**Document Tasks**

**LAB # 05**

****

**Fall 2021**

**CSE304L Computer Organization & Architecture**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

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

Submitted to:

**Prof: Ammad khalil**

December 10, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Document Tasks:**

**Objective:**

* To know about different commands.

**Task\_4\_1.s:**

**Source Code:**

.data

str1: .asciiz"The Quotient is: "

str2: .asciiz"\nThe Remender is: "

str3: .asciiz"\nMultiplication is: "

.text

main: #---------------------

addi $t0, $0, 60

addi $t1, $0, 7

mult $t0, $t1 #multiplication

mflo $t6

div $t0, $t1 #division

li $v0,4

la $a0,str1

syscall

li $v0,1

mflo $a0

syscall

li $v0,4

la $a0,str2

syscall

li $v0,1

mfhi $a0

syscall

li $v0,4

la $a0,str3

syscall

li $v0,1

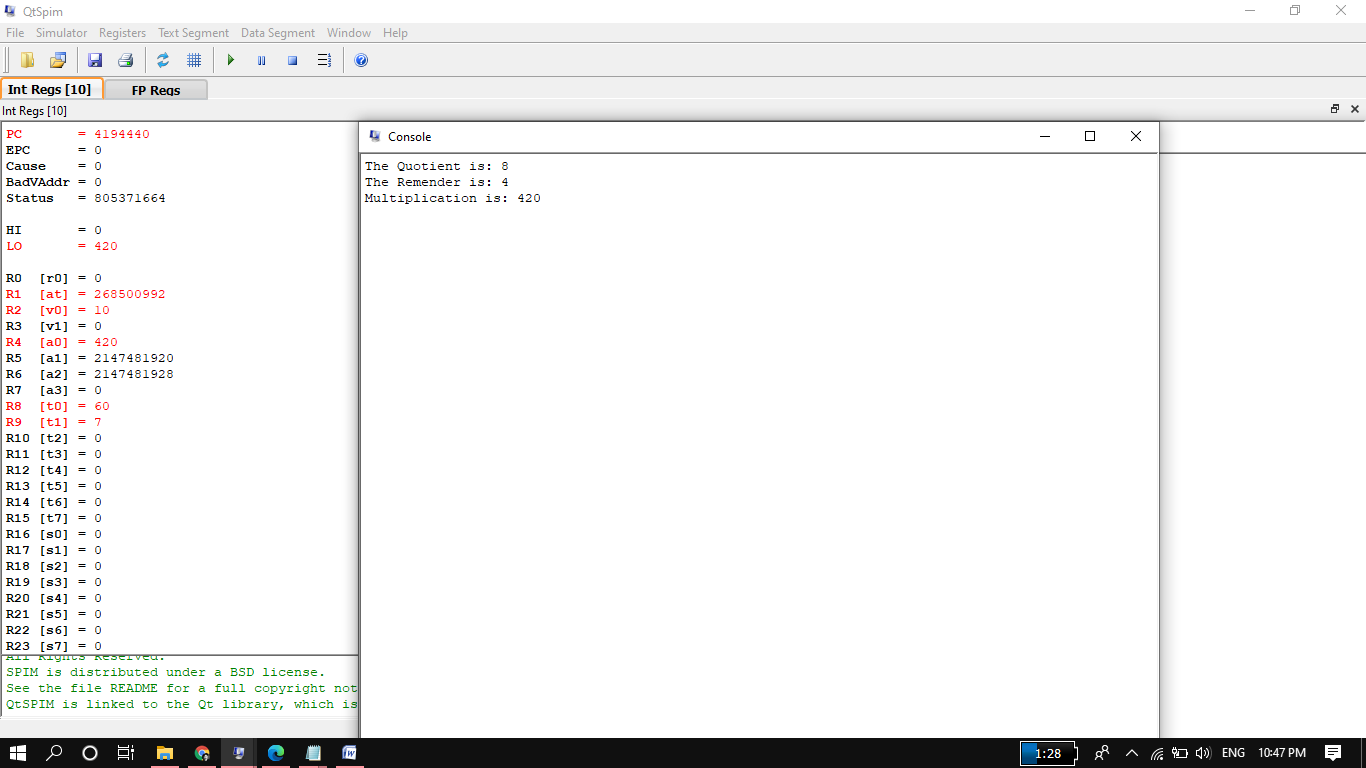
move $a0,$t6

syscall

li $v0,10

syscall

**Output:**



**Task4\_2.s:**

**Source Code:**

.data

str1: .asciiz"The Right Shift is: "

str2: .asciiz"\nThe left Shift is: "

.text

main: #---------------------

addi $t0, $0, 60

li $v0,4

la $a0,str1

syscall

srl $a0, $t0, 2 # a0 = t0 shifted right once

li $v0,1

syscall

li $v0,4

la $a0,str2

syscall

sll $a0, $t0, 2 # a0 = t0 shifted left once

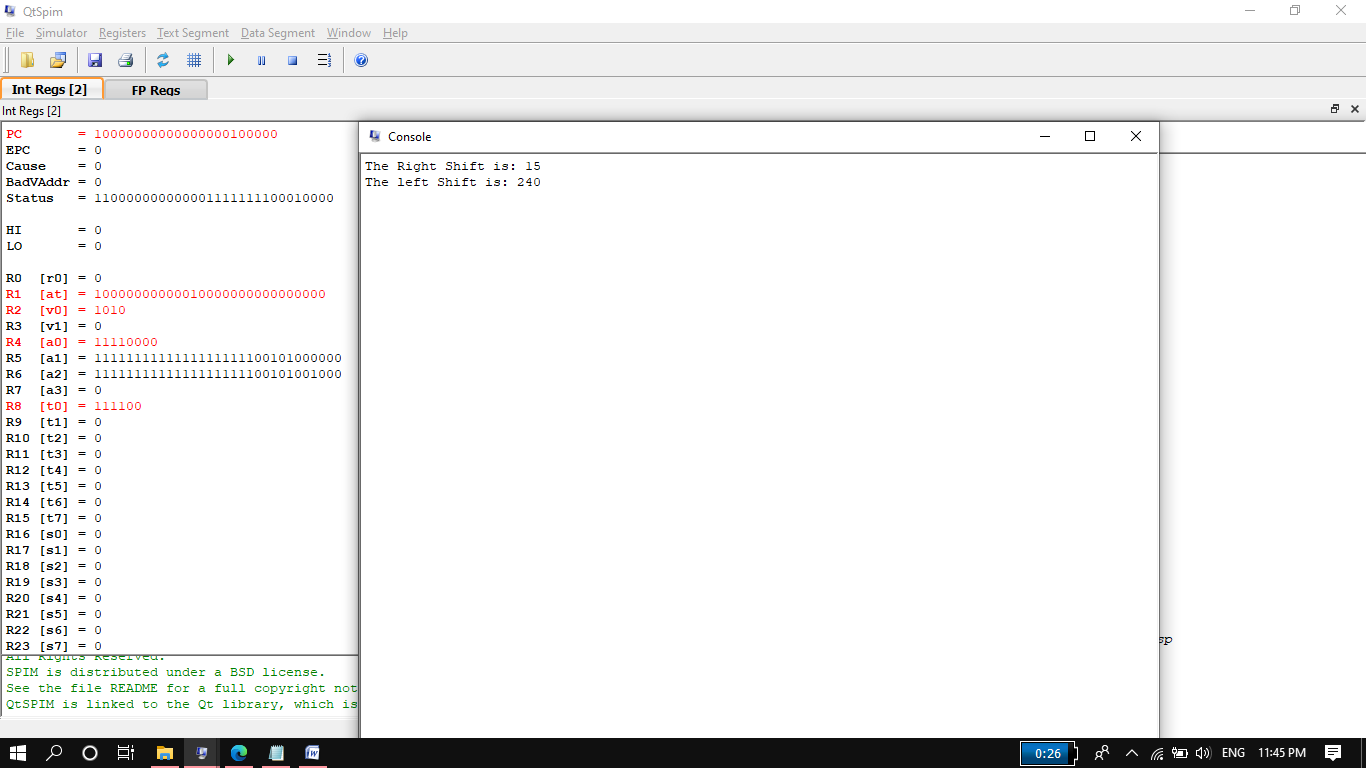
li $v0,1

syscall

#---------------------

fini: jr $ra

**Output:**



**Remarks:**

* In above right shift by 2 mean that we are actually dividing 60 by 2^2=4 so SR=15. And left shift by 2 mean we are multiplying 60 by 2^2=4 so SL=240. Similarly right shift by n mean that we are actually dividing Operand by 2^n and left shift by n mean we are multiplying Operand by 2^n.
* In case of binary, right shift by 2 mean we are deleting the 2 LSBs of 60, while left shift by 2 mean we are adding two zeros to LSBs of 60.
* The right shift is sign sensitive. It doesn’t properly perform division when operand is negative.
* The left shift is not sign sensitive. It properly performs multiplication when operand is negative.
* But what will happen if operand is floating number, Is sll or srl performing work properly? Or if t0%n !=0 then as srl work properly? Sorry for that b/c we have not studied floating numbers yet.

**Task4\_3.s:**

**Source Code:**

.data

str1: .asciiz"please enter num x: "

str2: .asciiz"the result in the form 18x: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

sll $t1,$t0,4 #its mean we multiply t0 by 2^4=16

sll $t2,$t0,1 #its mean we multiply t0 by 2^1=2

add $t3,$t1,$t2 #we will add t0 and t1 it will be equal to 18\*x

li $v0,4

la $a0,str2

syscall

li $v0,1

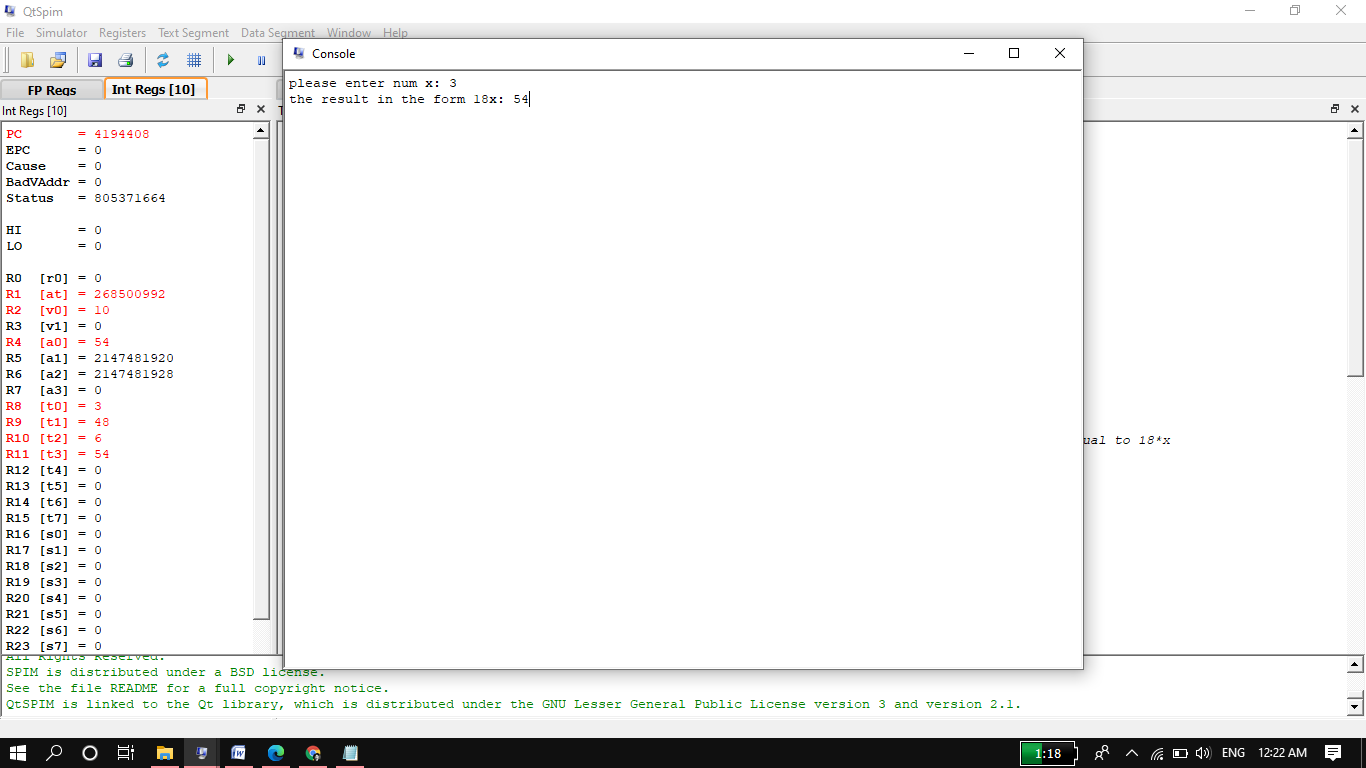
move $a0,$t3

syscall

li $v0,10

syscall

**Output:**



**Remarks:**

**Why is shifting a register faster than multiplying it?**

>>Shifting is generally **a lot faster than multiplying at an instruction level** but we may well be wasting your time doing premature optimizations. The compiler may well perform these optimizations at compile time. Doing it yourself will affect readability and possibly have no effect on performance.

**Task4\_4.s:**

**Source Code:**

.data

str1: .asciiz"please! Enter a number: "

str2: .asciiz"The result is complement of 10th bit: "

.text

main:

li $v0,4

la $a0,str1

syscall

li,$v0,5

syscall

move $t0,$v0

li $v0,4

la $a0,str2

syscall

sll $a0,$t0,21 #Algorithm 1

srl $a0,$a0,31

andi $a0, $t0, 1024 #Algorithm 2 both algorithms have same function (give us same result)

srl $a0, $a0, 10

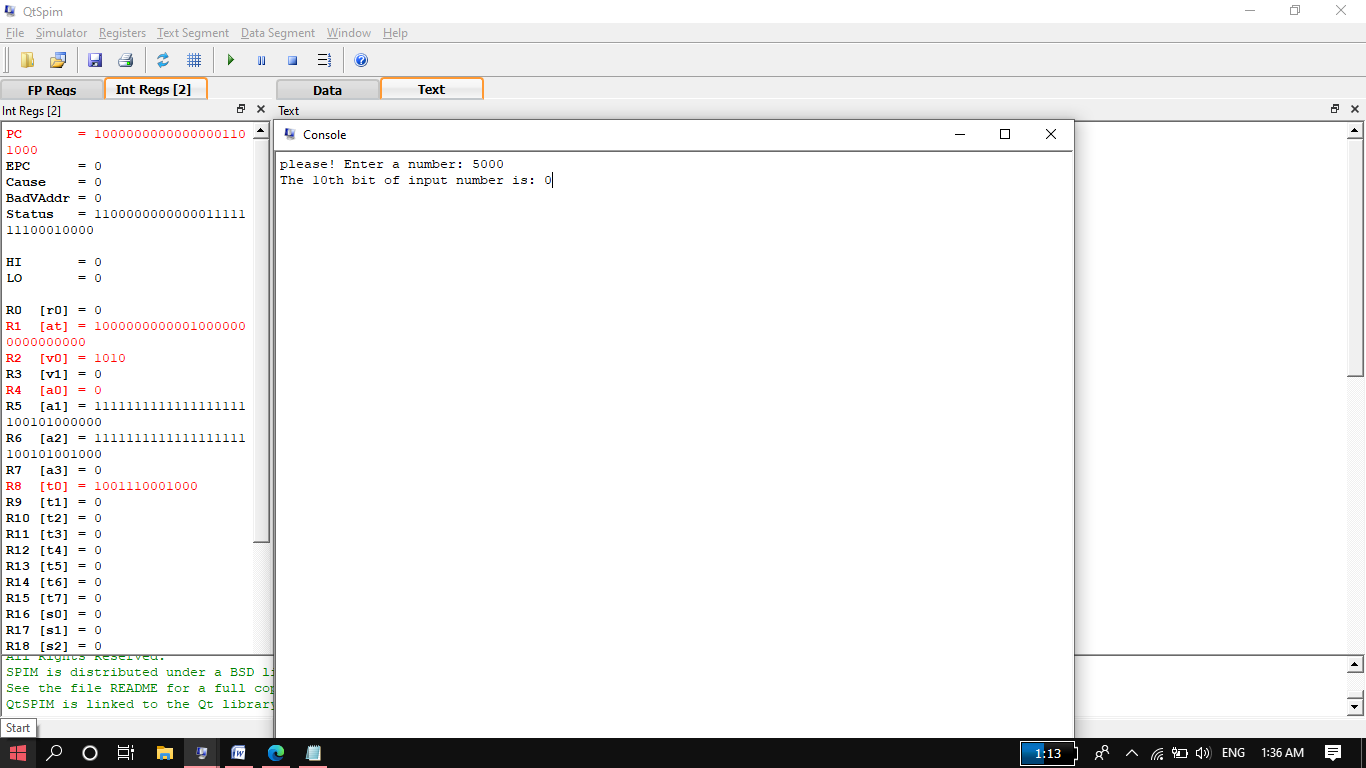
li $v0,1

syscall

li $v0,10

syscall

**Output:**



**Remarks:**

>>We concluded that both algorithms do same work.

>>The word "mask" is often used to describe something that blocks the view except for a hole through which one can see. Does the number 1024 behave like a mask in this sense?

**Task4\_5.s:**

**Source Code:**

.data

str1: .asciiz"\nPlease! Enter a Number: "

str2: .asciiz"The Result 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 $t5,0xffff #these three line for storing the mask no in t5 given in task 14 in binary form.

sll $t5,$t5,16 #shifted left 16 times.

ori $t5,$t5,0xfbff

and $a0, $t0,$t5

li $t6,1024 #2nd method for storing the mask no in t7 given in task 14 in binary form.

nor $t7,$t6,$t6

and $a0, $t0,$t7

li $v0,1

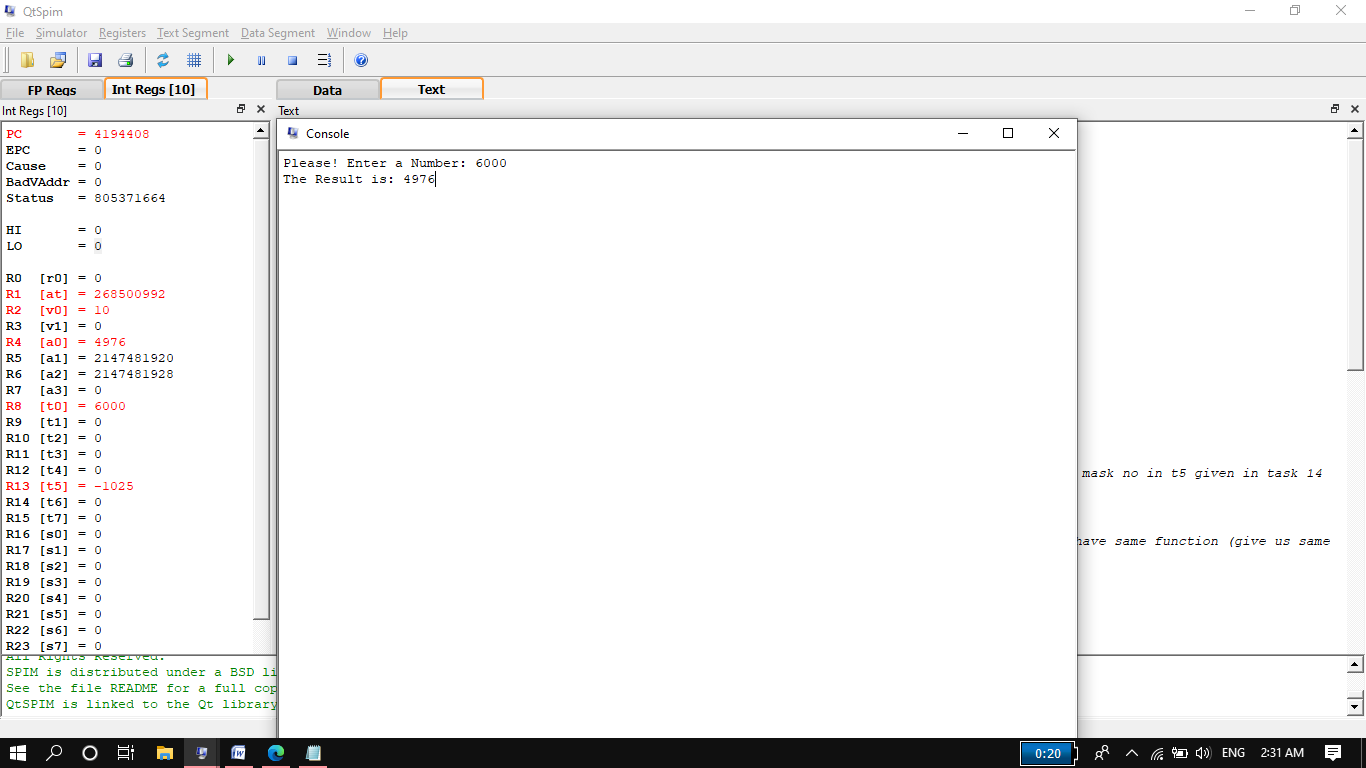
syscall

j main #for jumping

li $v0,10

syscall

**Output:**



**Remarks:**

Here we created two masks. Both masks perform same function.

**Task4\_6.s:**

**Source Code:**

.data

str1: .asciiz"\nPlease! Enter a number you want to flip its 10th bit: "

str2: .asciiz"After flipping the 10th bit the number become: "

.text

main:

li $v0,4

la $a0,str1

syscall

li $v0,5

syscall

move $t0,$v0

li $t1,512 #appropirate mask that we will XOR with input number and it will flip 10th bit only of input number.

xor $t2,$t0,$t1

li $v0,4

la $a0,str2

syscall

li $v0,1

move $a0,$t2

syscall

j main #for jumping.

li $v0,10

syscall

**Output:**

