Laboratorio #3

Laboratorio #3

Walter Saldaña #19897

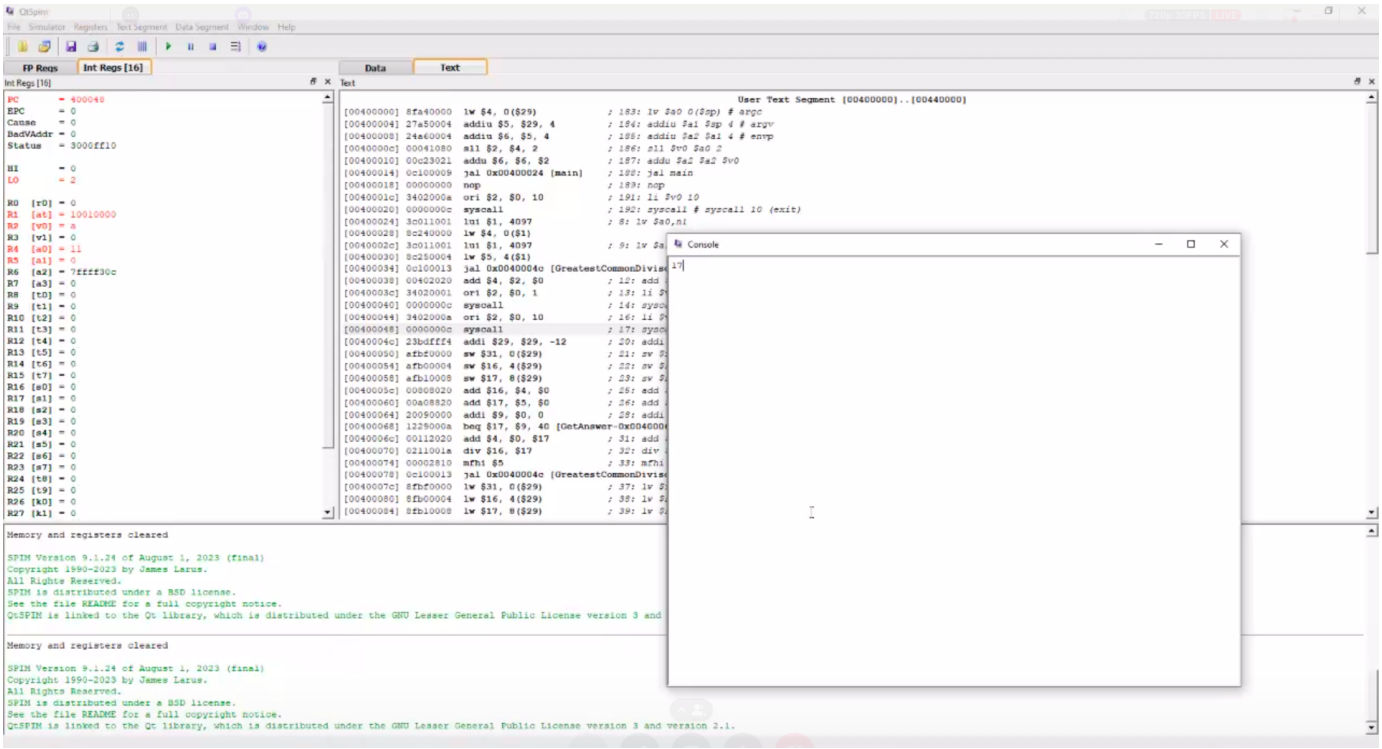
MCD en MIPS

Referencia del algoritmo usado: https://www.cs.jhu.edu/~phi/csf/slides/lecture-mips-functions.pdf

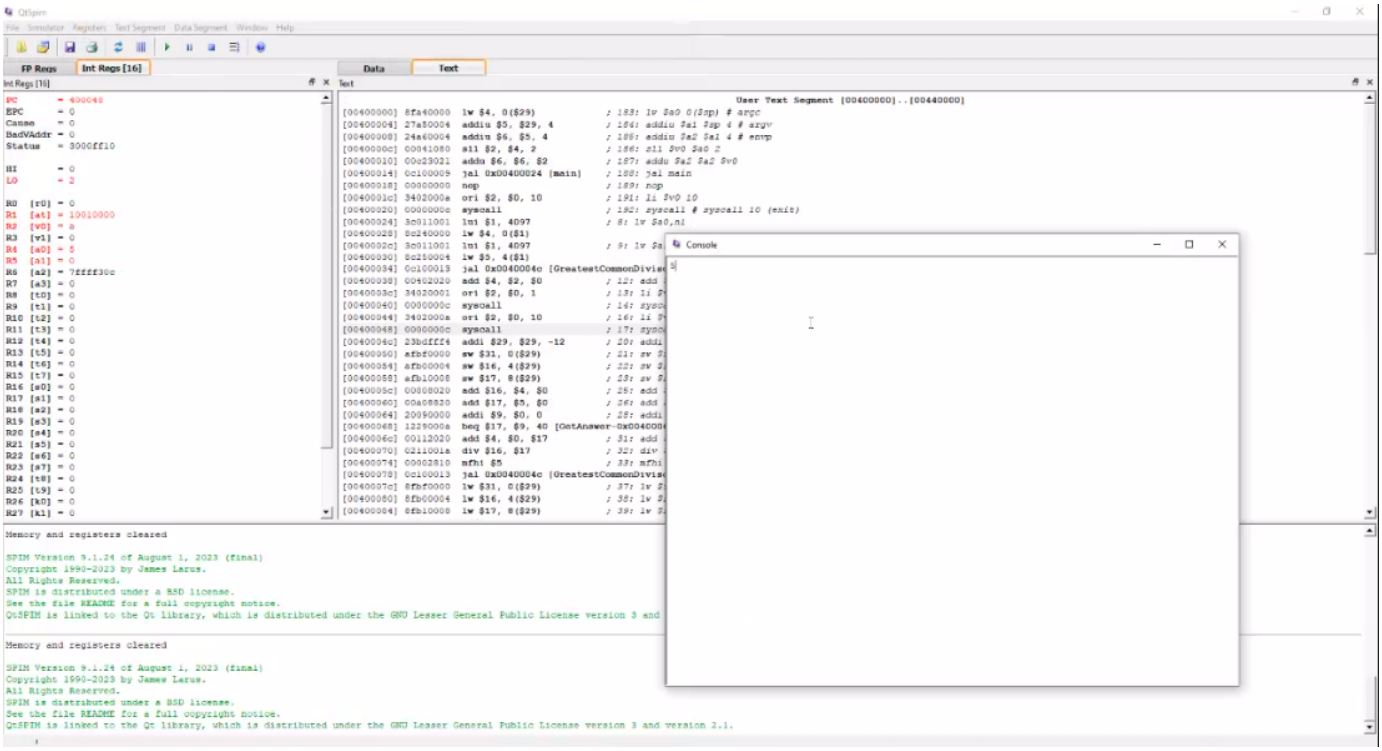
```
.data
n1: .word 99
n2: .word 101

.text
.globl main

main:
    lw $a0,n1
    lw $a1,n2
    jal GreatestCommonDivisor
    add $a0,$v0,$zero
    li $v0,1
    syscall
li $v0, 10
syscall

GreatestCommonDivisor:
    addi $sp, $sp, -12
    sw $ra, 0($sp)
    sw $s0, 4($sp)
    sw $s1, 8($sp)
    add $s0, $a0, $zero
    add $s1, $a1, $zero
    addi $t1, $zero, 0
    beq $s1, $t1, GetAnswer
    add $a0, $zero, $s1
    div $s0, $s1
    mfhi $a1
    jal GreatestCommonDivisor
```

```
ExitGCD:
    lw $ra, 0($sp)
    lw $s0, 4($sp)
    lw $s1, 8($sp)
    addi $sp, $sp, 12
    jr $ra

GetAnswer:
    add $v0, $zero, $s0
    j ExitGCD
```

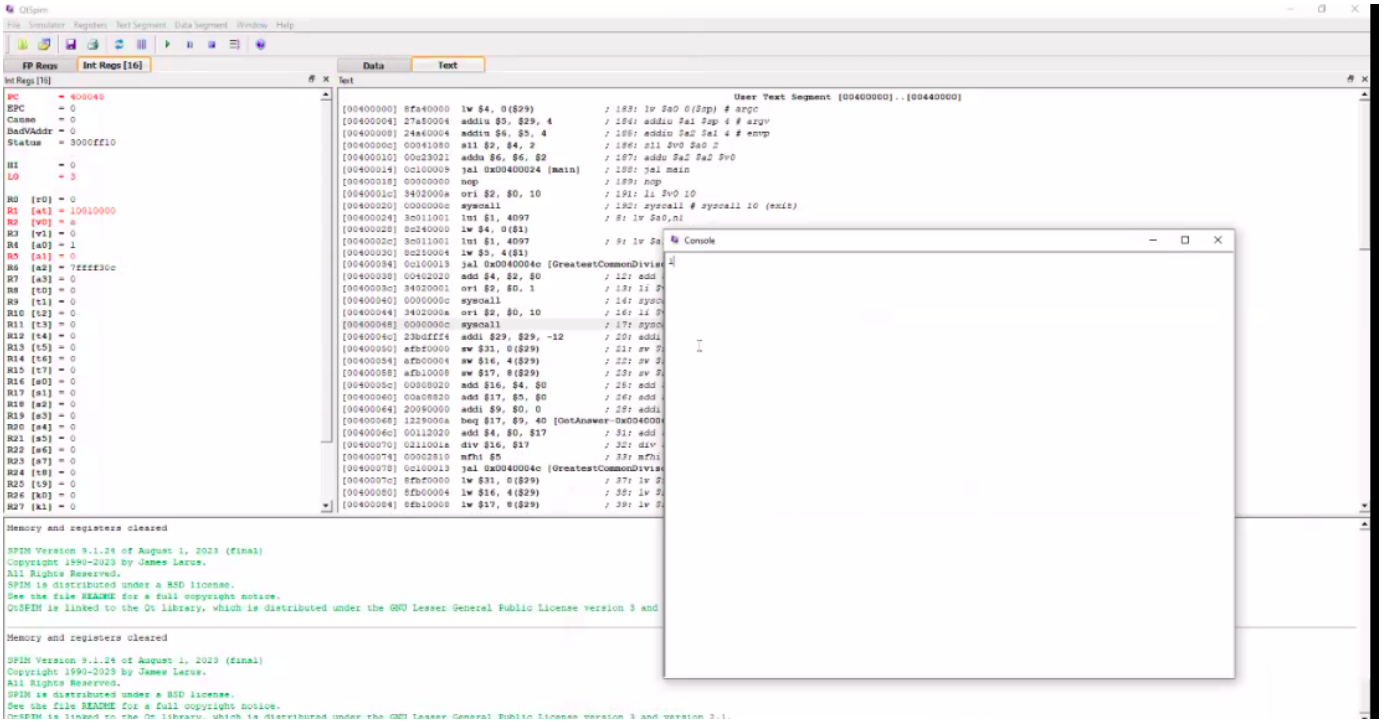
Ejecución #1 (17, 34)



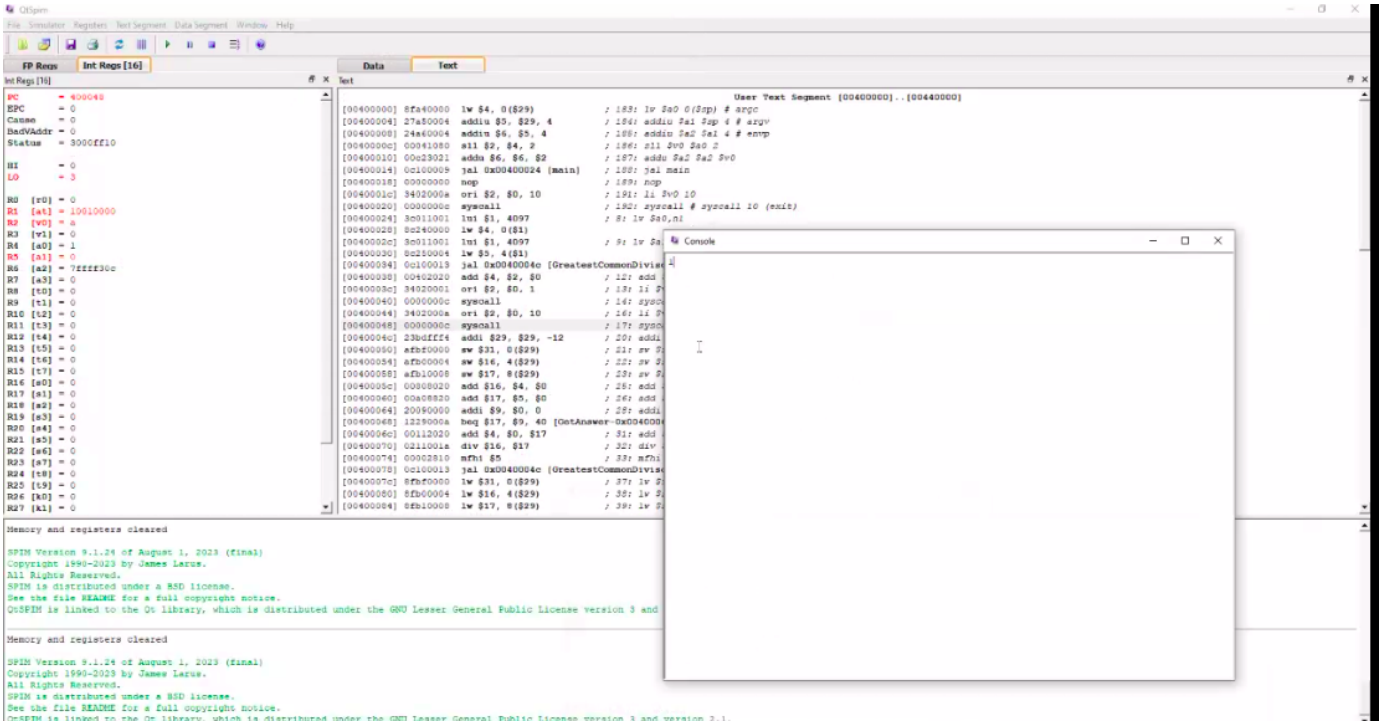
Ejecución #2 (120, 55)



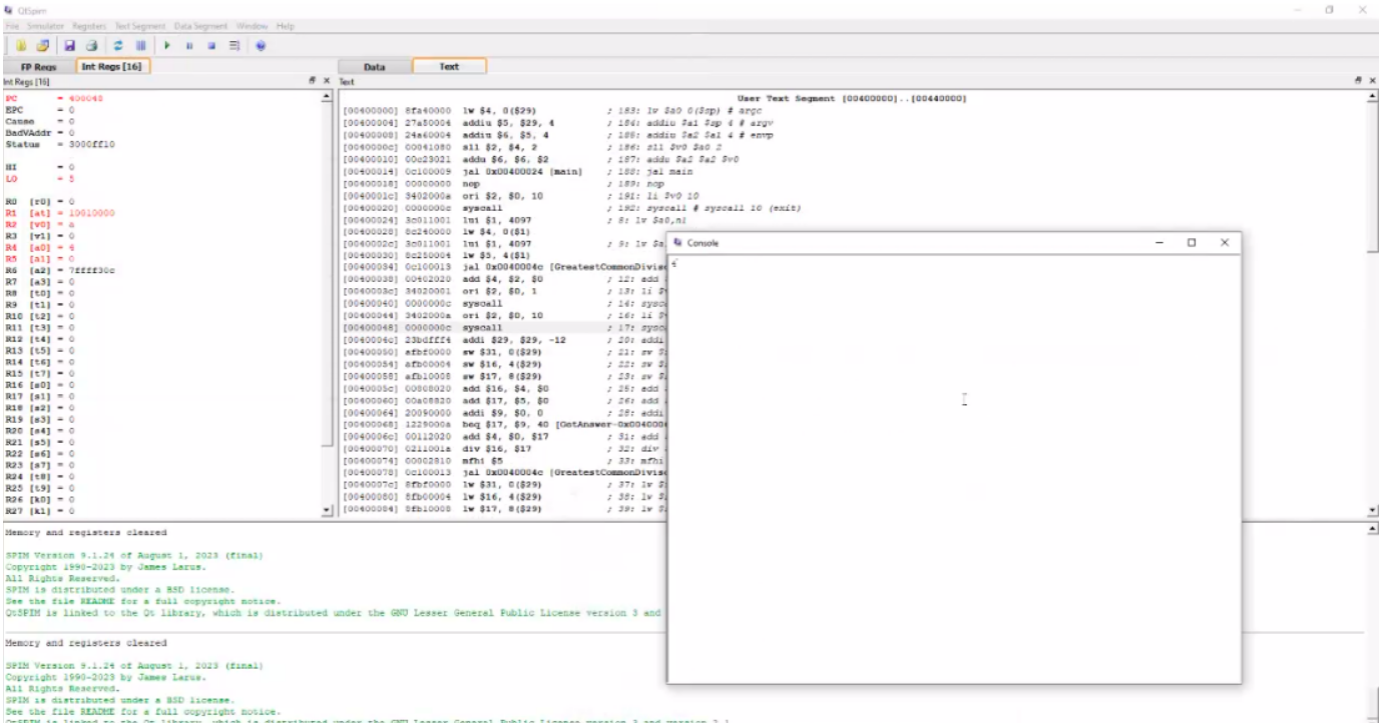
Ejecución #3 (77, 37)



Ejecución #4 (99, 301)



Ejecución #5 (404, 688)



Traducción aritmética a MIPS

Suma:

YAPL:

```
let num <- a + b
```

TAC:

```
T1 = a
T2 = b
num = T1 + T2
```

MIPS:

```
lw $t1, a
lw $t2, b
add $s0, $t1, $t2
```

Resta:

YAPL:

```
let num <- a - b
```

TAC:

```
T1 = a
T2 = b
num = T1 - T2
```

MIPS:

```
lw $t1, a
lw $t2, b
sub $s0, $t1, $t2
```

Multiplicación:

YAPL:

```
let num <- a * b
```

TAC:

```
T1 = a
T2 = b
num = T1 * T2
```

MIPS:

```
lw $t1, a
lw $t2, b
mul $s0, $t1, $t2
```

División:

YAPL:

```
let num <- a / b
```

TAC:

```
T1 = a
T2 = b
num = T1 / T2
```

MIPS:

```
lw $t1, a
lw $t2, b
div $t1, $t2
mflo $s0
```

Programa de ejemplo

```
class Main {
   main(): Object {
       let num1 : Int <- 5;
       let num2 : Int <- 7;
       let resultado : Int <- num1 + num2;
       out_string("El resultado es: ");
       out_int(resultado);
   };
};
```

TAC

```
CLASS Main

FUNC Main.main:
    num1 = 5
    num2 = 7
    T1 = num1 + num2
    resultado = T1
    CALL out_string
    CALL out_int
```

MIPS

```
.data
    resultado: .space 4
.text
    main:
        li $t0, 5
        li $t1, 7
        add $t2, $t0, $t1
        sw $t2, resultado
        li $v0, 4
        la $a0, resultado_str
        syscall
        li $v0, 1
        lw $a0, resultado
        syscall
        li $v0, 10
        syscall
```