Computer Organization And Architecture

Lab – 5

S Abhishek

AM.EN.U4CSE19147

**1. Construct a solid rectangle with '\*' using MIPS assembly language. Get the length and breadth from the user.**

**(Hint :Use jump/conditional branching)**

**Eg: Length = 6, breadth = 2**

.data

prompt1: .asciiz "Enter the Length : "

prompt2: .asciiz "Enter the Breadth : "

result: .asciiz "\*"

newline: .asciiz "\n"

.text

.globl main

main:

addi $v0,$0,4

la $a0,prompt1

syscall

addi $v0,$0,5

syscall

add $8,$0,$v0

addi $v0,$0,4

la $a0,prompt2

syscall

addi $v0,$0,5

syscall

add $9,$0,$v0

li $10,0

Loop:

li $11,0

bne $10,$9, Print

j Exit

Print:

addi $11,$11,1

addi $v0,$0,4

la $a0,result

syscall

bne $11,$8, Print

beq $11,$8, Line

Line:

addi $v0,$0,4

la $a0,newline

syscall

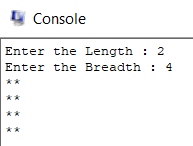
addi $10,$10,1

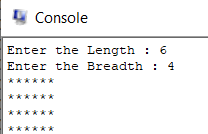
j Loop

Exit:

ori $v0,$0,10

syscall





**2. Write a program in MIPS assembly language to swap two values. Get input from the user and display the values on screen after swap operation.**

.data

prompt1: .asciiz "Enter the Number 1 : "

prompt2: .asciiz "Enter the Number 2 : "

output1: .asciiz "\nNumber 1 : "

output2: .asciiz "\nNumber 2 : "

output3: .asciiz "\n\nAfter Swapping Number 1 : "

output4: .asciiz "\nAfter Swapping Number 2 : "

.text

.globl main

main:

addi $v0,$0,4

la $a0,prompt1

syscall

addi $v0,$0,5

syscall

add $8,$0,$v0

addi $v0,$0,4

la $a0,prompt2

syscall

addi $v0,$0,5

syscall

add $9,$0,$v0

li $v0,4

la $a0,output1

syscall

li $v0,1

move $a0,$8

syscall

li $v0,4

la $a0,output2

syscall

li $v0,1

move $a0,$9

syscall

move $10,$8

move $8,$9

move $9,$10

li $v0,4

la $a0,output3

syscall

li $v0,1

move $a0,$8

syscall

li $v0,4

la $a0,output4

syscall

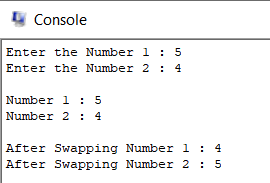
li $v0,1

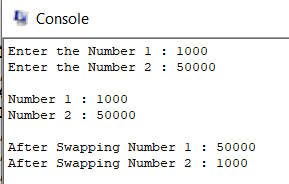
move $a0,$9

syscall

ori $v0,$0,10

syscall

****

****

**3. How can we implement logical not? Write down the statement and show the result. (Hint : Use NOR)**

.data

prompt1: .asciiz "Enter the Number : "

result: .asciiz "The Result of NOT is : "

.text

.globl main

main:

addi $v0,$0,4

la $a0,prompt1

syscall

addi $v0,$0,5

syscall

add $8,$0,$v0

nor $9,$8,$8

li $v0,4

la $a0,result

syscall

li $v0,1

move $a0,$9

syscall

ori $v0,$0,10

syscall

****

****

**4. Write a program that reads two integer numbers A and B. The program must indicate if one of these numbers is multiple of the other one.**

.data

prompt1: .asciiz "Enter the Number 1 : "

prompt2: .asciiz "Enter the Number 2 : "

prompt3: .asciiz "It is the Divisible of One Another."

prompt4: .asciiz "It is not the Divisible of One Another."

.text

.globl main

main:

addi $v0,$0,4

la $a0,prompt1

syscall

addi $v0,$0,5

syscall

add $8,$0,$v0

addi $v0,$0,4

la $a0,prompt2

syscall

addi $v0,$0,5

syscall

add $9,$0,$v0

slt $10,$8,$9

beq $t1,1,B\_Greater

beq $10,$zero,A\_Greater

B\_Greater:

div $t1, $t0

mfhi $a0

beq $a0,$0,P1

j P2

A\_Greater:

div $t0, $t1

mfhi $a0

beq $a0,$0,P1

j P2

P1:

addi $v0,$0,4

la $a0,prompt3

syscall

j Exit

P2:

addi $v0,$0,4

la $a0,prompt4

syscall

j Exit

Exit:

ori $v0,$0,10

syscall

****

****

****

****

**5. Run the following code in QtSPIM.**

**a. Comment the following code and find its C equivalent.**

**b. State the addressing mode and instruction type for each instruction in the given code.**

**c. Explain branch instruction (bne) in detail with respect to the given code's PC values.**

**.data**

**nl: .asciiz "\n"**

**.text**

**.globl main**

**main:**

**li $t0, 0**

**li $t1, 10**

**loop:**

**addi $t0, $t0, 1**

**bne $t1, $t0, loop**

**sll $t0,$t1,2**

**li $v0,1**

**move $a0, $t0**

**syscall**

**li $v0,4**

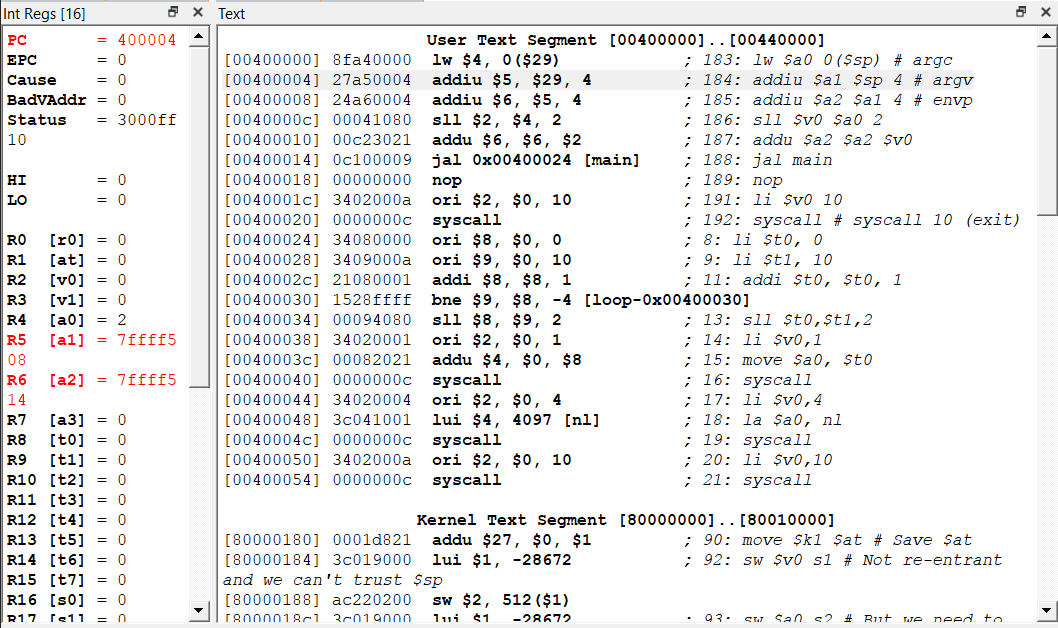
**la $a0, nl**

**syscall**

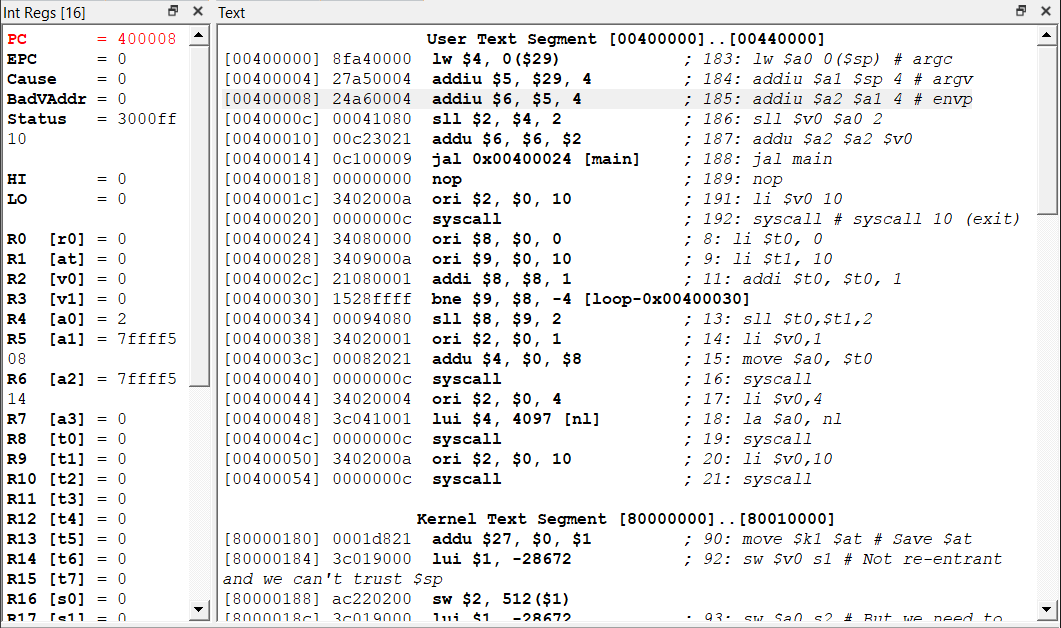
**li $v0,10**

**syscall**

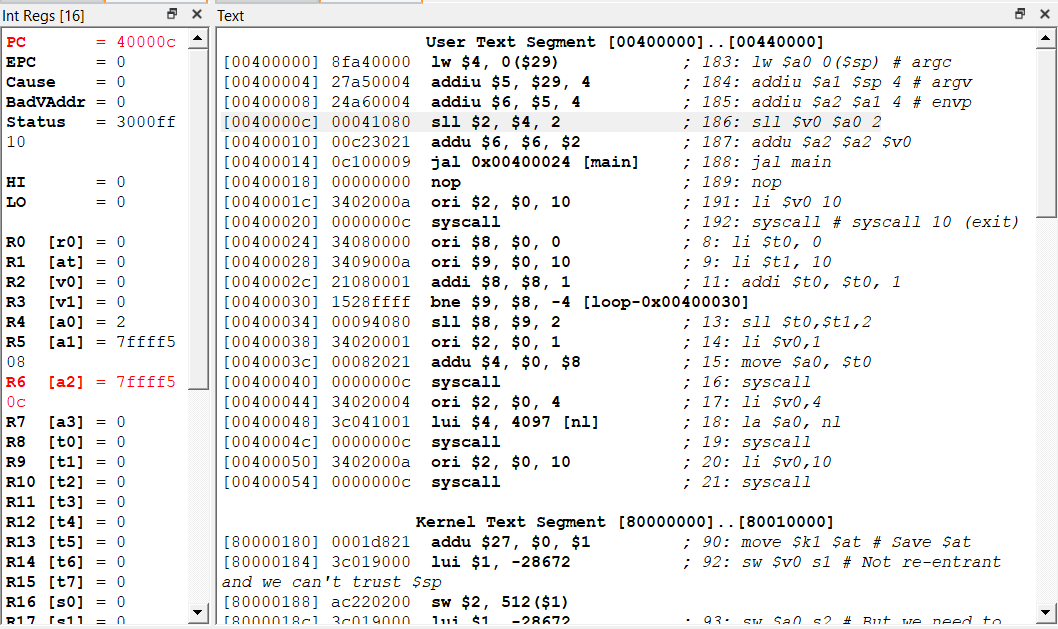
* This is a looping program.
* The initial value of the counter is 0 and the loop terminates when the counter value becomes 10.
* So, the loop iterates 10 times and each time the counter value is increased by 1.

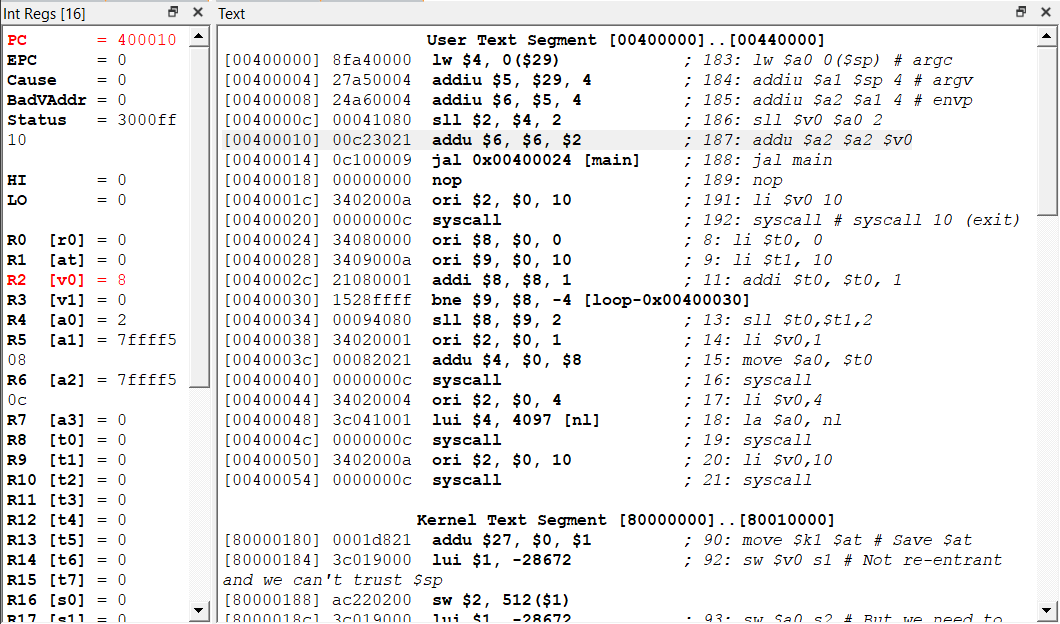
****

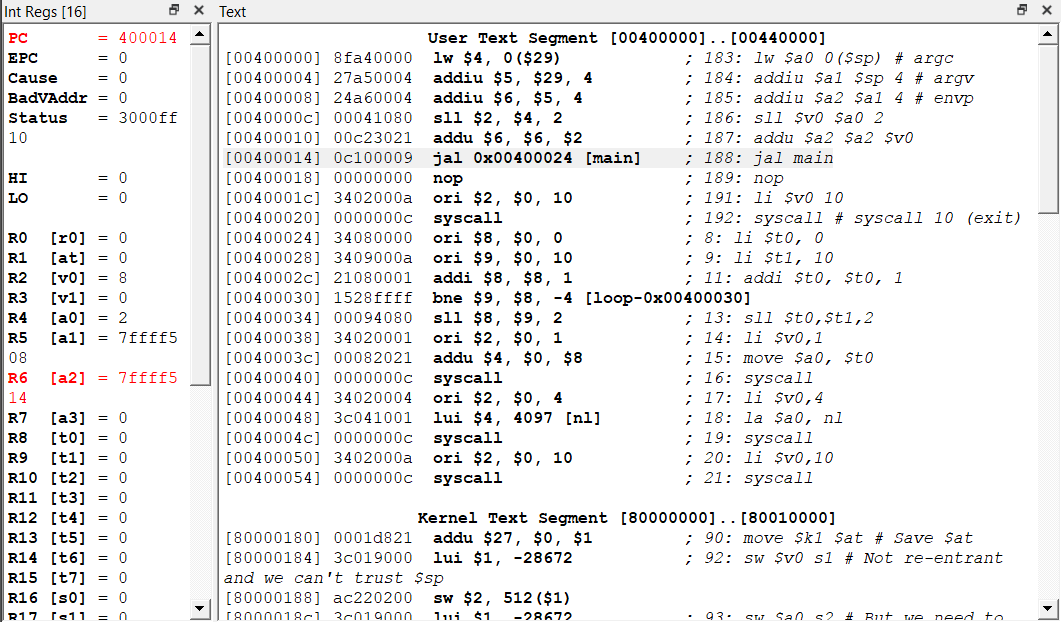
* In the first image we shall see that the initial program counter is 400004 and the initial counter value is 0.
* When it enters the Loop its added with 1 and the counter value becomes 1.
* Now it checks whether the counter is equal to 10.
* In this case its not equal, so the looping continues to the next iteration.
* So its sequential execution and the next PC value will be **Current PC + 4.**
* New PC value: 400004 + 4 = 400008
* This PC + 4 step continues until the counter value becomes 10.

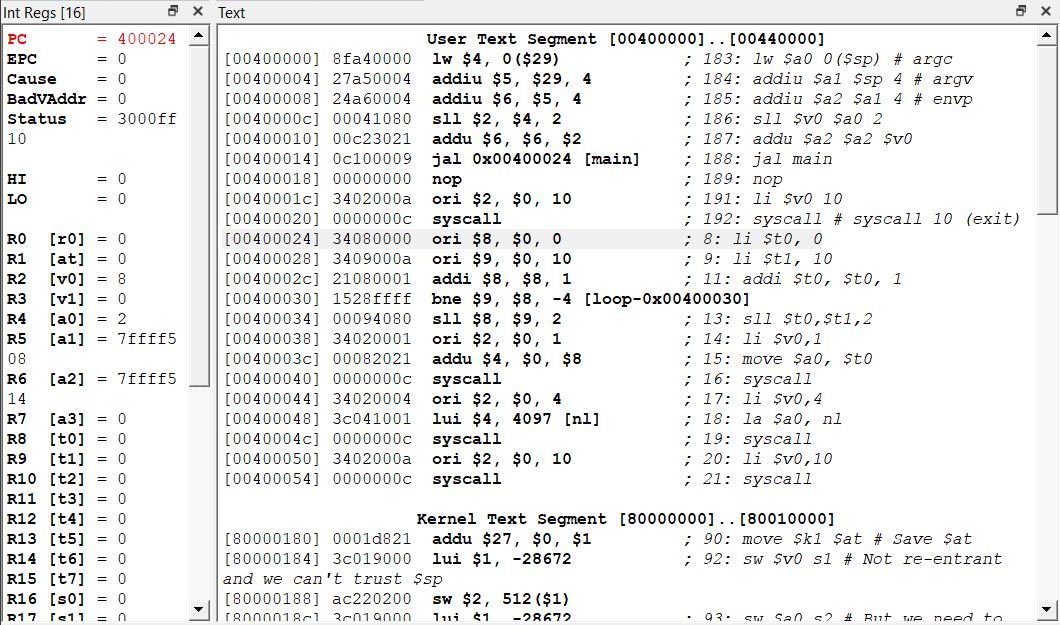
****

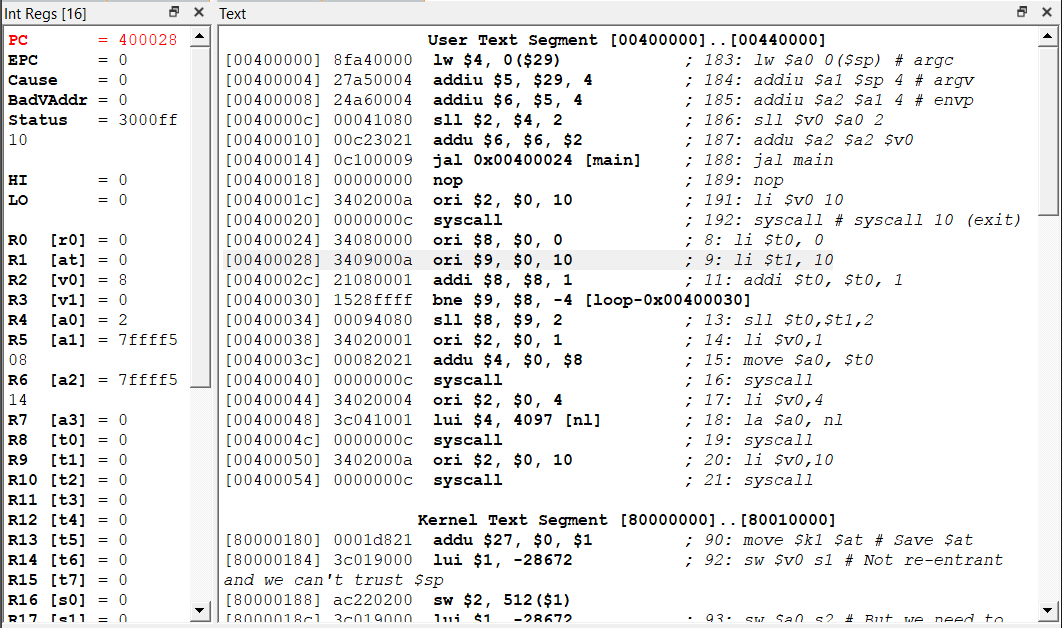
**PC + 4 = 400008 + 4 = 40000c**

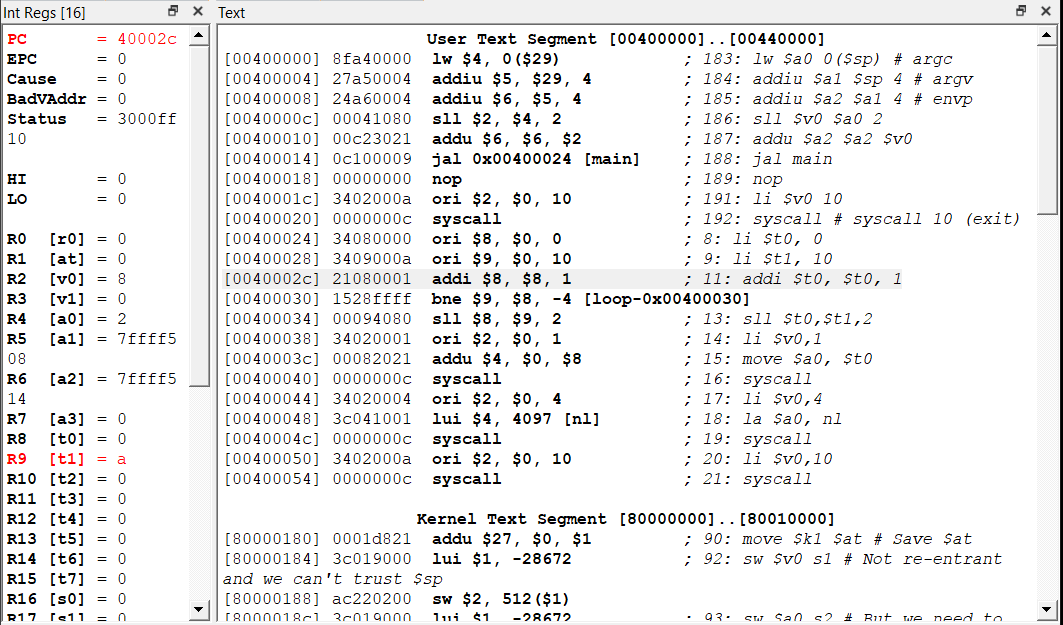
****

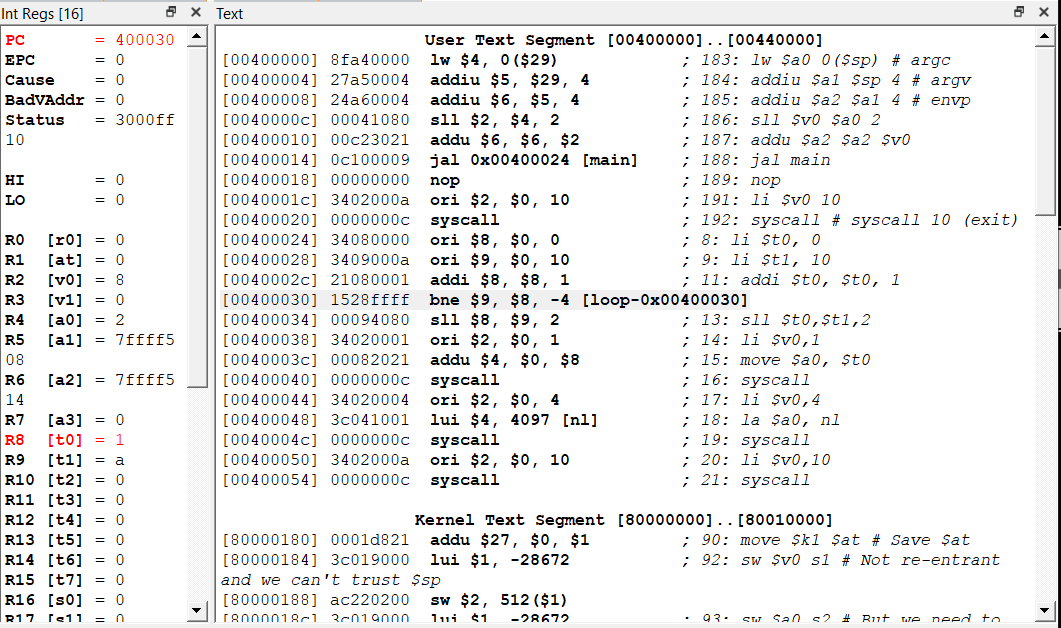
****

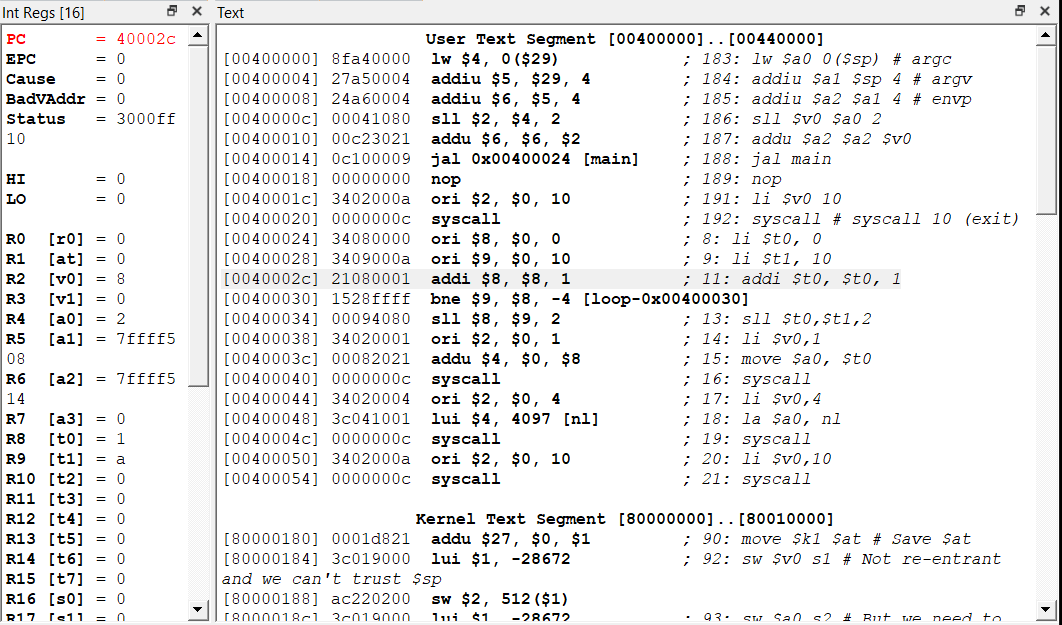
****

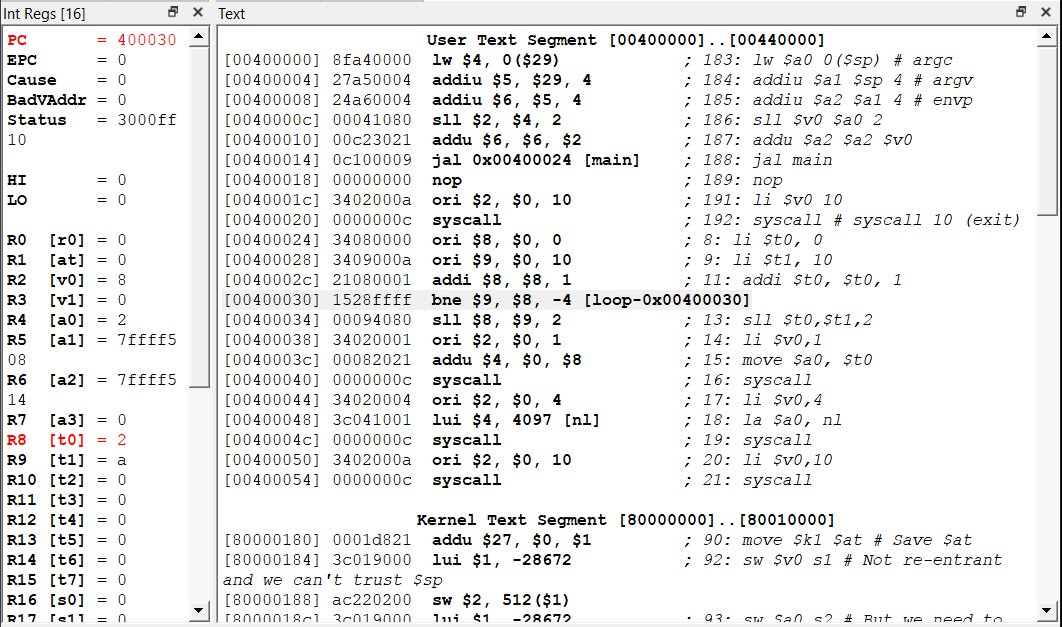
****

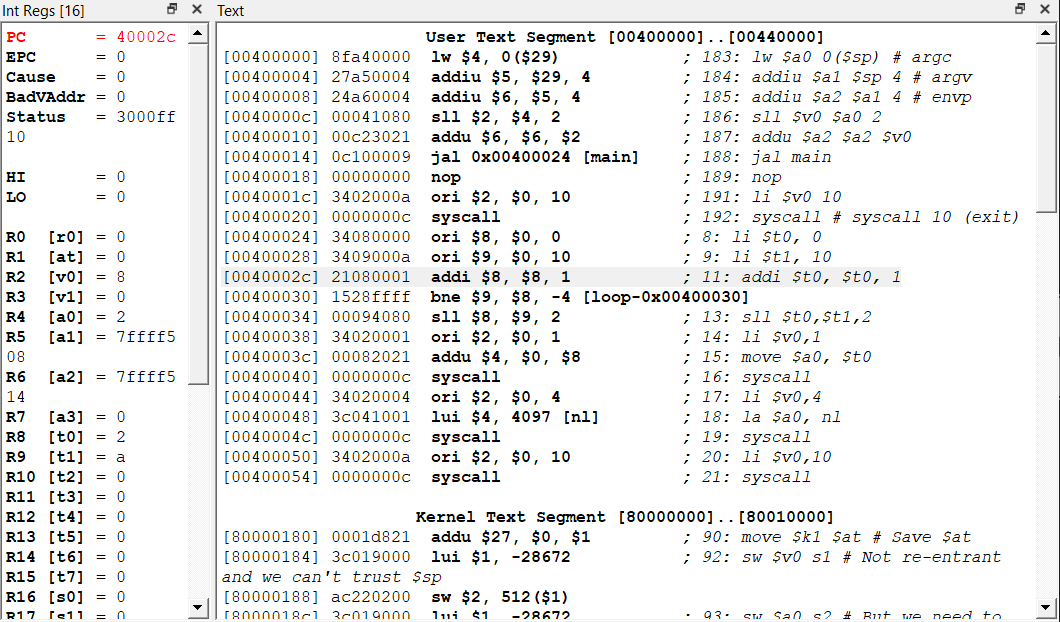
****

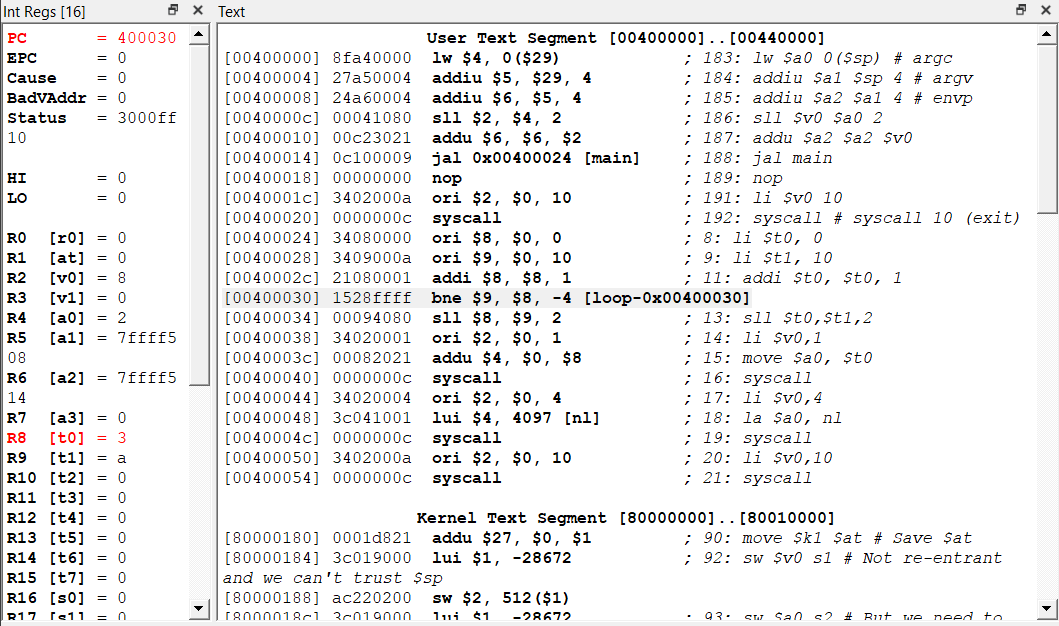
****

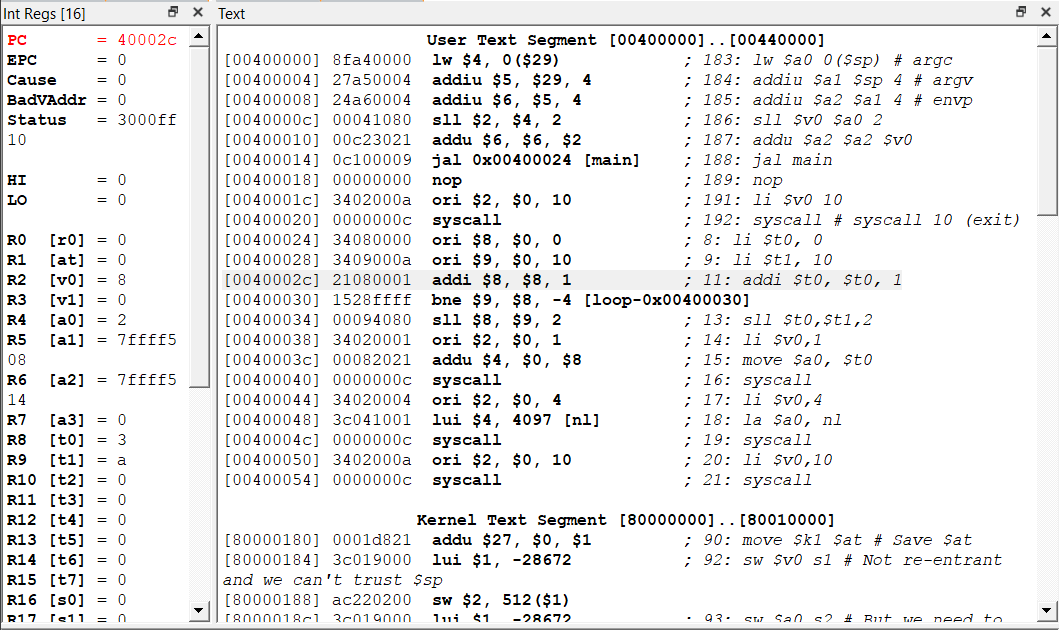
****

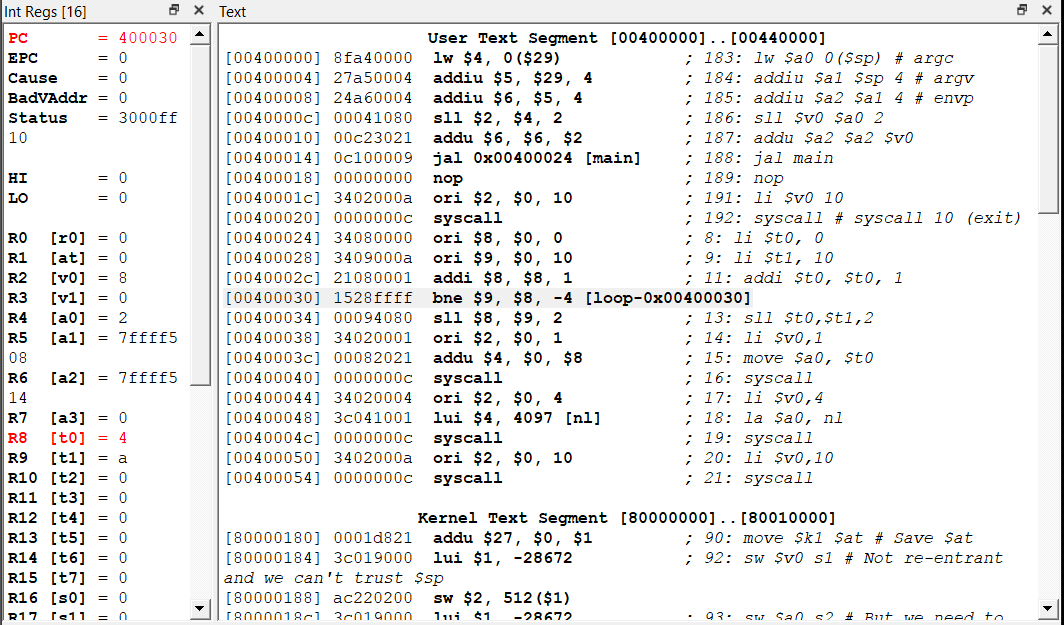
****

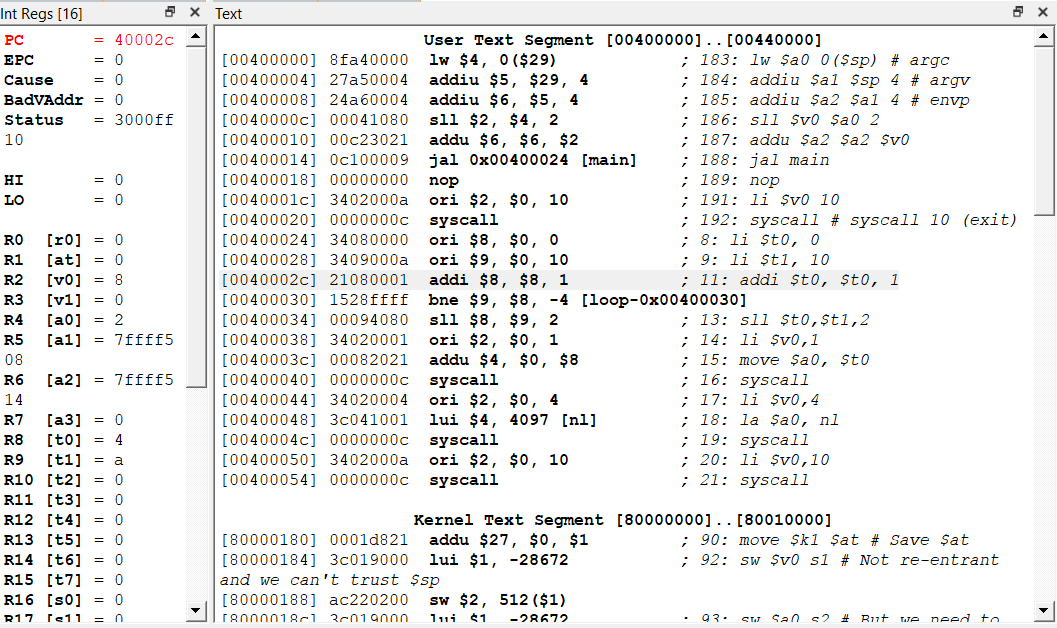
****

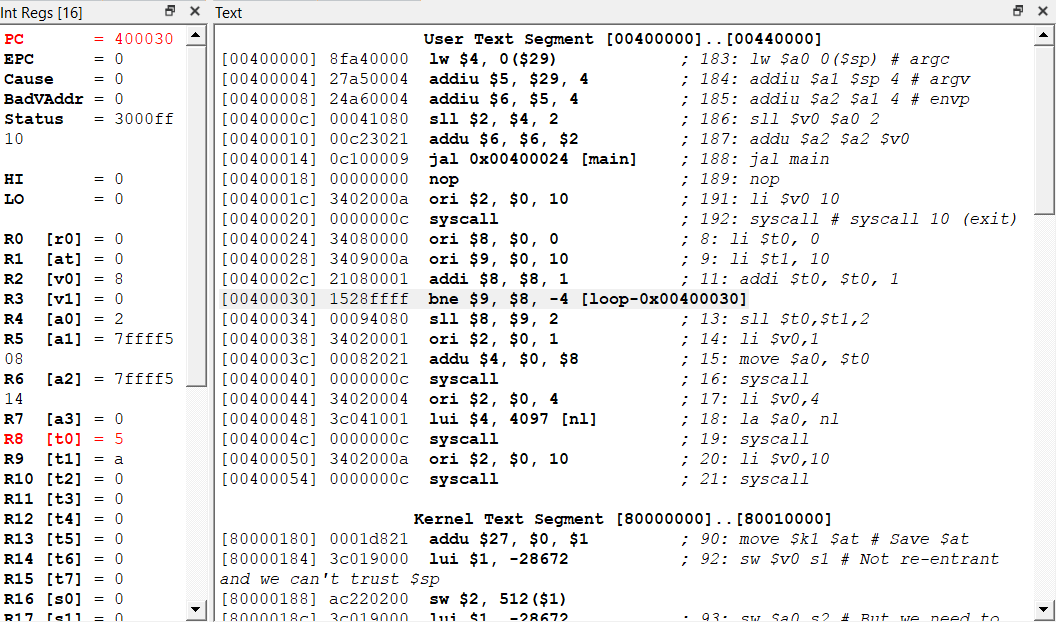
****

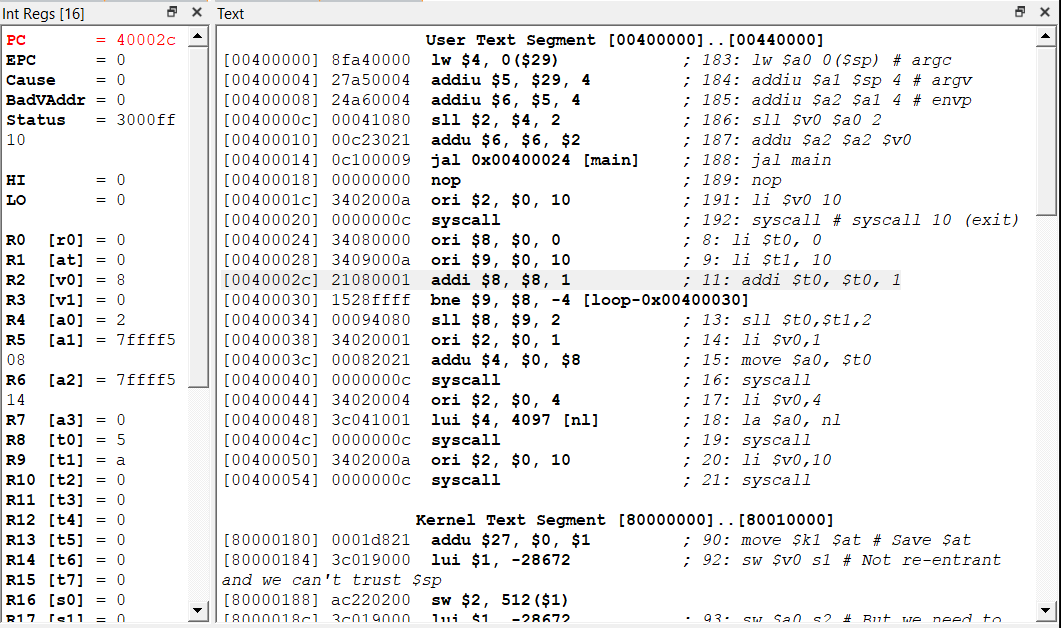
****

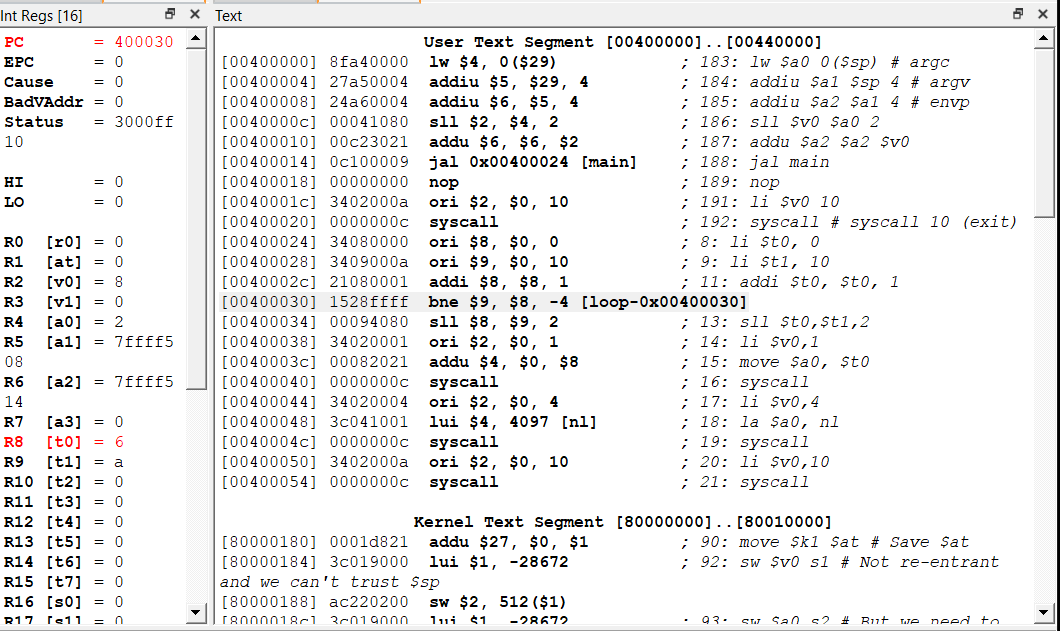
****

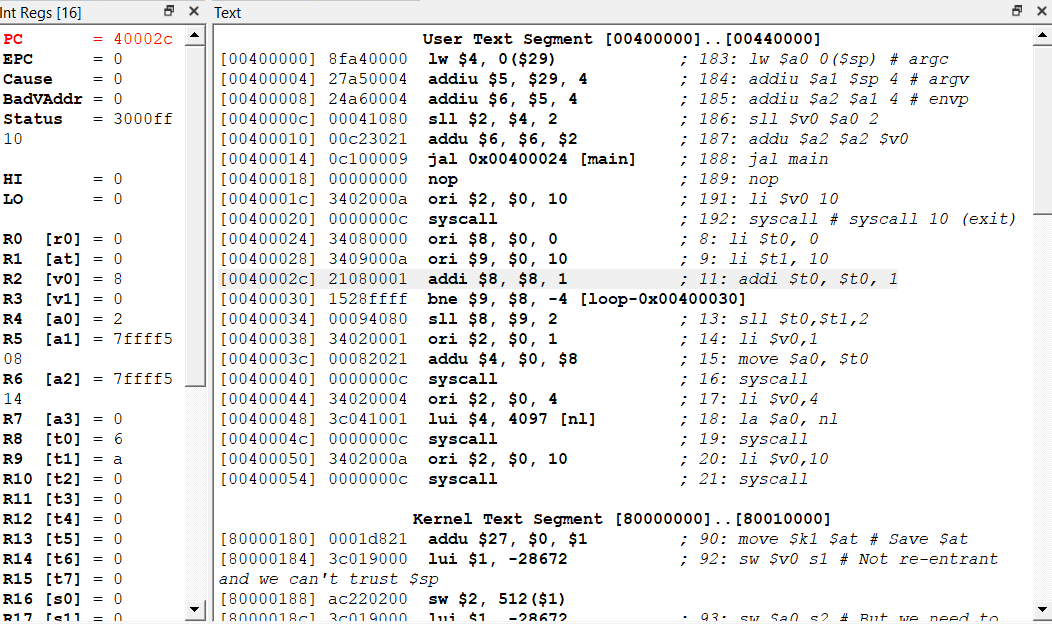
****

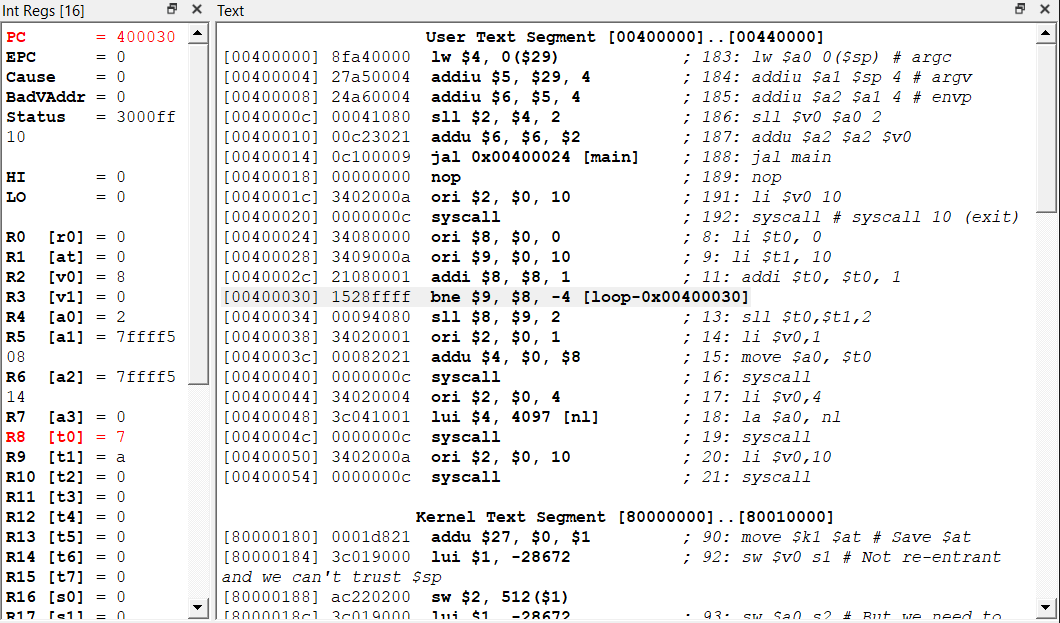
****

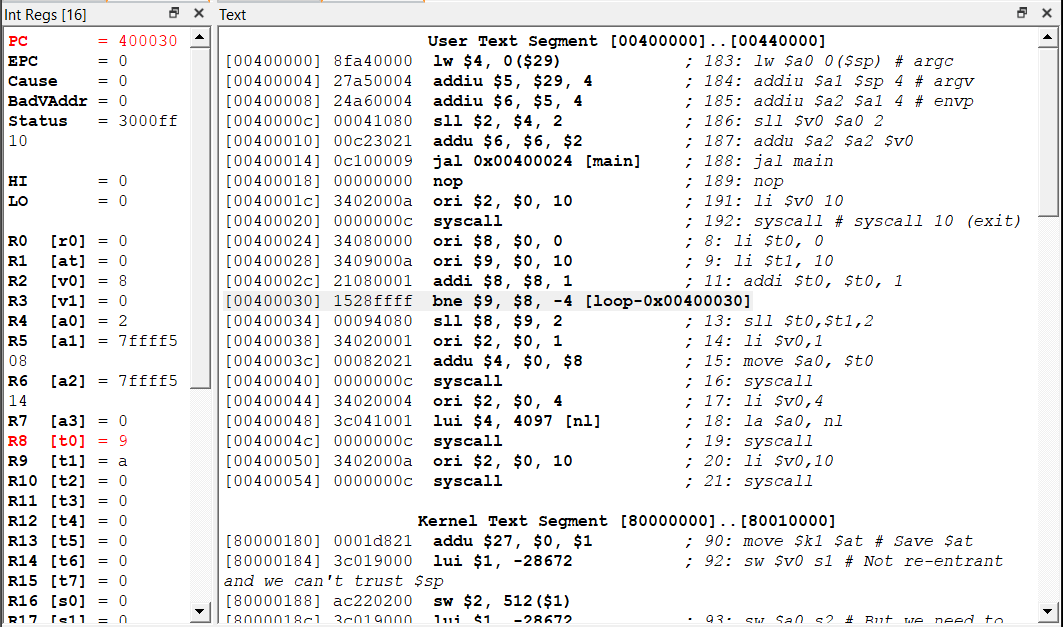
****

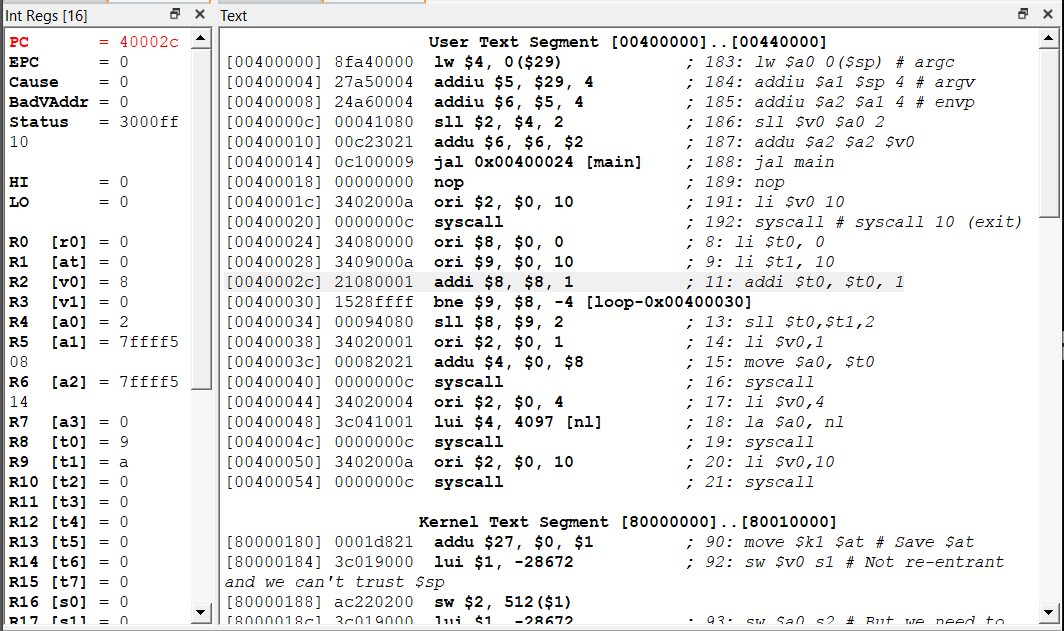
****

****

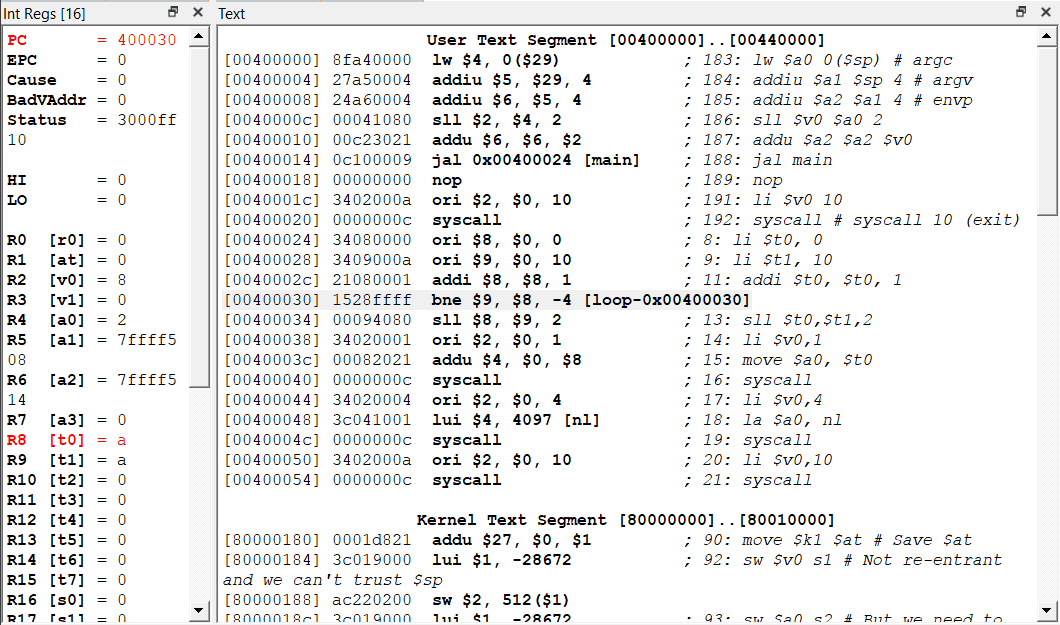
****

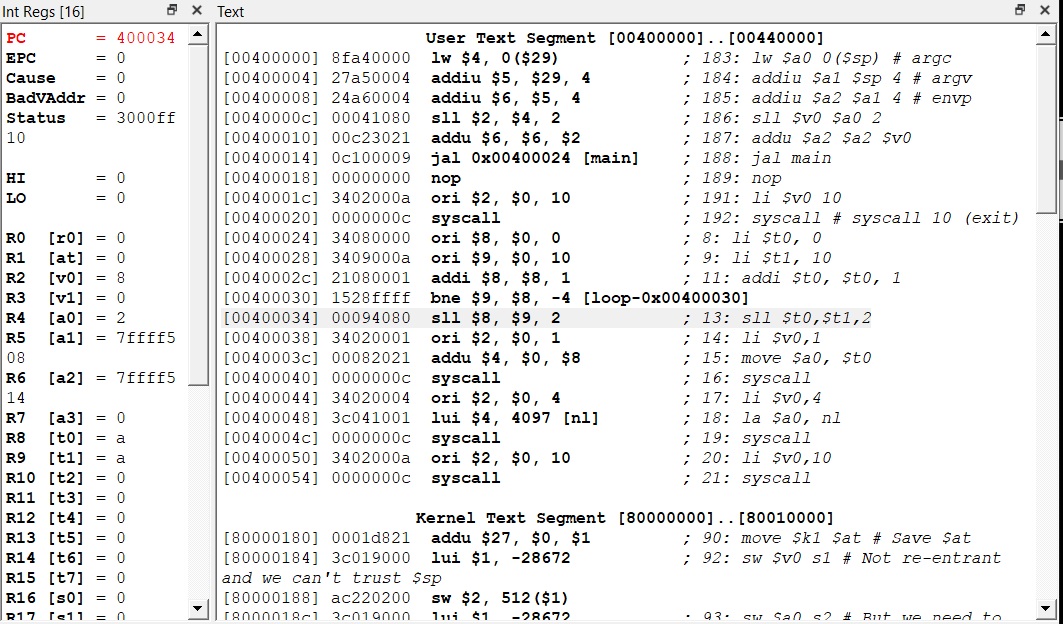
****

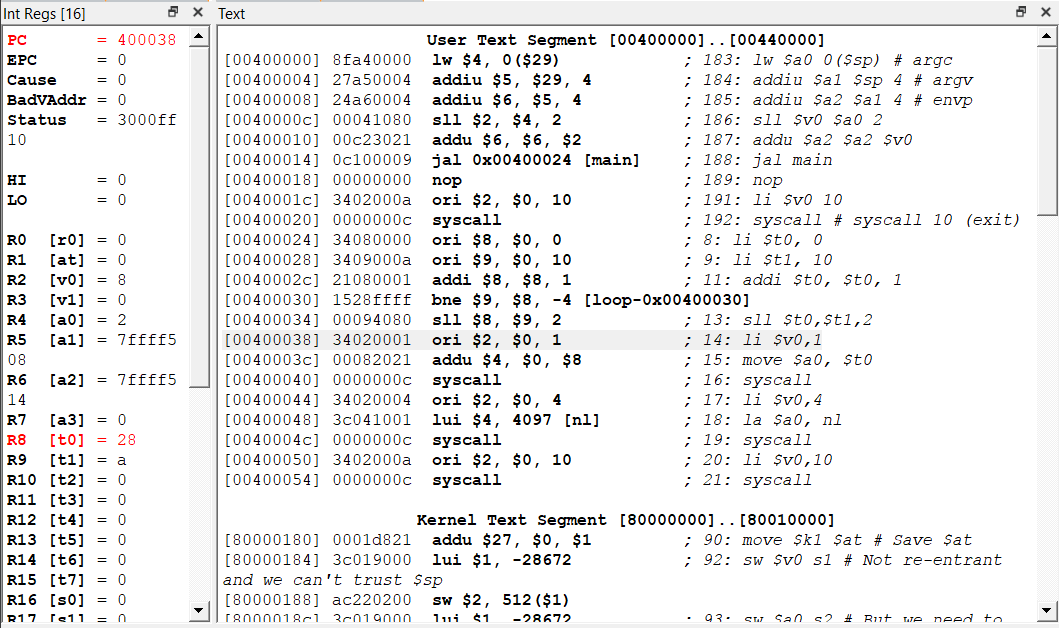
****

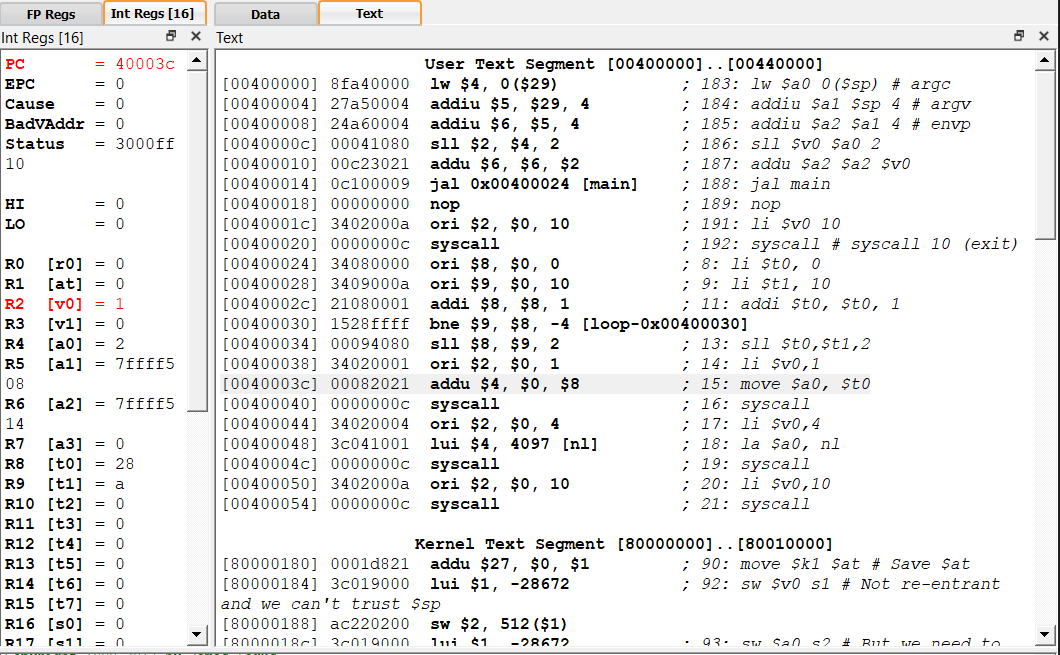
****

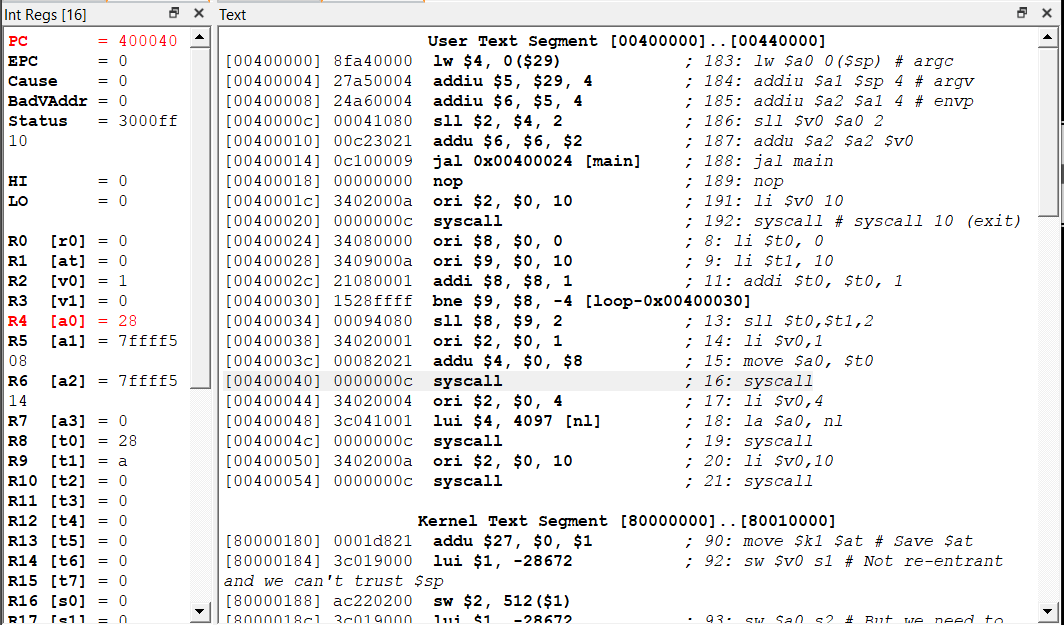
* In the next step the counter value becomes 10 and branch target will be executed.
* In the next image we can see that the count became 10 (ie a)

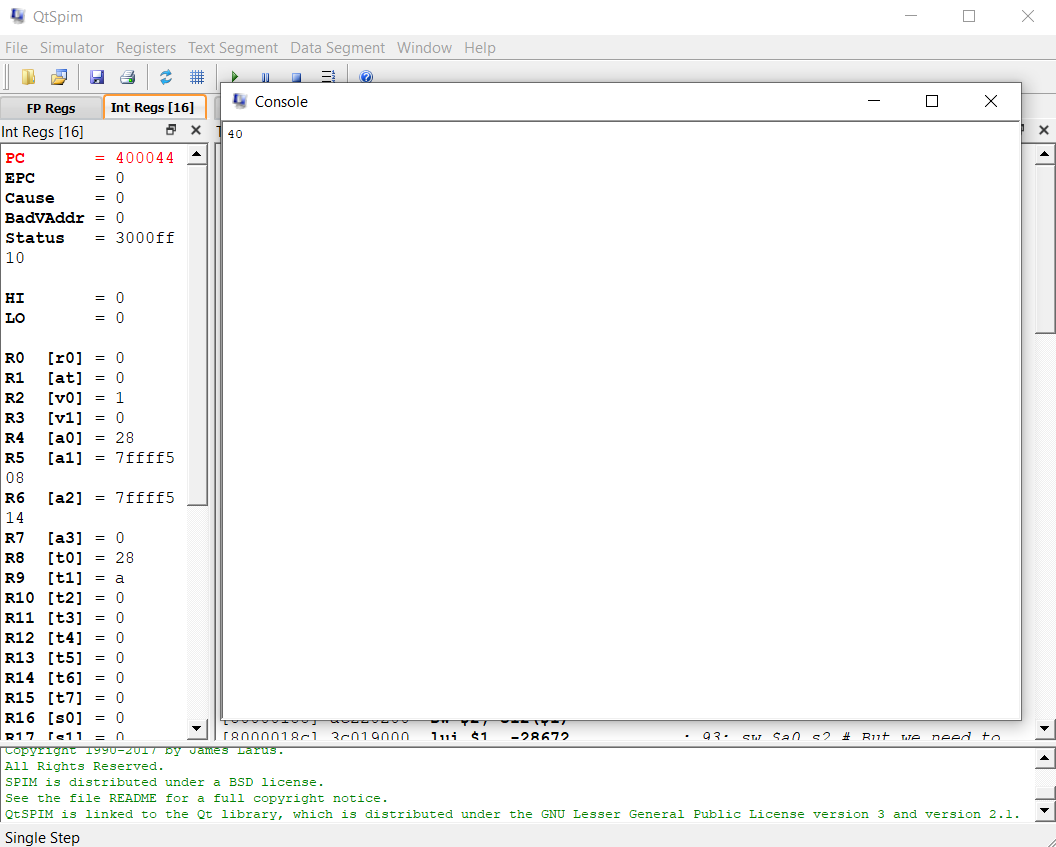
****

****

****

****

****

****