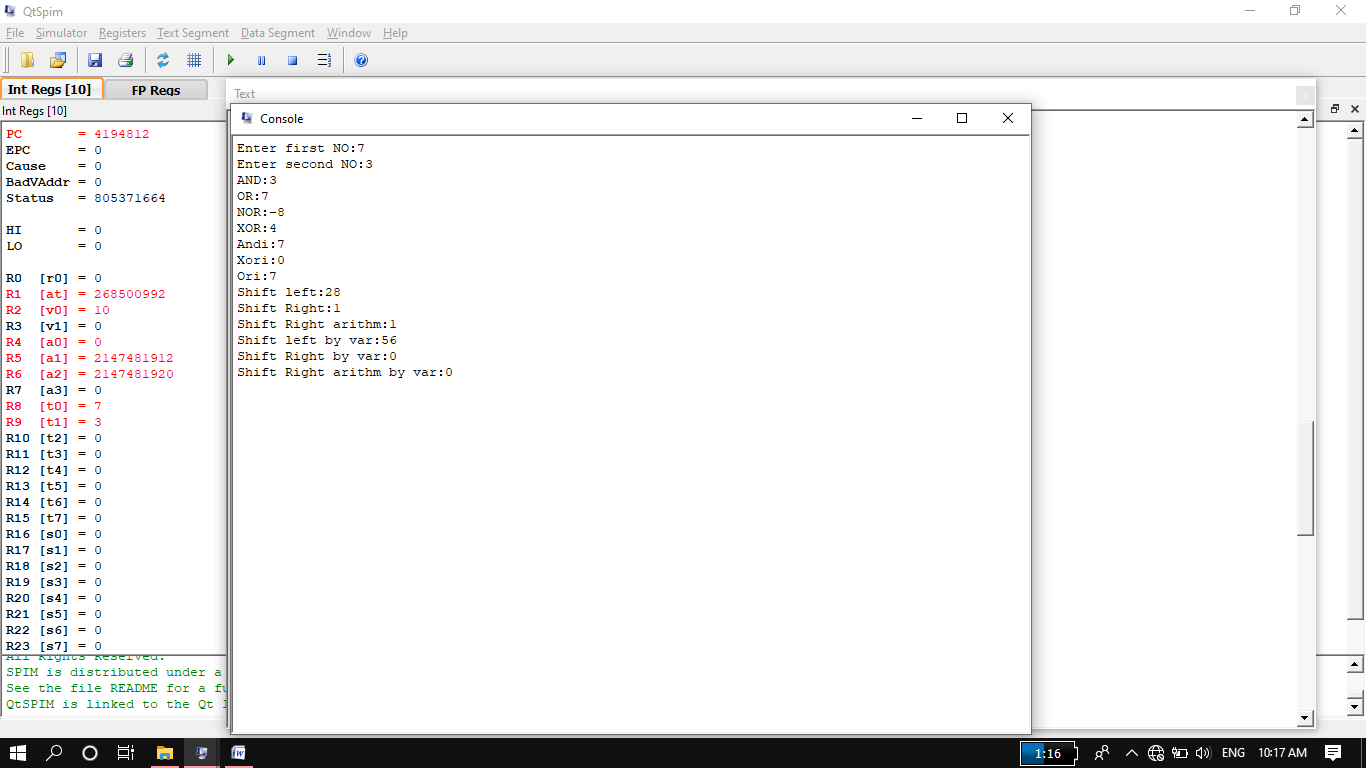
li $v0,1

syscall

li $v0,10 #program termination

syscall

**Output:**

****

## THE END