Experiment 1 Machine Language Instructions

Lab Instructor: Christos M.

Date Performed: 02/01/17

Date Submitted: 02/15/17

Name: Leonardo Roman lr534

Purpose:

The goal of this lab is to exercise the some of the basic operations of assembly language using QtSpim.

Assignment 1:

For the instructions below, indicate what the corresponding Machine Language Instructions in Binary codes are in the table below.

Instruction	op	rs	rt	rd	shamt	funct
add \$9, \$10, \$9	000000	01010	01001	01001	00000	100000
					Constant	
addi \$10,\$10, -1	001000	01010	01010		1111111111111111	

Converting -1 into 16-bit binary number

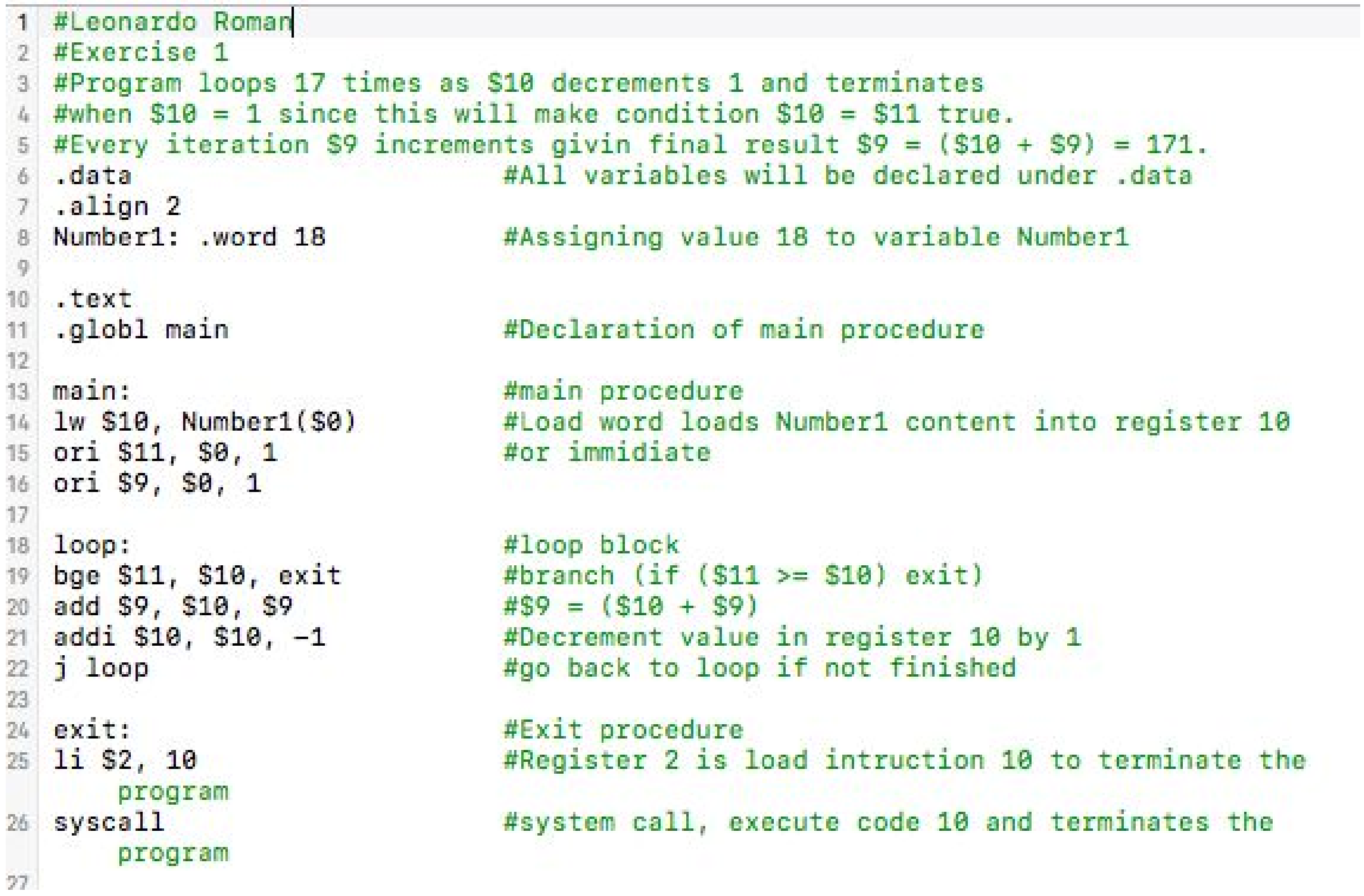
$0000\ 0000\ 0000\ 0001_2 \rightarrow 1111\ 1111\ 1111\ 1110_2 + 1 = 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

Assignment 2:

Writing comments to well explain the functionality of the program

a) Put the comments to the end of each line after a sharp sign (#), for example, li $2, 10 #Place (load immediate) number 10 into register $2.

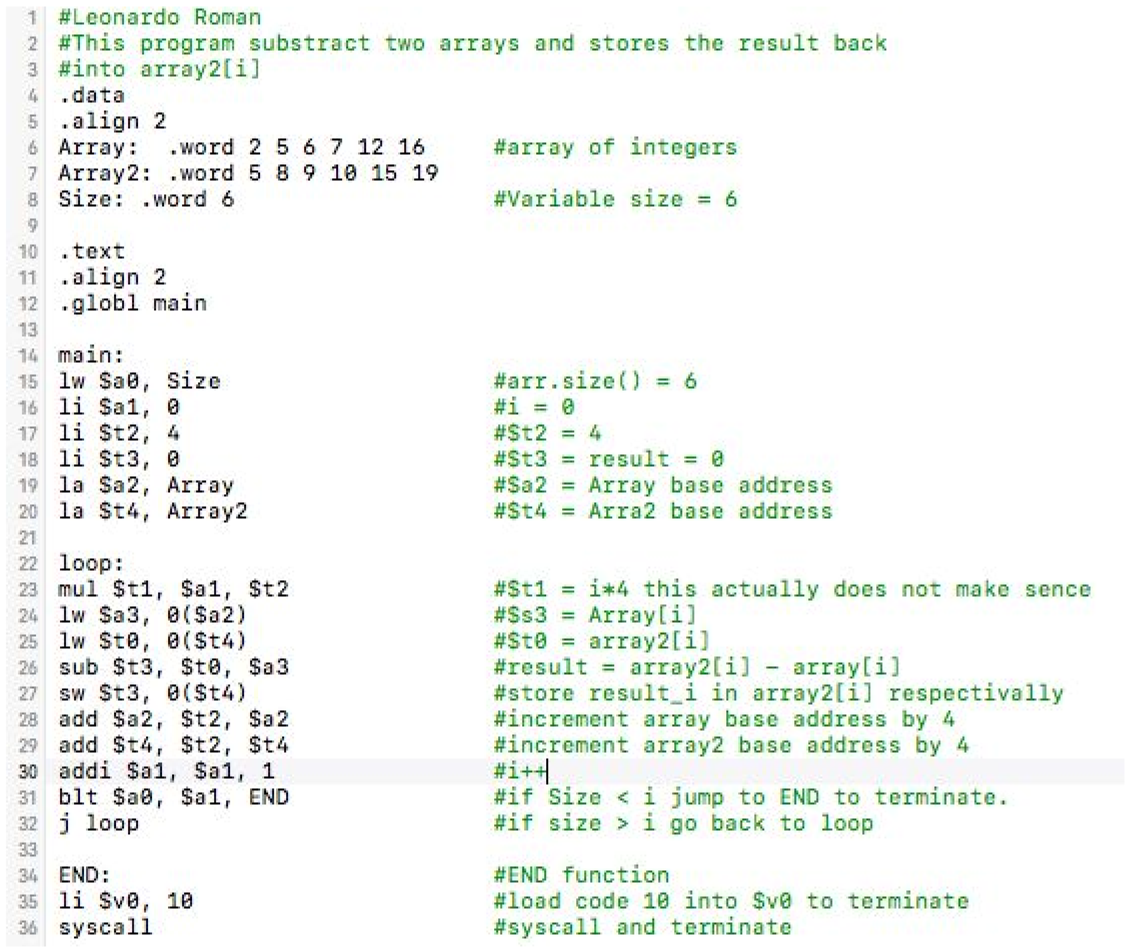
b) Include a preamble with comments explaining what the program does. Do not write more than 3 to 4 lines.



```
#Leonardo Roman
#Exercise 1
#Program loops 17 times as $10 decrements 1 and terminates
#when $10 = 1 since this will make condition $10 = $11 true.
#Every iteration $9 increments givin final result $9 = ($10 + $9) = 171.
.data                           #All variables will be declared under .data
.align 2
Number1: .word 18               #Assigning value 18 to variable Number1

.text
.globl main                     #Declaration of main procedure

main:                           #main procedure
lw $10, Number1($0)             #Load word loads Number1 content into register 10
ori $11, $0, 1                  #or immidiate
ori $9, $0, 1

loop:                           #loop block
bge $11, $10, exit              #branch (if ($11 >= $10) exit)
add $9, $10, $9                 #$9 = ($10 + $9)
addi $10, $10, -1               #Decrement value in register 10 by 1
j loop                          #go back to loop if not finished

exit:                           #Exit procedure
li $2, 10                       #Register 2 is load intruction 10 to terminate the
    program
syscall                         #system call, execute code 10 and terminates the
    program
```

Assignment 3:

1. Fill out the missing instructions according to the comments.



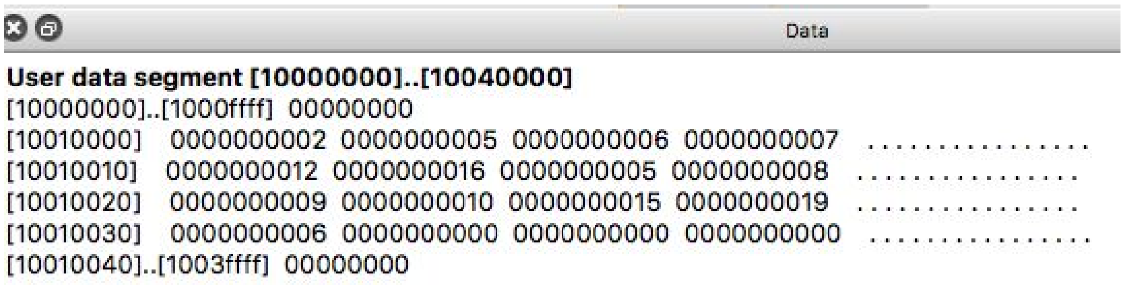
```
#Leonardo Roman
#This program substract two arrays and stores the result back
#into array2[i]
.data
.align 2
Array:  .word 2 5 6 7 12 16     #array of integers
Array2: .word 5 8 9 10 15 19
Size: .word 6                   #Variable size = 6

.text
.align 2
.globl main

main:
lw $a0, Size                    #arr.size() = 6
li $a1, 0                       #i = 0
li $t2, 4                       #$t2 = 4
li $t3, 0                       #$t3 = result = 0
la $a2, Array                   #$a2 = Array base address
la $t4, Array2                  #$t4 = Arra2 base address

loop:
mul $t1, $a1, $t2               #$t1 = i*4 this actually does not make sence
lw $a3, 0($a2)                  #$s3 = Array[i]
lw $t0, 0($t4)                  #$t0 = array2[i]
sub $t3, $t0, $a3               #result = array2[i] - array[i]
sw $t3, 0($t4)                  #store result_i in array2[i] respectively
add $a2, $t2, $a2               #increment array base address by 4
add $t4, $t2, $t4               #increment array2 base address by 4
addi $a1, $a1, 1                #i++
blt $a0, $a1, END               #if Size < i jump to END to terminate.
j loop                          #if size > i go back to loop

END:                            #END function
li $v0, 10                      #load code 10 into $v0 to terminate
syscall                         #syscall and terminate
```

2. Run the program step by step by pressing the function key f10. Observe the change of each register and each memory place at each step. Record the registers at the end of the program.

$a0	$a1	$t2	$t3	$a2	$t4	$a3	$t0
6	7	4	3	1001001c	10010034	3	6

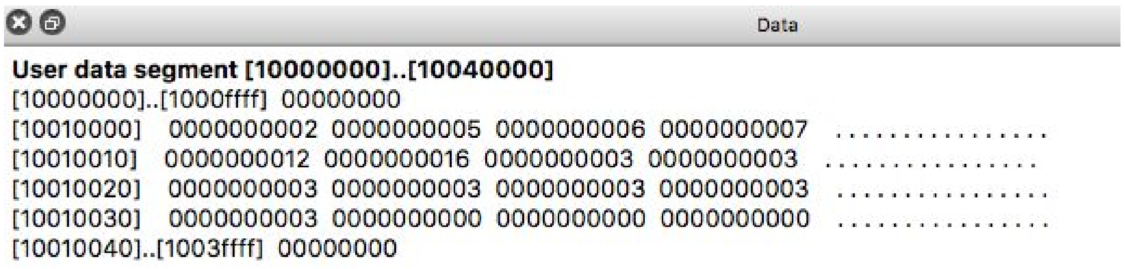
3. Record the data segment of this program and check the output.

Before



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]   0000000002  0000000005  0000000006  0000000007   ................
[10010010]   0000000012  0000000016  0000000005  0000000008   ................
[10010020]   0000000009  0000000010  0000000015  0000000019   ................
[10010030]   0000000006  0000000000  0000000000  0000000000   ................
[10010040]..[1003ffff]  00000000
```

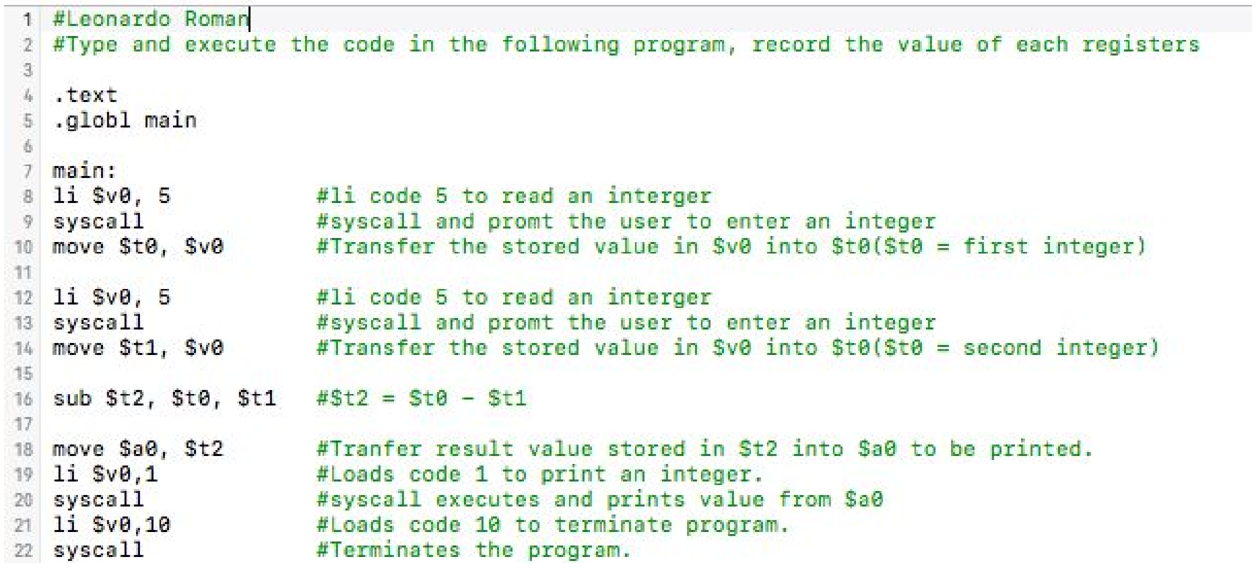
After



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]   0000000002  0000000005  0000000006  0000000007   ................
[10010010]   0000000012  0000000016  0000000003  0000000003   ................
[10010020]   0000000003  0000000003  0000000003  0000000003   ................
[10010030]   0000000003  0000000000  0000000000  0000000000   ................
[10010040]..[1003ffff]  00000000
```

Assignment 4:

1- Type and execute the code in following program, record the value of each register.

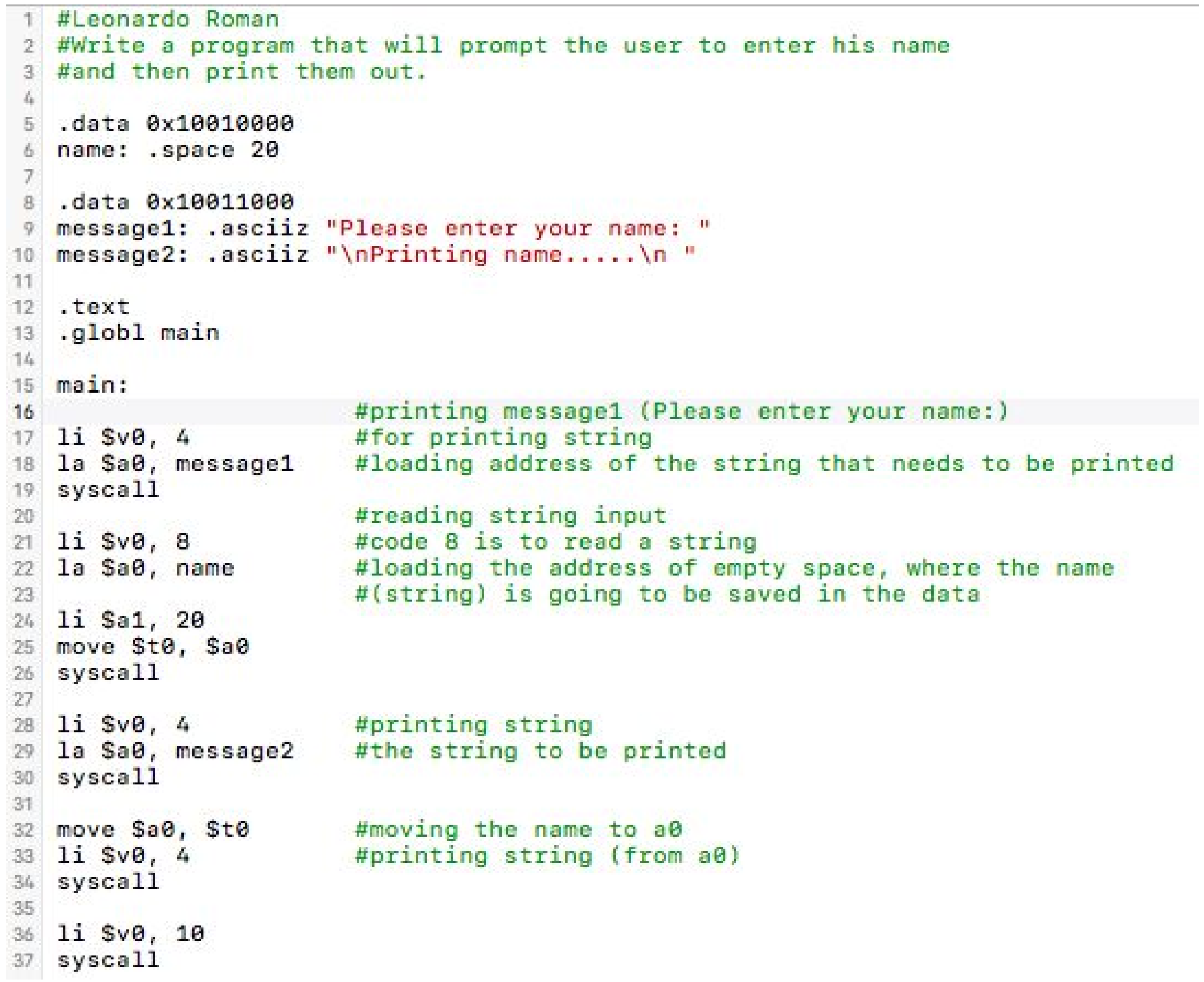


```
#Leonardo Roman
#Type and execute the code in the following program, record the value of each registers

.text
.globl main

main:
li $v0, 5           #li code 5 to read an interger
syscall             #syscall and promt the user to enter an integer
move $t0, $v0       #Transfer the stored value in $v0 into $t0($t0 = first integer)

li $v0, 5           #li code 5 to read an interger
syscall             #syscall and promt the user to enter an integer
move $t1, $v0       #Transfer the stored value in $v0 into $t0($t0 = second integer)

sub $t2, $t0, $t1   #$t2 = $t0 - $t1

move $a0, $t2       #Tranfer result value stored in $t2 into $a0 to be printed.
li $v0,1            #Loads code 1 to print an integer.
syscall             #syscall executes and prints value from $a0
li $v0,10           #Loads code 10 to terminate program.
syscall             #Terminates the program.
```

$v0	$t0	$v0	$t1	$t2	$a0	$v0	$v0
5	6	5	5	1	1	1	a

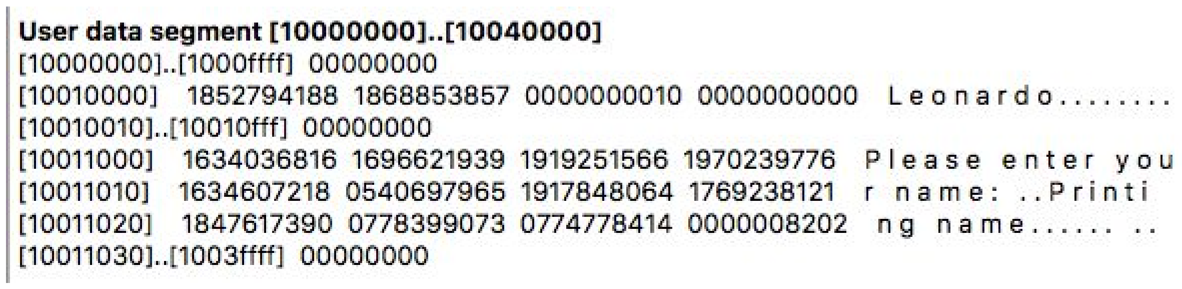
2- Discuss the importance and value of register $v0 in using syscall.
li $v0, 5 prompts user to input an integer value. After user enters a value, lets say 5, it is stored in $t0. Then the next command is to move the value stored in $v0 to $t0.Then register $t0 changes from 0 to 5. $v0 loads 4 and I really don't know why or what is going on with this command. Afterwards, $v0 loads the next code which is 5, to read an integer instruction. Then the program does system call and this prompts the user to enter an integer value which in this case is 22 and it is read into $v0. After user input $v0 is moved to $t0, $v0 is assign code 5 again to read second user value. Syscall prompts the user to enter second integer, 32, and then the value gets read into $v0. After user enters second value $v0 gets moved to $t1 and the next command will be subtract $t0 - $t1. Result gets stored into register $t2 and moved to $a0 to be printed. $v0 gets assigned this time code 1 to print an integer. Syscall executes and the result value in $a0 gets printed in the console window. Lastly $v0 gets assigned code 10 to exit the program. Syscall executes $v0 and program is finished.

3- Write a program in mips assembly that will prompt the user to enter his/her name and then print it out.



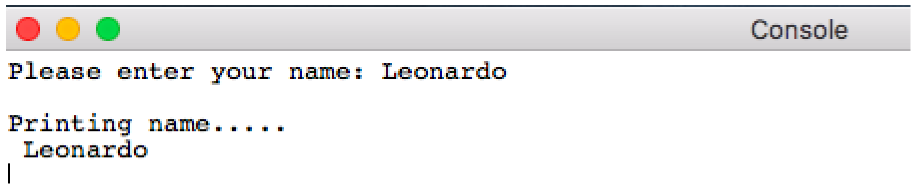
```
#Leonardo Roman
#Write a program that will prompt the user to enter his name
#and then print them out.

.data 0x10010000
name: .space 20

.data 0x10011000
message1: .asciiz "Please enter your name: "
message2: .asciiz "\nPrinting name.....\n "

.text
.globl main

main:
                        #printing message1 (Please enter your name:)
li $v0, 4               #for printing string
la $a0, message1        #loading address of the string that needs to be printed
syscall
                        #reading string input
li $v0, 8               #code 8 is to read a string
la $a0, name            #loading the address of empty space, where the name
                        #(string) is going to be saved in the data
li $a1, 20
move $t0, $a0
syscall

li $v0, 4               #printing string
la $a0, message2        #the string to be printed
syscall

move $a0, $t0           #moving the name to a0
li $v0, 4               #printing string (from a0)
syscall

li $v0, 10
syscall
```

Data segment



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]   1852794188  1868853857  0000000010  0000000000   L e o n a r d o . . . . . . . .
[10010010]..[10010fff]  00000000
[10011000]   1634036816  1696621939  1919251566  1970239776   P l e a s e   e n t e r   y o u
[10011010]   1634607218  0540697965  1917848064  1769238121   r   n a m e :   . . P r i n t i
[10011020]   1847617390  0778399073  0774778414  0000008202   n g   n a m e . . . . . .   . .
[10011030]..[1003ffff]  00000000
```

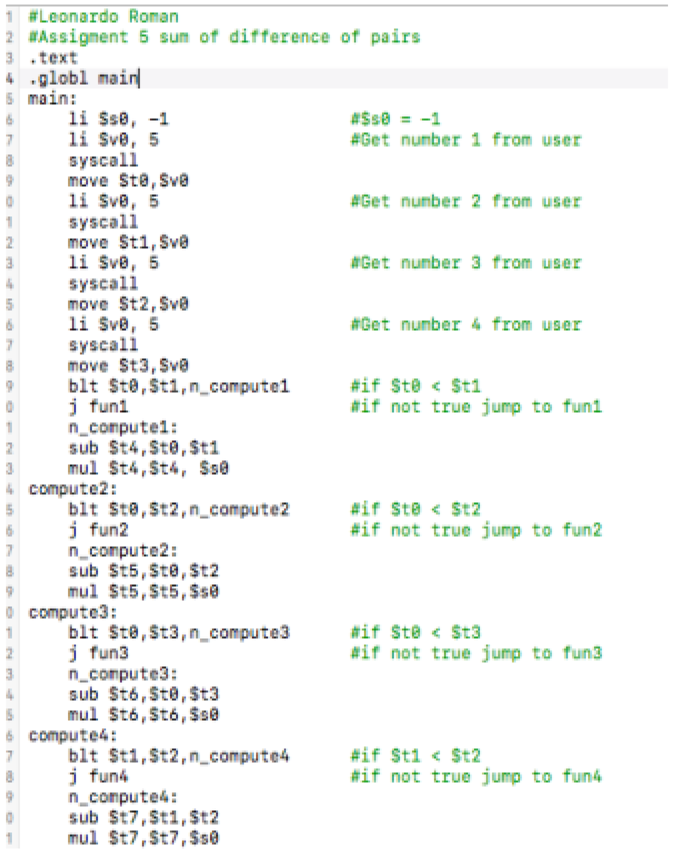
Output



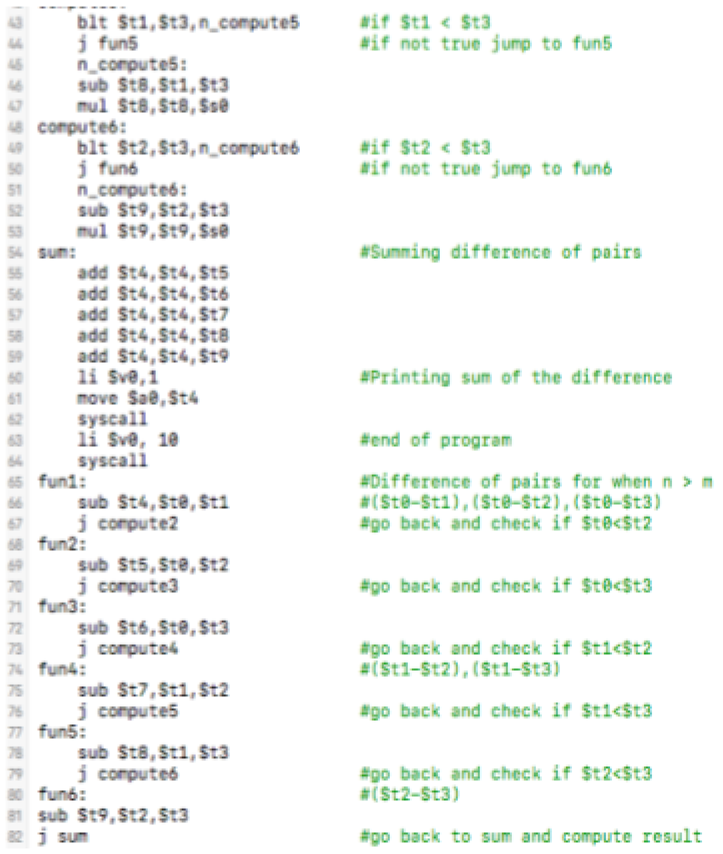
```
Please enter your name: Leonardo

Printing name.....
 Leonardo
```

Assignment 5

Write a program in Mips assembly that gets four integers from the user and calculates the sum of differences of all pairs created with these numbers.



```
#Leonardo Roman
#Assigment 5 sum of difference of pairs
.text
.globl main
main:
    li $s0, -1                  #$s0 = -1
    li $v0, 5                   #Get number 1 from user
    syscall
    move $t0,$v0
    li $v0, 5                   #Get number 2 from user
    syscall
    move $t1,$v0
    li $v0, 5                   #Get number 3 from user
    syscall
    move $t2,$v0
    li $v0, 5                   #Get number 4 from user
    syscall
    move $t3,$v0
    blt $t0,$t1,n_compute1      #if $t0 < $t1
    j fun1                      #if not true jump to fun1
    n_compute1:
    sub $t4,$t0,$t1
    mul $t4,$t4, $s0
compute2:
    blt $t0,$t2,n_compute2      #if $t0 < $t2
    j fun2                      #if not true jump to fun2
    n_compute2:
    sub $t5,$t0,$t2
    mul $t5,$t5,$s0
compute3:
    blt $t0,$t3,n_compute3      #if $t0 < $t3
    j fun3                      #if not true jump to fun3
    n_compute3:
    sub $t6,$t0,$t3
    mul $t6,$t6,$s0
compute4:
    blt $t1,$t2,n_compute4      #if $t1 < $t2
    j fun4                      #if not true jump to fun4
    n_compute4:
    sub $t7,$t1,$t2
    mul $t7,$t7,$s0
```



```
    blt $t1,$t3,n_compute5      #if $t1 < $t3
    j fun5                      #if not true jump to fun5
    n_compute5:
    sub $t8,$t1,$t3
    mul $t8,$t8,$s0
compute6:
    blt $t2,$t3,n_compute6      #if $t2 < $t3
    j fun6                      #if not true jump to fun6
    n_compute6:
    sub $t9,$t2,$t3
    mul $t9,$t9,$s0
sum:                            #Summing difference of pairs
    add $t4,$t4,$t5
    add $t4,$t4,$t6
    add $t4,$t4,$t7
    add $t4,$t4,$t8
    add $t4,$t4,$t9
    li $v0,1                    #Printing sum of the difference
    move $a0,$t4
    syscall
    li $v0, 10                  #end of program
    syscall
fun1:                           #Difference of pairs for when n > m
    sub $t4,$t0,$t1             #($t0-$t1),($t0-$t2),($t0-$t3)
    j compute2                  #go back and check if $t0<$t2
fun2:
    sub $t5,$t0,$t2
    j compute3                  #go back and check if $t0<$t3
fun3:
    sub $t6,$t0,$t3
    j compute4                  #go back and check if $t1<$t2
fun4:                           #($t1-$t2),($t1-$t3)
    sub $t7,$t1,$t2
    j compute5                  #go back and check if $t1<$t3
fun5:
    sub $t8,$t1,$t3
    j compute6                  #go back and check if $t2<$t3
fun6:                           #($t2-$t3)
sub $t9,$t2,$t3
j sum                           #go back to sum and compute result
```

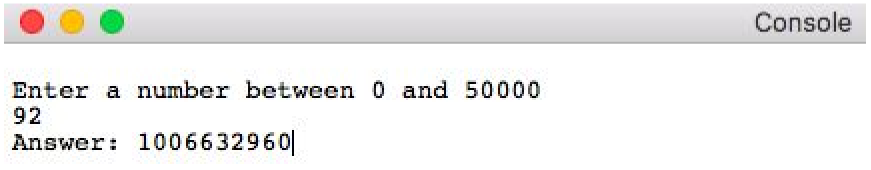
Assignment 6:

Explain what the main function of ex5.asm is. How is this function realized? What would happen if the user was allowed to give a much larger number? Why is that?

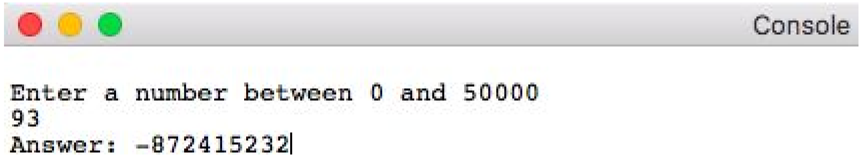
The main function of ex5.asm is to loop n-3 times and increment $t2 by multiplying the past value of $t2 times $t0 whenever $t0%3=0. The final value of $t2 will be displayed as soon as $t0 = 3 as $t0 decrements by 1 every iteration. I.e if the user enters 12

$result = 6[\,9(12 * 1)\,] = 648$

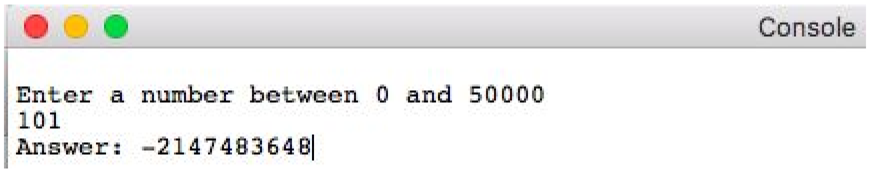
The answer to the next question can not be answered since this program fails after entering a number larger than 150. All integers greater than 101 will give a result of 0, hence 50000 is also out of this program scope. The reason why, is because the $t2 is rising exponentially and the register can not hold values greater than $2^{31}-1$. Therefore the numbers to be entered should be $0 \le n \le 101$ also $n \ge 92$ will give negative number as result.



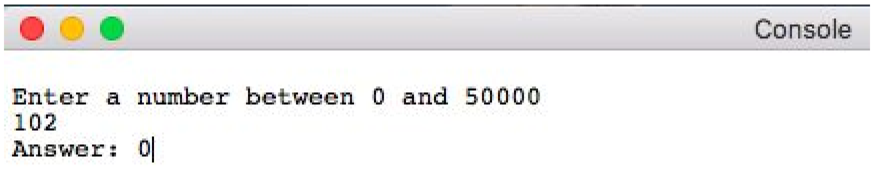
```
Enter a number between 0 and 50000
92
Answer: 1006632960
```



```
Enter a number between 0 and 50000
93
Answer: -872415232
```



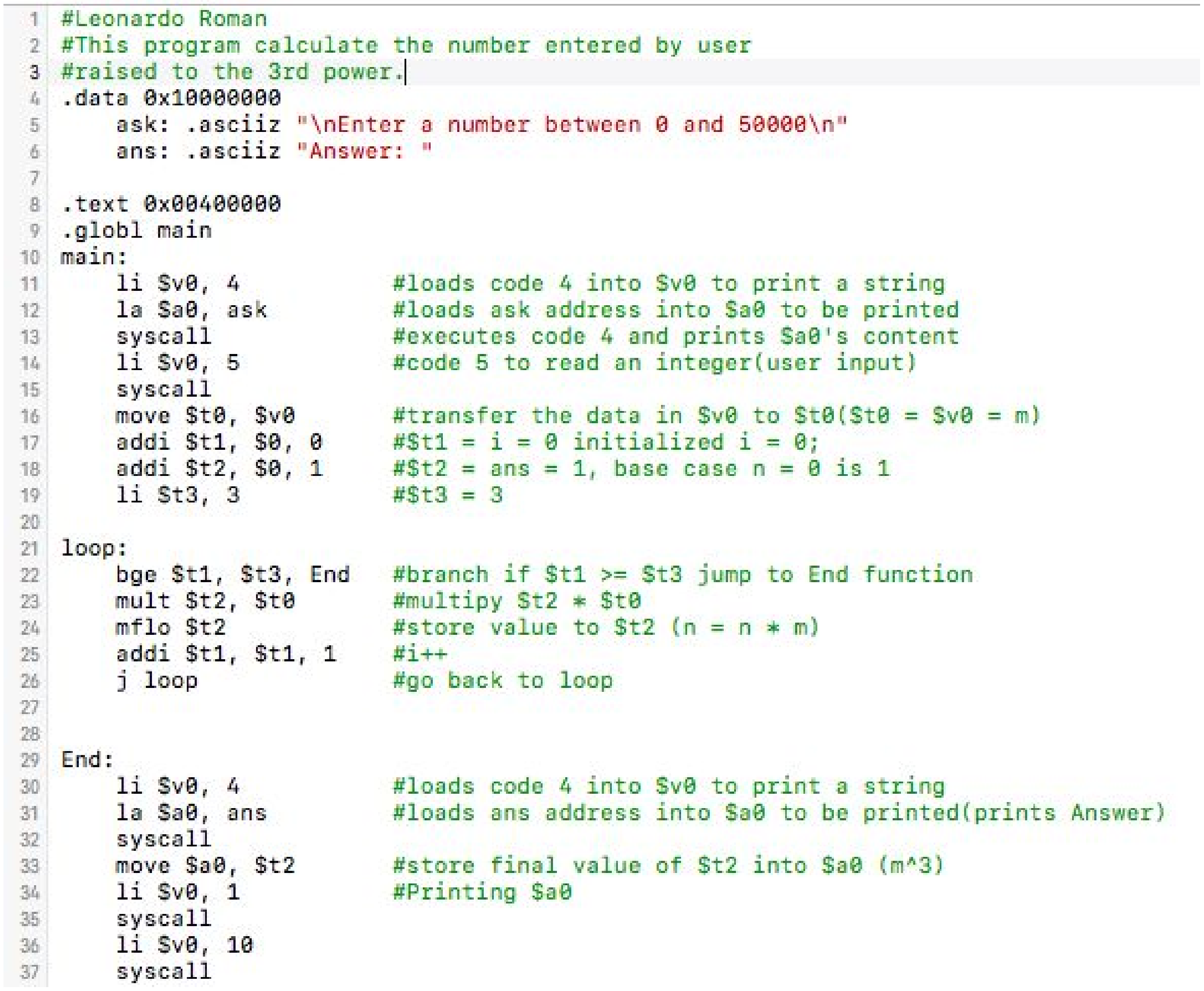
```
Enter a number between 0 and 50000
101
Answer: -2147483648
```



```
Enter a number between 0 and 50000
102
Answer: 0
```

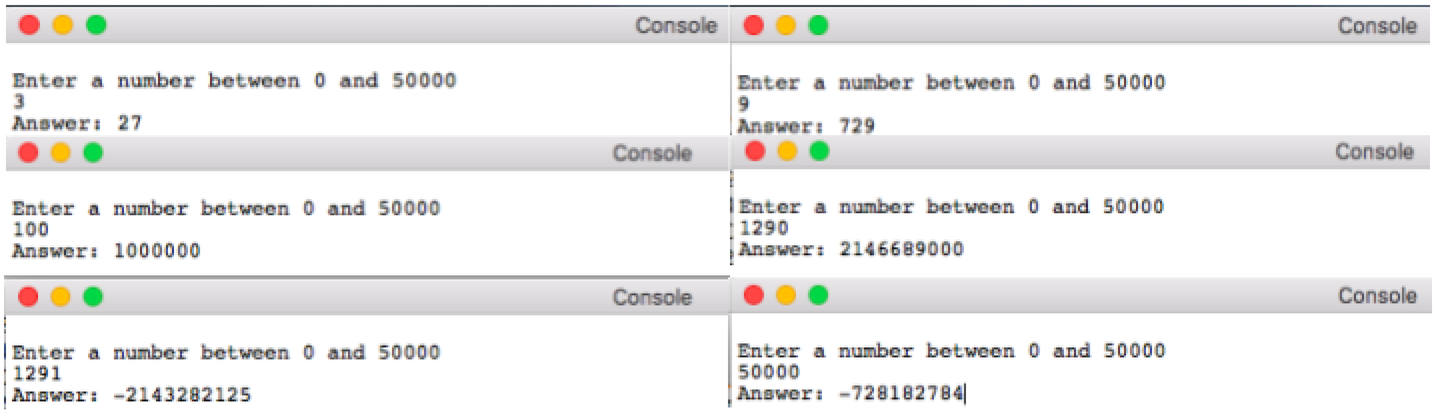
Assignment 7

Change the main function of Ex5.asmso that it provides the power of 3 that corresponds to the user’s input.



```
#Leonardo Roman
#This program calculate the number entered by user
#raised to the 3rd power.
.data 0x10000000
    ask: .asciiz "\nEnter a number between 0 and 50000\n"
    ans: .asciiz "Answer: "

.text 0x00400000
.globl main
main:
    li $v0, 4           #loads code 4 into $v0 to print a string
    la $a0, ask         #loads ask address into $a0 to be printed
    syscall             #executes code 4 and prints $a0's content
    li $v0, 5           #code 5 to read an integer(user input)
    syscall
    move $t0, $v0       #transfer the data in $v0 to $t0($t0 = $v0 = m)
    addi $t1, $0, 0     #$t1 = i = 0 initialized i = 0;
    addi $t2, $0, 1     #$t2 = ans = 1, base case n = 0 is 1
    li $t3, 3           #$t3 = 3

loop:
    bge $t1, $t3, End   #branch if $t1 >= $t3 jump to End function
    mult $t2, $t0       #multipy $t2 * $t0
    mflo $t2            #store value to $t2 (n = n * m)
    addi $t1, $t1, 1    #i++
    j loop              #go back to loop


End:
    li $v0, 4           #loads code 4 into $v0 to print a string
    la $a0, ans         #loads ans address into $a0 to be printed(prints Answer)
    syscall
    move $a0, $t2       #store final value of $t2 into $a0 (m^3)
    li $v0, 1           #Printing $a0
    syscall
    li $v0, 10
    syscall
```

The answer depends on how big the entered number is. Since the register can only hold $2^{31}-1$ numbers the max value before overflow will be 1290 as shown in the figure below. If the number entered is greater than 1290 the register will overflow one bit and the resulting number will be negative as shown in the figure below.



```
Enter a number between 0 and 50000
3
Answer: 27
```

```
Enter a number between 0 and 50000
9
Answer: 729
```

```
Enter a number between 0 and 50000
100
Answer: 1000000
```

```
Enter a number between 0 and 50000
1290
Answer: 2146689000
```

```
Enter a number between 0 and 50000
1291
Answer: -2143282125
```

```
Enter a number between 0 and 50000
50000
Answer: -728182784
```

Conclusion

We practiced some basic operations of assembly language by simple testing some already written programs given in the lab manual. All programs were rewritten in sublime text editor and then load it into QtSpim. Features in QtSpim such as f10(step by step test) were explored to see how every register behaved as it is fetch from memory. Some of the basic topics discussed in lecture were put in practice in this lab assignment. Visualization in how a computer executes assembly and machine language instructions were demonstrated in this lab to understand the use and behavior of different registers in a computer e.g. how to load or store words, to perform arithmetic operations, to read and print integers and strings, to use conditional instructions, and to use iterations.