MIPS Assembly Language Assignment, Oliver Heib 17003448

# Task 1

# Task 1.A

## Code (v1.1)

.data

Name:

.space 64 #initiates Name and restrict it to 64 characters

.text

#inputting name

li $v0, 8 #syscall number for inputting a string

la $a0, Name #sets the input to the Name variable

li $a1, 64 #restricts the input to 64 characters

syscall

add $s0, $a0, $zero #copies the input to another register

#inputting ID

li $v0, 5 #syscall number for inputting an integer

syscall

add $s1, $v0, $zero #copies the input to another register

#outputting name

add $a0, $s0, $zero #putting the users name in the correct register for outputting

li $v0, 4 #syscall number for outputting an integer

syscall

#outputting ID

add $a0, $s1, $zero #putting the users ID in the correct register for outputting

li $v0, 1 #syscall number for outputting a string

syscall

## Screenshots

### Inputting name

# 

# 

### Inputting ID

## Output

## Description

The first text selection inputs the name by putting the syscall number in the relevant register and the maximum length of the input is put into the relevant register as well. If the input is too big the rest of the input after is left out. The input is then copied from the register the syscall places it into the register $s0. The second section inputs ID by putting the syscall number in the relevant register and then copying the value from where the syscall placed the value into the register $s1. The third section the name is outputted by copying the value of $s0 in the relevant register for output. The syscall value is also put in the relevant register. The third section does the same but with register $s1 and the syscall number for outputting an integer.

# Task 1.B

## Code (v1.1)

.text

addi $s0, $zero, 2 #initiating division number

#inputting ID

li $v0, 5 #syscall number for inputting an integer

syscall

add $s1, $v0, $zero #copying the input to another register

#dividing by 2

div $s1, $s0 #dividing input by 2

#outputting ID

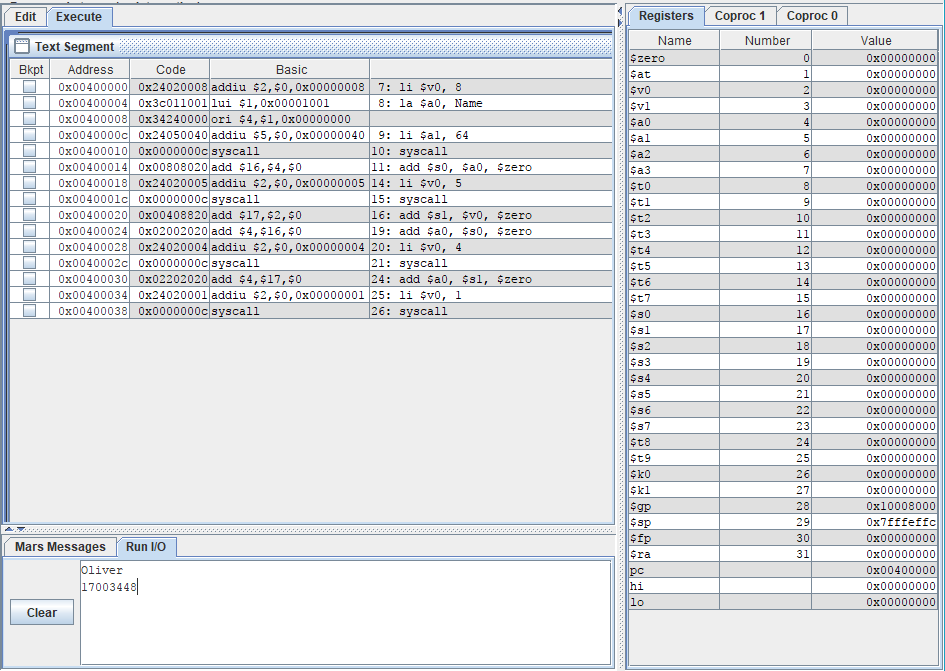
mflo $a0 # moving the quotient to another register

li $v0, 1 #syscall number for outputting an integer

Syscall

## Screenshots

### Input



### Output

## Description

The first section puts the value for dividing (2) in a register. The second section puts the relevant syscall number in the correct register to allows the user to input an integer. The program then copies it to another register. In the third section the program divides the inputted value by 2 which is initiated in section 1. And then as the result is placed in the hi register it has to be copied to the correct register for outputting. The relevant syscall number is loaded and the result is outputted.

# Task 1.C

## Flowchart

## 

## Code (v1.15)

.text

#inputting ID

li $v0, 5 #syscall number for inputting an integer

syscall

add $t0, $v0, $zero #copying the input to another register

li $t3, 160 #setting the number of times loop2 will loop when it is called

Loop3:

li $t2, 999 #setting the number of times loop2 will loop when it is called

addi $t3 , $t3, -1 #counting down loops for loop2

loop2:

li $t1, 999 #setting the number of times loop1 will loop when it is called

addi $t2 , $t2, -1 #counting down loops for loop2

loop1:

addi $t1 , $t1, -1 #counting down loops for loop1

add $s0, $t0,$t0 #adding the two number

#end of loop1

bgez $t2, loop2 #if the loop2 counter is not 0 then it branches to loop2

#end of loop2

bgez $t3, loop2 #if the loop2 counter is not 0 then it branches to loop2

#end of loop3

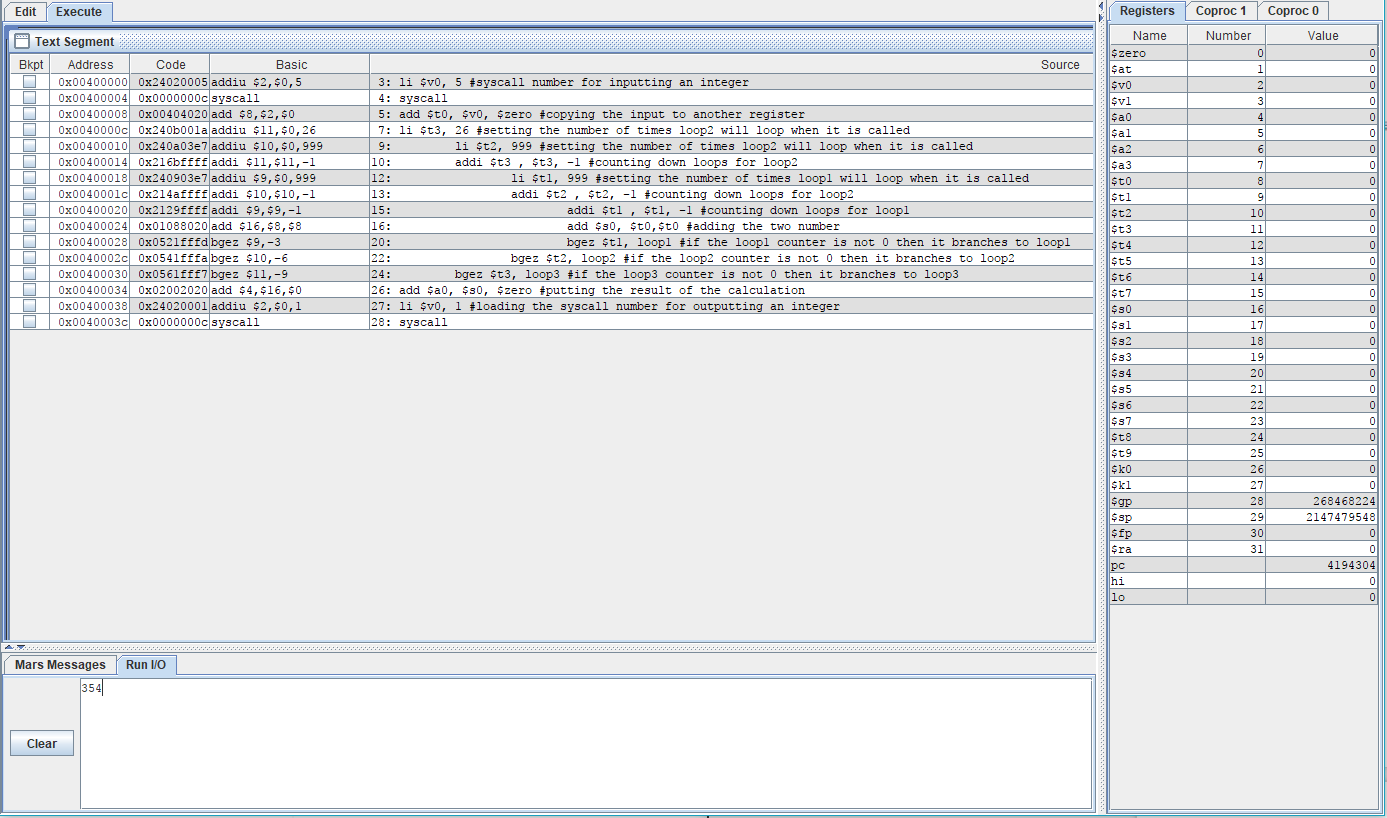
add $a0, $s0, $zero #putting the result of the calculation in the correct register for outputting

li $v0, 1 #loading the syscall number for outputting an integer

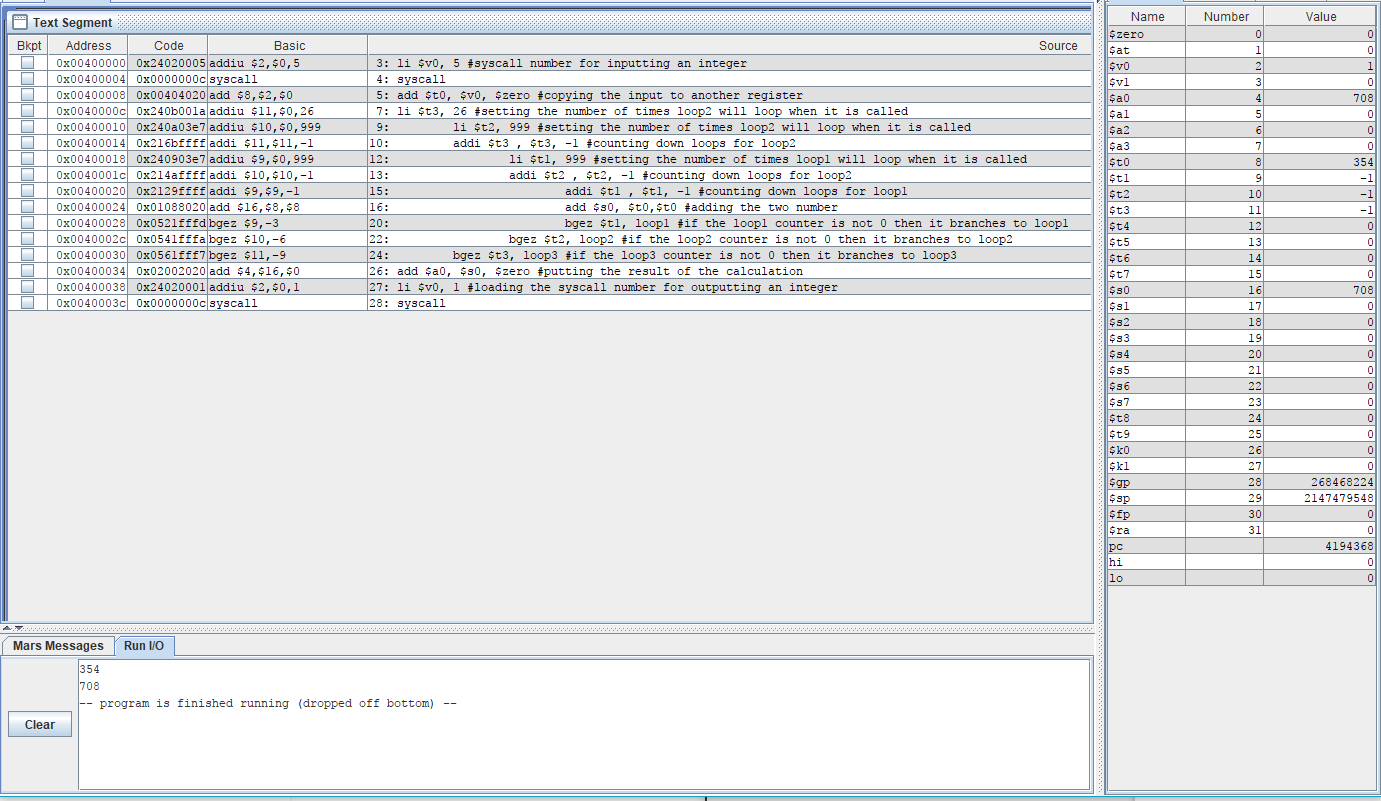
syscall

## Screenshots

### Input



### Output



### System Specifications

## Description

Above the syscall number for inputting is put in the correct register for arguments and then the syscall is called. The input is then set to the number for adding together and given its own register. The number of loops for loop3 is initiated, given its own register and then starts loop 3. The number of iterations of loop2 are initiated and given their own register. The number of loops for loop 3 is decremented. Loop2 starts off with setting the number of iterations for loop1, giving it its own register and then decrementing the loops for loop2. Loop1 starts with decrementing the loops for loop1 and then adding the two numbers together and placing it in the correct register for outputting. The next two lines put the correct syscall number for outputting an integer in the argument register and calls the syscall. Loop1 then checks whether the remaining loops for loop1 is not 0. If this is true the program then branches to the start of loop1 starting the loop again. If not then the program drops out of the loop into the rest of loop2. The end of loop 2 does the same checking for the remaining loops for loop2. If the reming loops is not 0 the program then starts loop 2 again witch will in turn reset the amount of loops for loop 1 starting all iterations of loop 1 again. If the number of loops is 0 the program the dropsa out to the rest of loop3. At the end of loop 3 the same thing happens again branching to the top of loop 3 running loop2 each time running loop 1 each time. If the remaining loops is 0 the program drops out of the loop where the program will drop off the bottom.