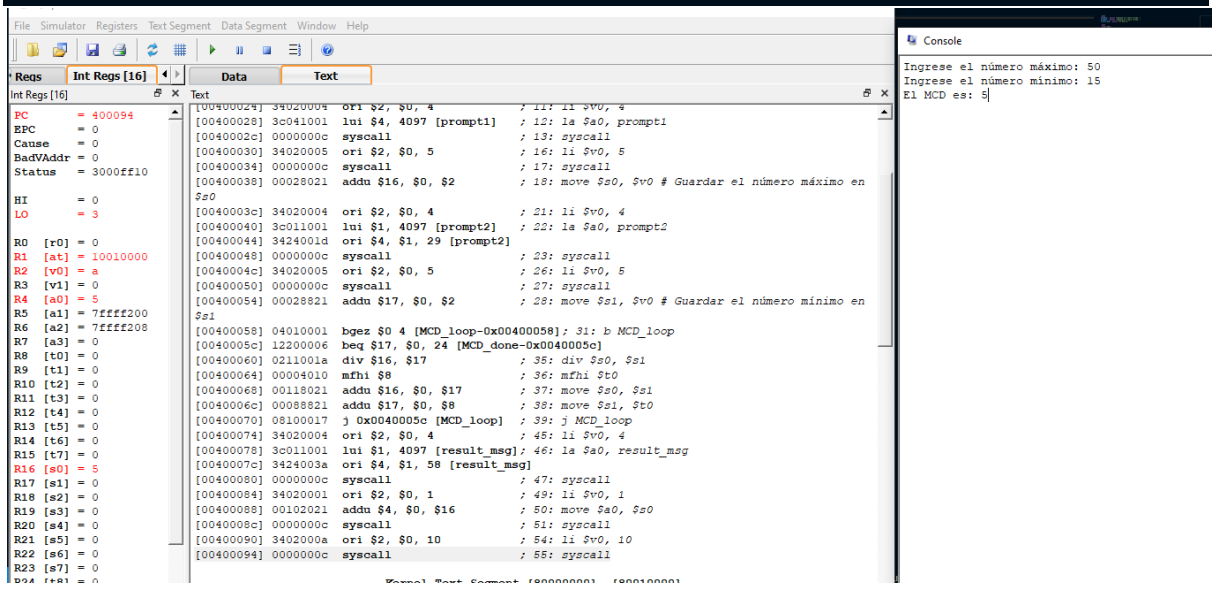
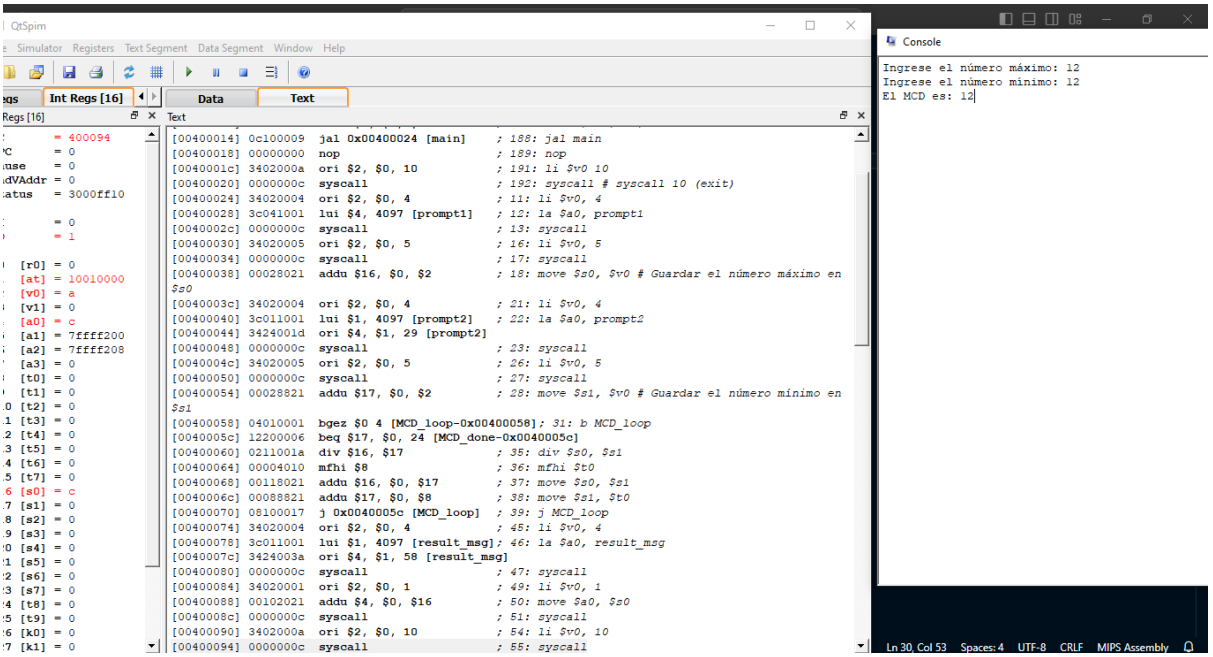
Laboratorio #3
Generación de Código Ensamblador
Traducción de Expresiones Aritméticas

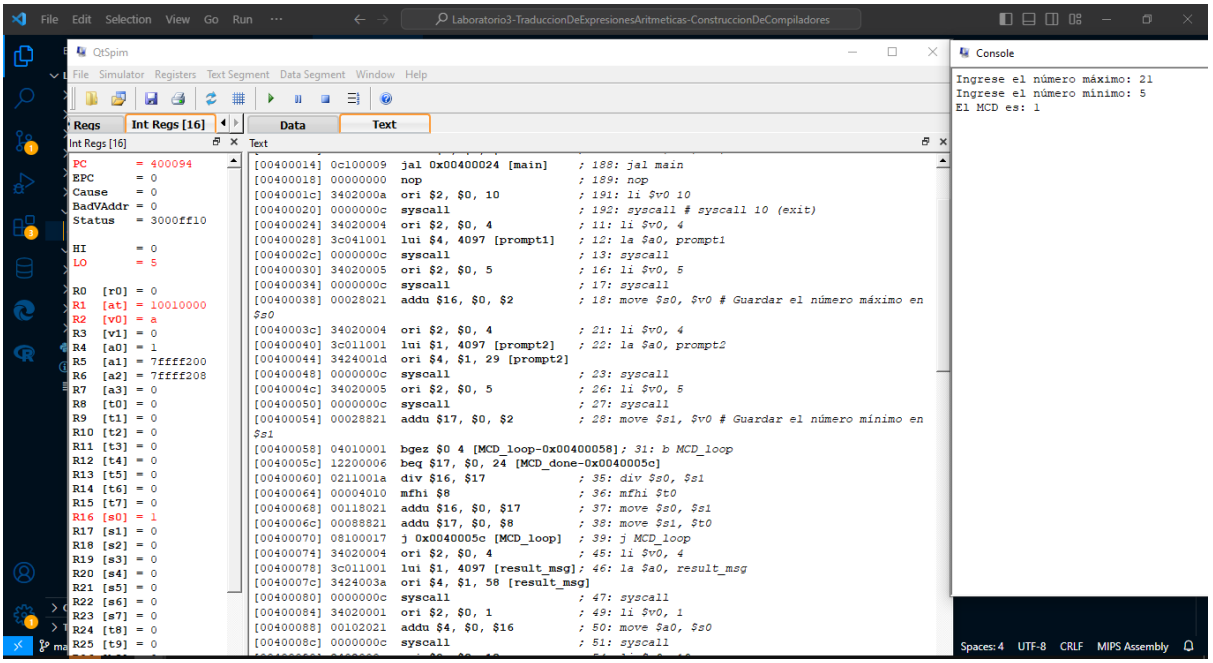
A. Generación de Código Ensamblador
Con el simulador listo, escribir un programa en MIPS que calcule el de forma recursiva el Máximo Común Divisor de dos números enteros cualesquiera.

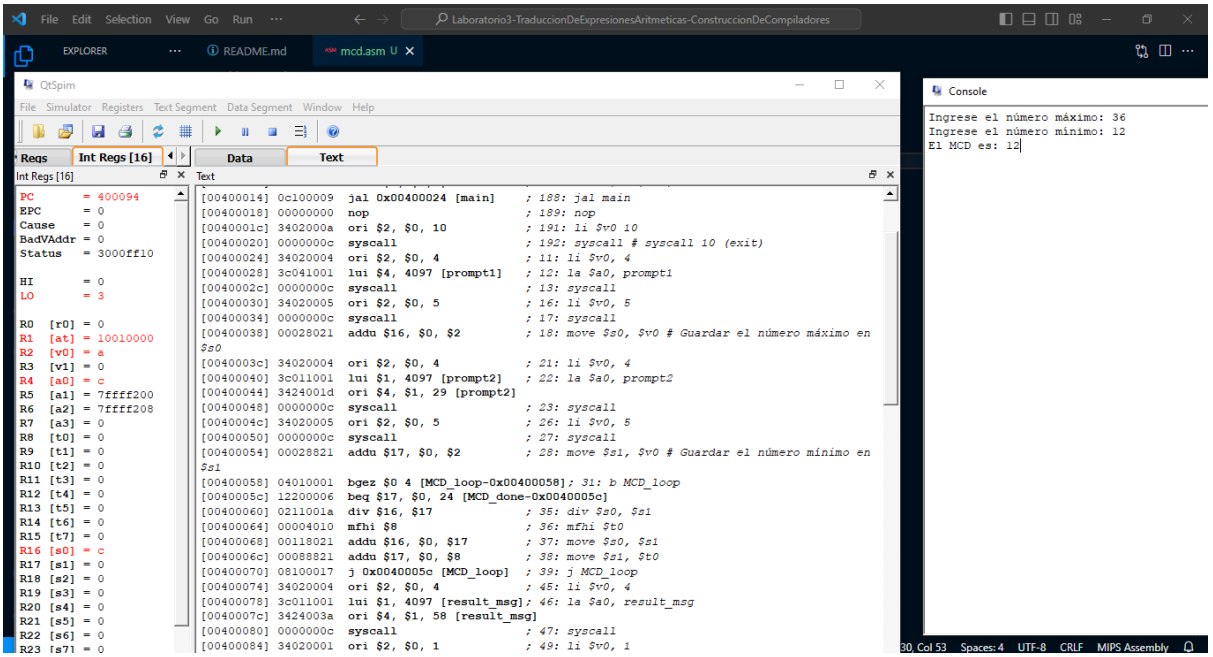
```
.data
prompt1: .asciiz "Ingrese el número máximo: "
prompt2: .asciiz "Ingrese el número mínimo: "
result_msg: .asciiz "El MCD es: "

.text
.globl main

main:
    # Preguntar al usuario que ingrese el número máximo
    li $v0, 4
    la $a0, prompt1
    syscall

    # Leer el número máximo
    li $v0, 5
    syscall
    move $s0, $v0  # Guardar el número máximo en $s0

    # Preguntar al usuario que ingrese el número mínimo
    li $v0, 4
    la $a0, prompt2
    syscall

    # Leer el número mínimo
    li $v0, 5
    syscall
    move $s1, $v0  # Guardar el número mínimo en $s1

    # Calcular el MCD usando el algoritmo de Euclides
    b MCD_loop
```

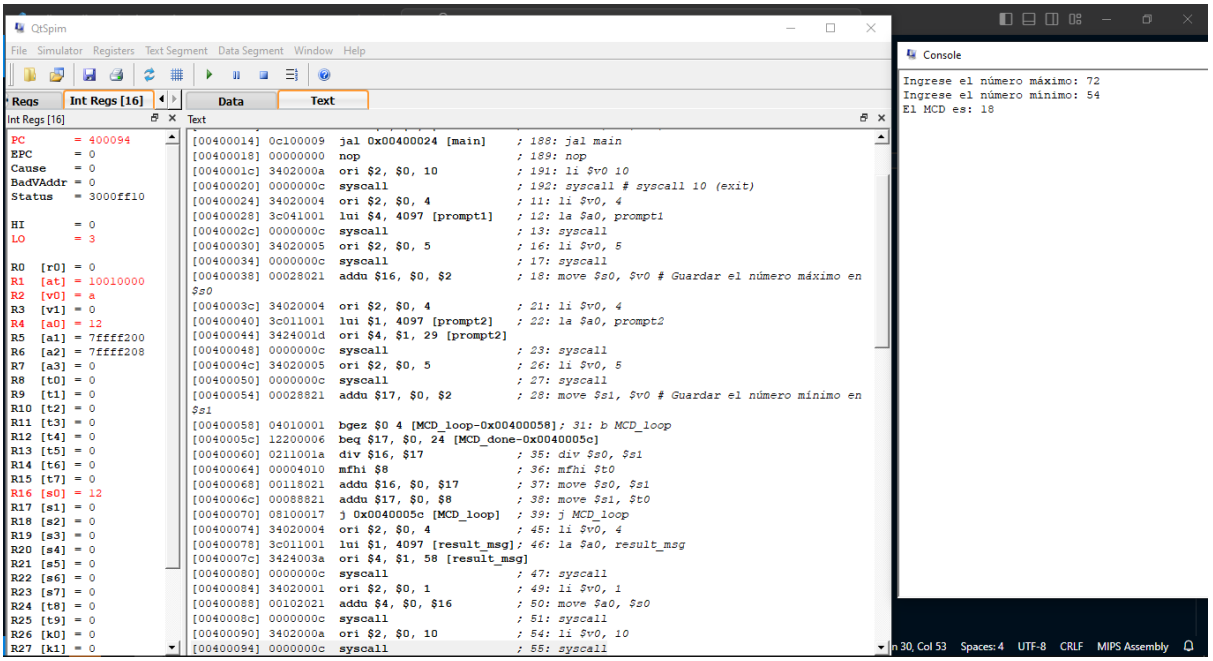
```
MCD_loop:
    beq $s1, $zero, MCD_done  # Si $s1 == 0, terminar el bucle
    div $s0, $s1
    mfhi $t0
    move $s0, $s1
    move $s1, $t0
    j MCD_loop

MCD_done:
    # El resultado (MCD) se encuentra en $s0

    # Imprimir el resultado
    li $v0, 4
    la $a0, result_msg
    syscall

    li $v0, 1
    move $a0, $s0
    syscall

    # Salir del programa
    li $v0, 10
    syscall
```




QtSpim
Simulator Registers Text Segment Data Segment Window Help
Int Regs [16]
Data
Text
Regs [16]
Text
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 34020004 ori $2, $0, 4 ; 11: li $v0, 4
[00400028] 3c041001 lui $4, 4097 [prompt1] ; 12: la $a0, prompt1
[0040002c] 0000000c syscall ; 13: syscall
[00400030] 34020005 ori $2, $0, 5 ; 16: li $v0, 5
[00400034] 0000000c syscall ; 17: syscall
[00400038] 00028021 addu $16, $0, $2 ; 18: move $s0, $v0 # Guardar el número máximo en $s0
[0040003c] 34020004 ori $2, $0, 4 ; 21: li $v0, 4
[00400040] 3c011001 lui $1, 4097 [prompt2] ; 22: la $a0, prompt2
[00400044] 3424001d ori $4, $1, 29 [prompt2]
[00400048] 0000000c syscall ; 23: syscall
[0040004c] 34020005 ori $2, $0, 5 ; 26: li $v0, 5
[00400050] 0000000c syscall ; 27: syscall
[00400054] 00028821 addu $17, $0, $2 ; 28: move $s1, $v0 # Guardar el número mínimo en $s1
[00400058] 04010001 bgez $0 4 [MCD_loop-0x00400058] ; 31: b MCD_loop
[0040005c] 12200006 beq $17, $0, 24 [MCD_done-0x0040005c]
[00400060] 0211001a div $16, $17 ; 35: div $s0, $s1
[00400064] 00004010 mfhi $8 ; 36: mfhi $t0
[00400068] 00118021 addu $16, $0, $17 ; 37: move $s0, $s1
[0040006c] 00088821 addu $17, $0, $8 ; 38: move $s1, $t0
[00400070] 08100017 j 0x0040005c [MCD_loop] ; 39: j MCD_loop
[00400074] 34020004 ori $2, $0, 4 ; 45: li $v0, 4
[00400078] 3c011001 lui $1, 4097 [result_msg] ; 46: la $a0, result_msg
[0040007c] 3424003a ori $4, $1, 58 [result_msg]
[00400080] 0000000c syscall ; 47: syscall
[00400084] 34020001 ori $2, $0, 1 ; 49: li $v0, 1
[00400088] 00102021 addu $4, $0, $16 ; 50: move $a0, $s0
[0040008c] 0000000c syscall ; 51: syscall
[00400090] 3402000a ori $2, $0, 10 ; 54: li $v0, 10
[00400094] 0000000c syscall ; 55: syscall
Console
Ingrese el número máximo: 12
Ingrese el número mínimo: 12
El MCD es: 12
Ln 30, Col 53 Spaces: 4 UTF-8 CRLF MIPS Assembly

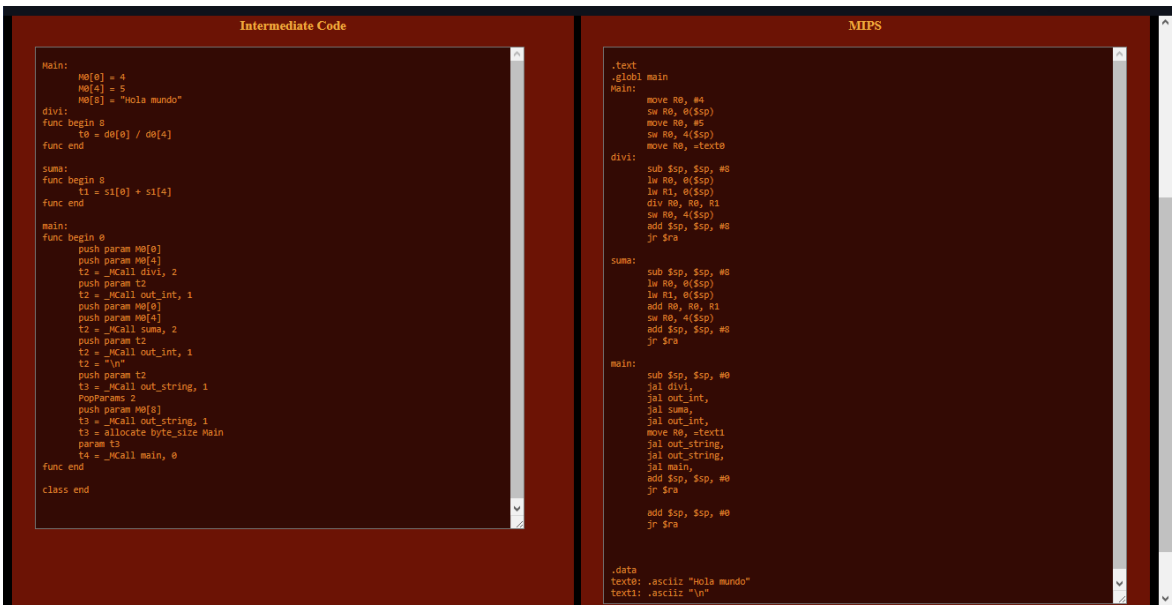

File Edit Selection View Go Run
Laboratorio3-TraduccionDeExpresionesAritmeticas-ConstruccionDeCompiladores
QtSpim
Simulator Registers Text Segment Data Segment Window Help
Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
PC = 400094
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 5
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7fffff200
R6 [a2] = 7fffff208
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 1
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 34020004 ori $2, $0, 4 ; 11: li $v0, 4
[00400028] 3c041001 lui $4, 4097 [prompt1] ; 12: la $a0, prompt1
[0040002c] 0000000c syscall ; 13: syscall
[00400030] 34020005 ori $2, $0, 5 ; 16: li $v0, 5
[00400034] 0000000c syscall ; 17: syscall
[00400038] 00028021 addu $16, $0, $2 ; 18: move $s0, $v0 # Guardar el número máximo en $s0
[0040003c] 34020004 ori $2, $0, 4 ; 21: li $v0, 4
[00400040] 3c011001 lui $1, 4097 [prompt2] ; 22: la $a0, prompt2
[00400044] 3424001d ori $4, $1, 29 [prompt2]
[00400048] 0000000c syscall ; 23: syscall
[0040004c] 34020005 ori $2, $0, 5 ; 26: li $v0, 5
[00400050] 0000000c syscall ; 27: syscall
[00400054] 00028821 addu $17, $0, $2 ; 28: move $s1, $v0 # Guardar el número mínimo en $s1
[00400058] 04010001 bgez $0 4 [MCD_loop-0x00400058] ; 31: b MCD_loop
[0040005c] 12200006 beq $17, $0, 24 [MCD_done-0x0040005c]
[00400060] 0211001a div $16, $17 ; 35: div $s0, $s1
[00400064] 00004010 mfhi $8 ; 36: mfhi $t0
[00400068] 00118021 addu $16, $0, $17 ; 37: move $s0, $s1
[0040006c] 00088821 addu $17, $0, $8 ; 38: move $s1, $t0
[00400070] 08100017 j 0x0040005c [MCD_loop] ; 39: j MCD_loop
[00400074] 34020004 ori $2, $0, 4 ; 45: li $v0, 4
[00400078] 3c011001 lui $1, 4097 [result_msg] ; 46: la $a0, result_msg
[0040007c] 3424003a ori $4, $1, 58 [result_msg]
[00400080] 0000000c syscall ; 47: syscall
[00400084] 34020001 ori $2, $0, 1 ; 49: li $v0, 1
[00400088] 00102021 addu $4, $0, $16 ; 50: move $a0, $s0
[0040008c] 0000000c syscall ; 51: syscall
Console
Ingrese el número máximo: 21
Ingrese el número mínimo: 5
El MCD es: 1
Spaces: 4 UTF-8 CRLF MIPS Assembly





B. Traducción de Expresiones Aritméticas

1. Para las operaciones de suma y división, dentro de sus expresiones aritméticas, realizar la traducción a instrucciones MIPS equivalentes (que cumplan la estructura y semántica de MIPS)




```
class Main inherits IO {
  a: Int <- 4;
  b: Int <- 5;
  x: String <- "Hola mundo";

  divi(y : Int, k : Int) : Int {
        y / k
  };

  suma(y : Int, k : Int) : Int {
        y + k
  };
  main() : Object {
    {
      out_int(divi(a, b));
      out_int(suma(a, b));
      out_string("\n");
      out_string(x);
(new Main).main();
    }
  };
};
```

```
 Main:
    M0[0] = 4
    M0[4] = 5
    M0[8] = "Hola mundo"
```

```
divi:
func begin 8
   t0 = d0[0] / d0[4]
func end

suma:
func begin 8
   t1 = s1[0] + s1[4]
func end

main:
func begin 0
   push param M0[0]
   push param M0[4]
   t2 = _MCall divi, 2
   push param t2
   t2 = _MCall out_int, 1
   push param M0[0]
   push param M0[4]
   t2 = _MCall suma, 2
   push param t2
   t2 = _MCall out_int, 1
   t2 = "\n"
   push param t2
   t3 = _MCall out_string, 1
   PopParams 2
   push param M0[8]
   t3 = _MCall out_string, 1
   t3 = allocate byte_size Main
   param t3
   t4 = _MCall main, 0
func end

class end
```

```
.text
.globl main
Main:
   move R0, #4
```

```
    sw R0, 0($sp)
    move R0, #5
    sw R0, 4($sp)
    move R0, =text0
divi:
    sub $sp, $sp, #8
    lw R0, 0($sp)
    lw R1, 0($sp)
    div R0, R0, R1
    sw R0, 4($sp)
    add $sp, $sp, #8
    jr $ra

suma:
    sub $sp, $sp, #8
    lw R0, 0($sp)
    lw R1, 0($sp)
    add R0, R0, R1
    sw R0, 4($sp)
    add $sp, $sp, #8
    jr $ra

main:
    sub $sp, $sp, #0
    jal divi,
    jal out_int,
    jal suma,
    jal out_int,
    move R0, =text1
    jal out_string,
    jal out_string,
    jal main,
    add $sp, $sp, #0
    jr $ra

    add $sp, $sp, #0
    jr $ra



.data
text0: .asciiz "Hola mundo"
text1: .asciiz "\n"
```

[redacted]