Computer Architecture Lab

Experiment 3

Arithmetic Operations(Basic and float) and Combinatorial Logic.

Lab Instructor: Christos M.

Date Performed: 03/01/17

Date Submitted: 03/21/17

Name: Leonardo Roman lr534

Purpose:

The purpose of this lab is to learn how to do arithmetic operations and combinatorial logic. Also in this lab, we will learn about floating point operations.

Assignment 1:

In this assignment, we will create a program that will prompt the user to enter 3 numbers A, B and C. Then the program will perform a boolean operation such that

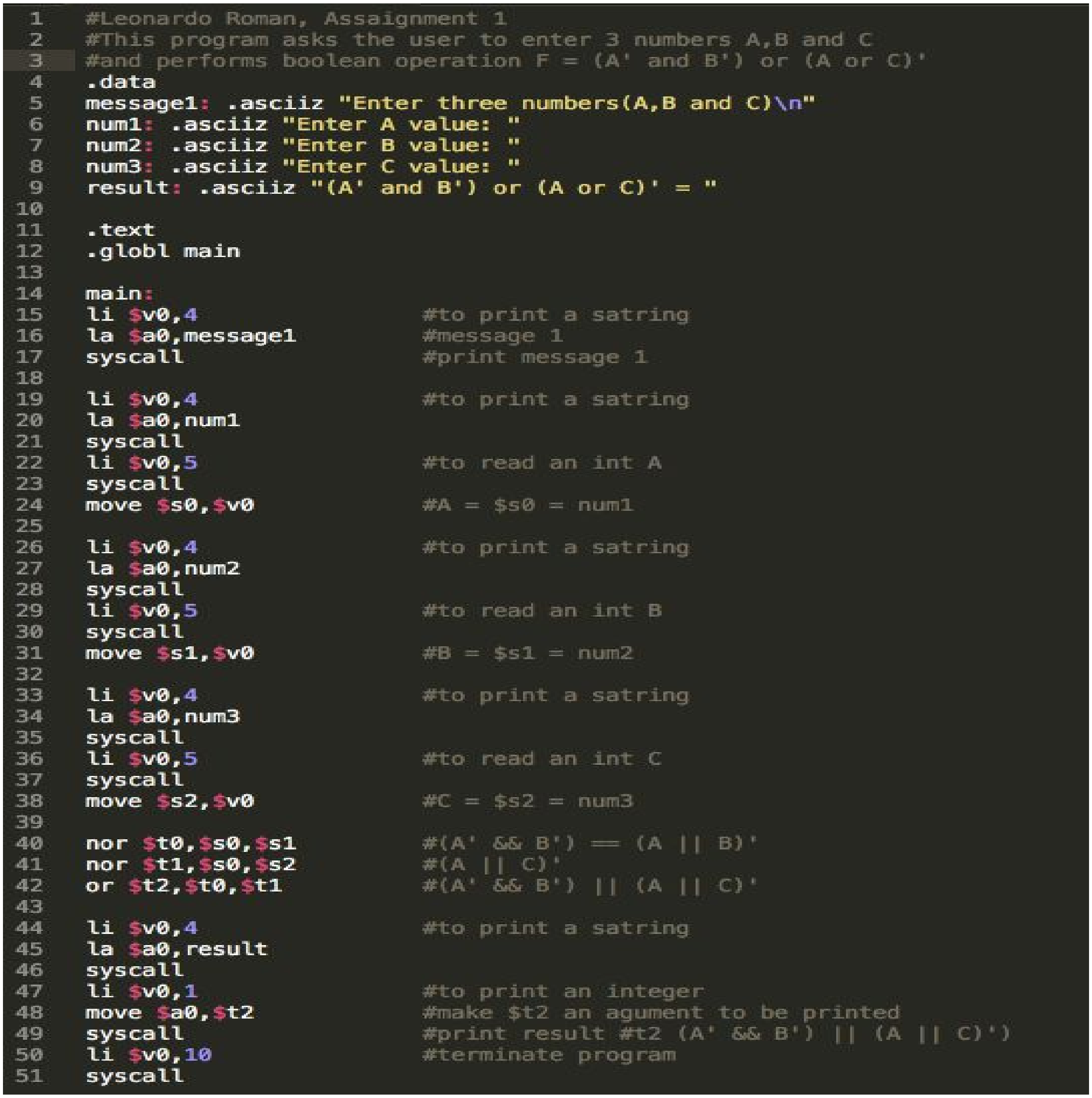
$$F = (A' \text{ and } B') \text{ or } (A \text{ or } C)'$$

Pseudocode:

```
Prompt user to enter 3 integers
A = number 1, B = number 2, C = number 3
norAB = !A and !B
norAC = !A and !C
F = norAB or norAC
print(F)
```

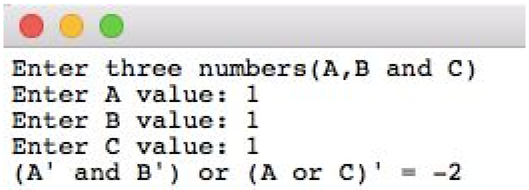
Conclusion:

The C pseudocode represents the visual operation that will be implemented in MIPS assembly language. By De Morgan’s law (A and C)’ is equivalent to A’ or C’ which can be implemented using “NOR” boolean logic. A,B and C are int variables while F contains the boolean product of the complements of such variables. Example if A = 1, B = 1 and C = 1 the complements of A,B and C would be -2,-2,-2 respectively and as result F would equal -2. The reason F = -2 is because the binary representation of 1 is 0001, in a 4-bit string, and its complement is 1110 which equals $(-2)^3 + 2^2 + 2^1 + 2^0 \;=\; -2$.



```
#Leonardo Roman, Assaignment 1
#This program asks the user to enter 3 numbers A,B and C
#and performs boolean operation F = (A' and B') or (A or C)'
.data
message1: .asciiz "Enter three numbers(A,B and C)\n"
num1: .asciiz "Enter A value: "
num2: .asciiz "Enter B value: "
num3: .asciiz "Enter C value: "
result: .asciiz "(A' and B') or (A or C)' = "

.text
.globl main

main:
li $v0,4                #to print a satring
la $a0,message1         #message 1
syscall                 #print message 1

li $v0,4                #to print a satring
la $a0,num1
syscall
li $v0,5                #to read an int A
syscall
move $s0,$v0            #A = $s0 = num1

li $v0,4                #to print a satring
la $a0,num2
syscall
li $v0,5                #to read an int B
syscall
move $s1,$v0            #B = $s1 = num2

li $v0,4                #to print a satring
la $a0,num3
syscall
li $v0,5                #to read an int C
syscall
move $s2,$v0            #C = $s2 = num3

nor $t0,$s0,$s1         #(A' && B') == (A || B)'
nor $t1,$s0,$s2         #(A || C)'
or $t2,$t0,$t1          #(A' && B') || (A || C)'

li $v0,4                #to print a satring
la $a0,result
syscall
li $v0,1                #to print an integer
move $a0,$t2            #make $t2 an agument to be printed
syscall                 #print result #t2 (A' && B') || (A || C)')
li $v0,10               #terminate program
syscall
```

Output:



```
Enter three numbers(A,B and C)
Enter A value: 1
Enter B value: 1
Enter C value: 1
(A' and B') or (A or C)' = -2
```

Assignment 2:

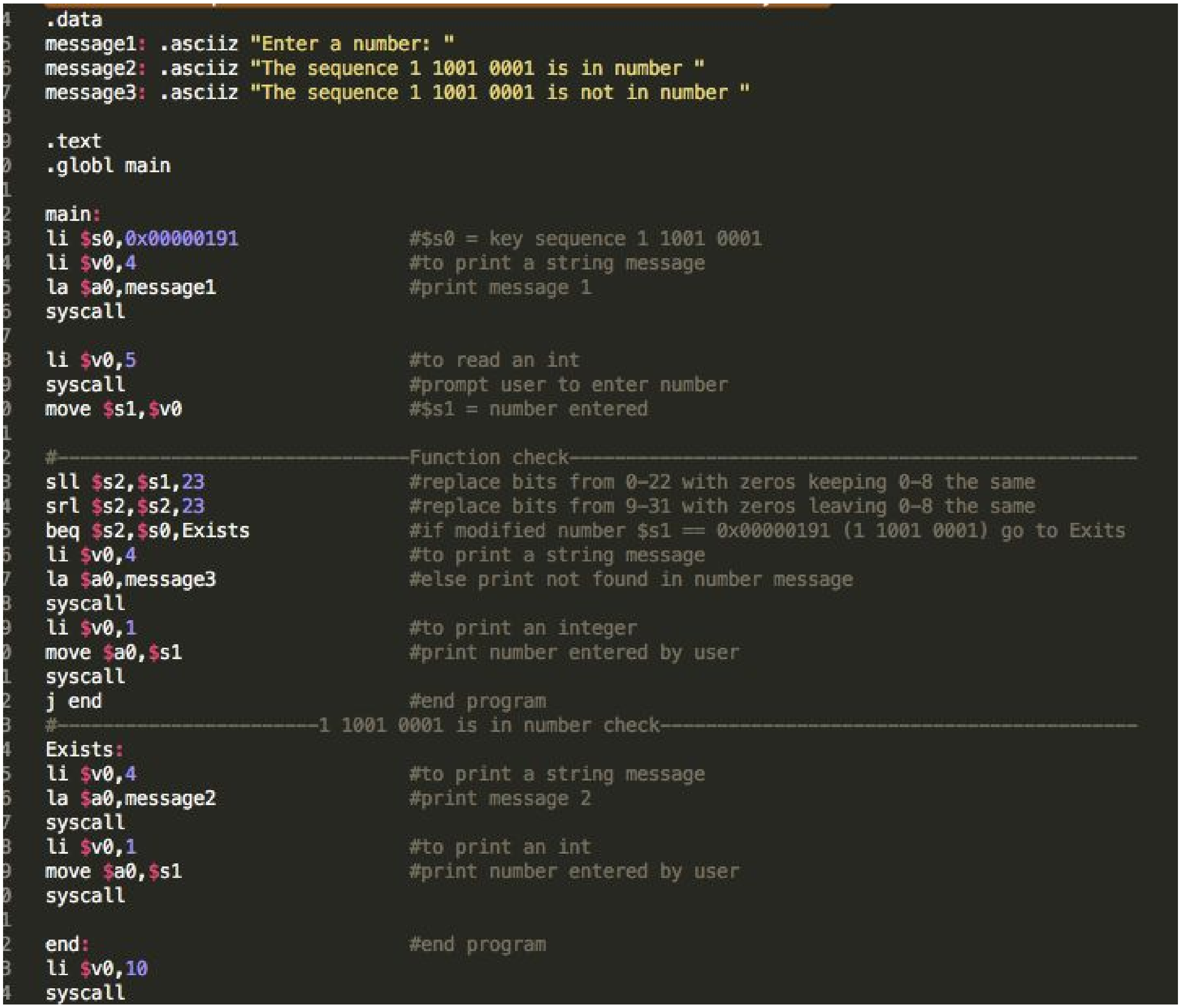
This program asks user to enter a number and checks if a sequence of bits 1 1001 0001 is in that number entered by user.

Pseudocode:

```
Let key[9] = 1 1 0 0 1 0 0 0 1
Prompt user for integer number
for(i=number;i>1;i/2) {size = i}
Function:
While ( number > 1 ) {
    remainder =number %2
    for(i = 0 ; i < size; i++){
        If ( remainder!=0) {arr[size] = 1}
        else arr[size] = 0
    }//end for loop
    number = number / 2;
} //end while loop
for(i = 0 ; i < 9; i++){
    if (key[i] == arr[i]){
        Print sequence is in number
    }else print sequence not in number
```

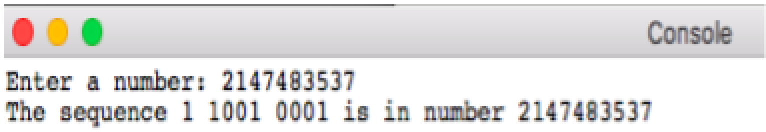
Conclusion:

This program compared a key value in hex, 0x00000191, with the bit sequence of a number entered by the user. This was accomplished first by getting rid off the 23 most righted bits using srl logic and then shifting the bits back to original place using sll 23. This way the 9 bits needed for comparison were kept and all other bits were replaced by zeros. After shifts were done the new value was compared to the one in key. If both values matched then the sequence 1 1001 0001 is in number.

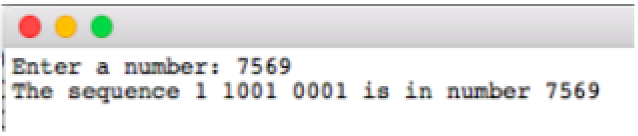


```
.data
message1: .asciiz "Enter a number: "
message2: .asciiz "The sequence 1 1001 0001 is in number "
message3: .asciiz "The sequence 1 1001 0001 is not in number "

.text
.globl main

main:
li $s0,0x00000191               #$s0 = key sequence 1 1001 0001
li $v0,4                        #to print a string message
la $a0,message1                 #print message 1
syscall

li $v0,5                        #to read an int
syscall                         #prompt user to enter number
move $s1,$v0                    #$s1 = number entered

#-------------------------------Function check----------------------------------------------
sll $s2,$s1,23                  #replace bits from 0-22 with zeros keeping 0-8 the same
srl $s2,$s2,23                  #replace bits from 9-31 with zeros leaving 0-8 the same
beq $s2,$s0,Exists              #if modified number $s1 == 0x00000191 (1 1001 0001) go to Exits
li $v0,4                        #to print a string message
la $a0,message3                 #else print not found in number message
syscall
li $v0,1                        #to print an integer
move $a0,$s1                    #print number entered by user
syscall
j end                           #end program
#-----------------------1 1001 0001 is in number check-----------------------------------------
Exists:
li $v0,4                        #to print a string message
la $a0,message2                 #print message 2
syscall
li $v0,1                        #to print an int
move $a0,$s1                    #print number entered by user
syscall

end:                            #end program
li $v0,10
syscall
```

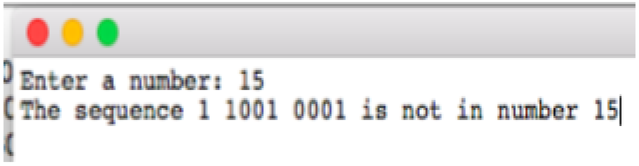
Output:



```
Enter a number: 2147483537
The sequence 1 1001 0001 is in number 2147483537
```



```
Enter a number: 7569
The sequence 1 1001 0001 is in number 7569
```



```
Enter a number: 15
The sequence 1 1001 0001 is not in number 15
```

Assignment 3:

This assignment is based on floating number conversion using IEEE 754 standard.

A.Calculate the number represented by

Sign	Exponent e(8 bits)	Fraction(23-bits)
1	11 00 11 10	00 10 11 00 11 11 10 00 10 11 00 0

Using the formula

$$number = (-1)^{sign}(1 + \sum_{i=1}^{23} b_{-i}2^{-i}) \times 2^{(e-127)}$$

$$(-1)^1 \times (1 + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-17} + 2^{-19} + 2^{-20}) \times 2^{(206-127)}$$

$$-1 \times 1.175669670 \times 2^{79}$$

$$=-7.106487097E23$$

B.Represent number -2.54_{10} by the binary string and fill the following table

$-1 = (-1)^1$

$exponent\, 2 = 2^1 = 2^{128-127} \rightarrow e = 128\ or\ 10\ 00\ 00\ 00$

$fraction\, 1.54 = 1 + 2^{-2} + 2^{-6} + 2^{-8} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-19} + 2^{-20} + 2^{-21}$

Sign	Exponent e(8 bits)	Fraction(23-bits)
1	10 00 00 00	01 00 01 01 00 01 11 10 10 11 10 0

Assignment 4:

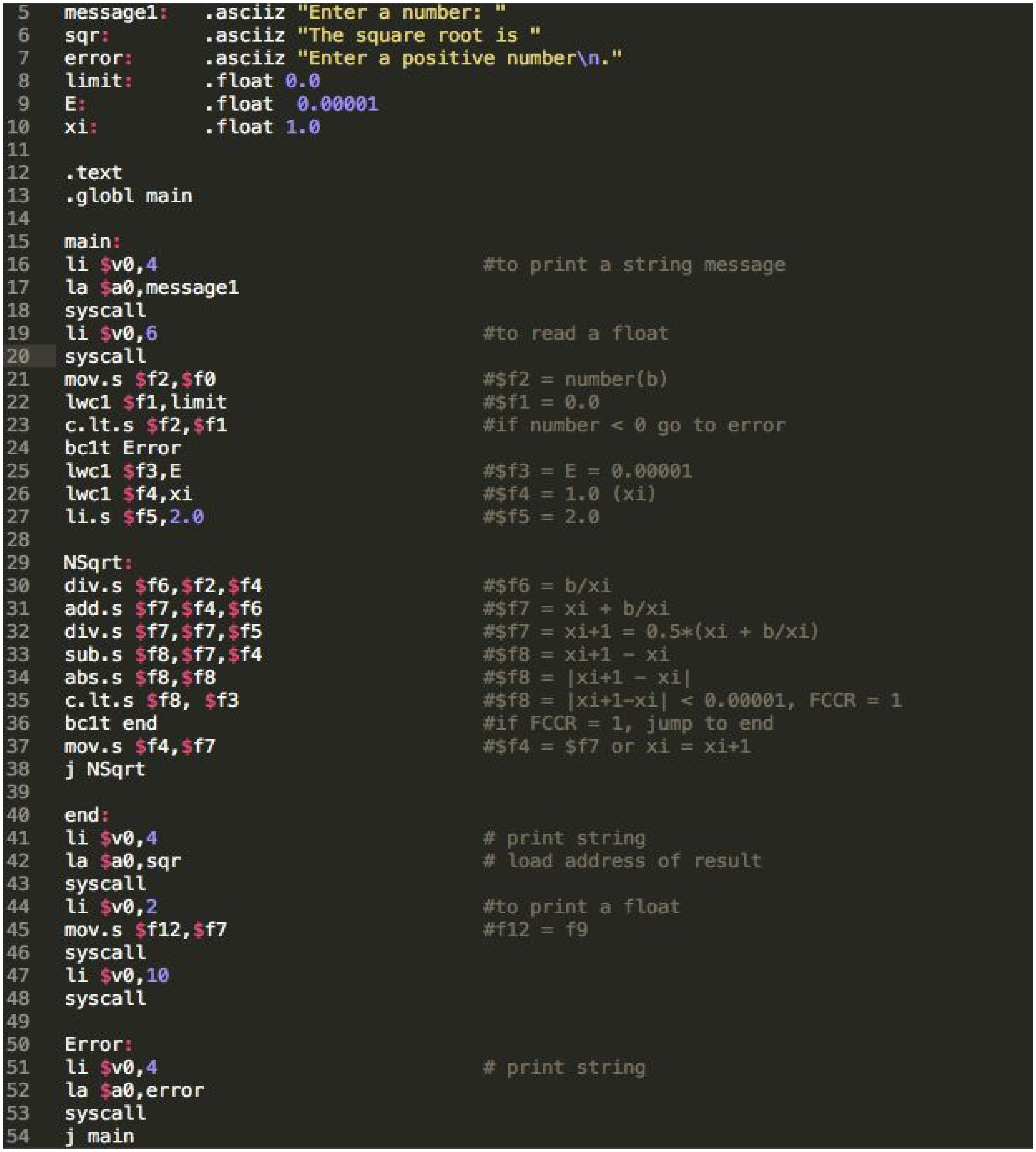
In this assignment, we will create a program to calculate the square root of a positive floating number using newton's method.

Pseudocode:

```
Prompt user for a positive float number
Input validation
Function:
While ( $|xi+1-xi|<0.00001$ ) {
    xi+1 = 0.5*(xi + b/xi)
}
Print the result
```

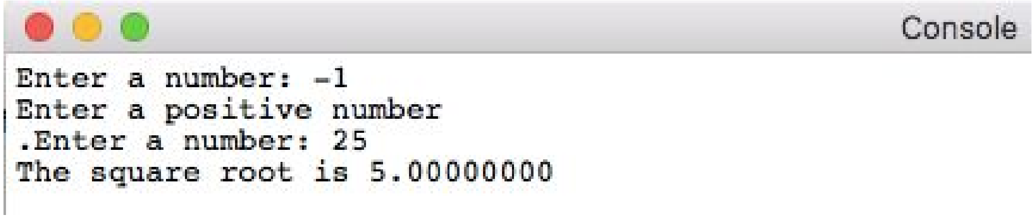
Conclusion:

The Newton’s law equation, xi+1 = 0.5*(xi + b/xi), is computed by breaking it into small branches and then by computing the small individual branches and combining them we obtain the square root of the number entered by user. Several instructions for floating point are used, along with the coprocessor registers to store the floating point numbers. The memory is used to allocate the floating point values for initial values 0.0, 1.0, and ε=0.00001.



```
message1:   .asciiz "Enter a number: "
sqr:        .asciiz "The square root is "
error:      .asciiz "Enter a positive number\n."
limit:      .float 0.0
E:          .float  0.00001
xi:         .float 1.0

.text
.globl main

main:
li $v0,4                            #to print a string message
la $a0,message1
syscall
li $v0,6                            #to read a float
syscall
mov.s $f2,$f0                       #$f2 = number(b)
lwc1 $f1,limit                      #$f1 = 0.0
c.lt.s $f2,$f1                      #if number < 0 go to error
bc1t Error
lwc1 $f3,E                          #$f3 = E = 0.00001
lwc1 $f4,xi                         #$f4 = 1.0 (xi)
li.s $f5,2.0                        #$f5 = 2.0

NSqrt:
div.s $f6,$f2,$f4                   #$f6 = b/xi
add.s $f7,$f4,$f6                   #$f7 = xi + b/xi
div.s $f7,$f7,$f5                   #$f7 = xi+1 = 0.5*(xi + b/xi)
sub.s $f8,$f7,$f4                   #$f8 = xi+1 - xi
abs.s $f8,$f8                       #$f8 = |xi+1 - xi|
c.lt.s $f8, $f3                     #$f8 = |xi+1-xi| < 0.00001, FCCR = 1
bc1t end                            #if FCCR = 1, jump to end
mov.s $f4,$f7                       #$f4 = $f7 or xi = xi+1
j NSqrt

end:
li $v0,4                            # print string
la $a0,sqr                          # load address of result
syscall
li $v0,2                            #to print a float
mov.s $f12,$f7                      #f12 = f9
syscall
li $v0,10
syscall

Error:
li $v0,4                            # print string
la $a0,error
syscall
j main
```

Output



```
Enter a number: -1
Enter a positive number
.Enter a number: 25
The square root is 5.00000000
```

Assignment 5

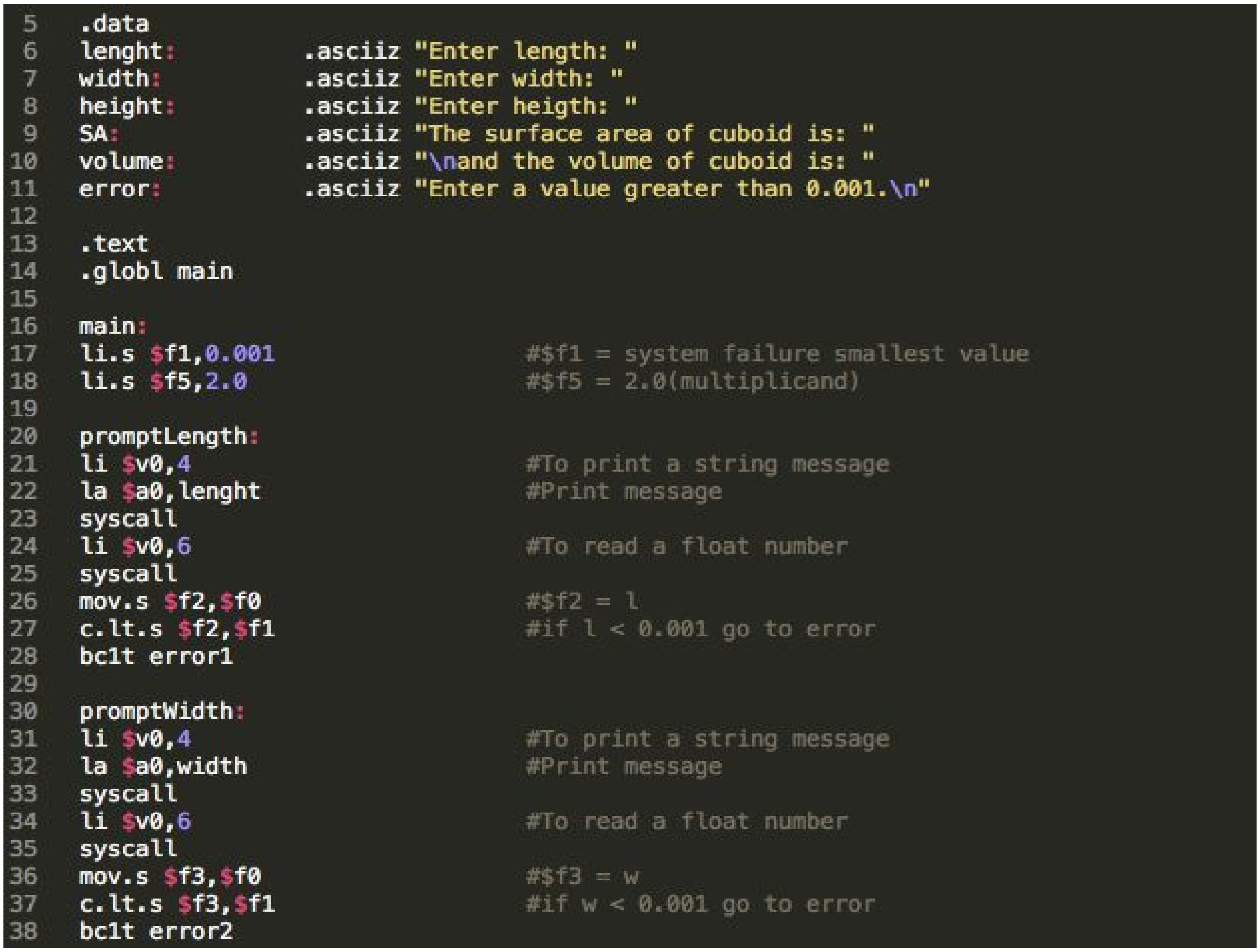
In this assignment, we will implement a program that calculates the surface area of a cuboid and its volume. By using the formula SA = 2(lw+lh+wh) and V = lwh, the program will prompt the user to enter the values of length, width and height.

Pseudocode:

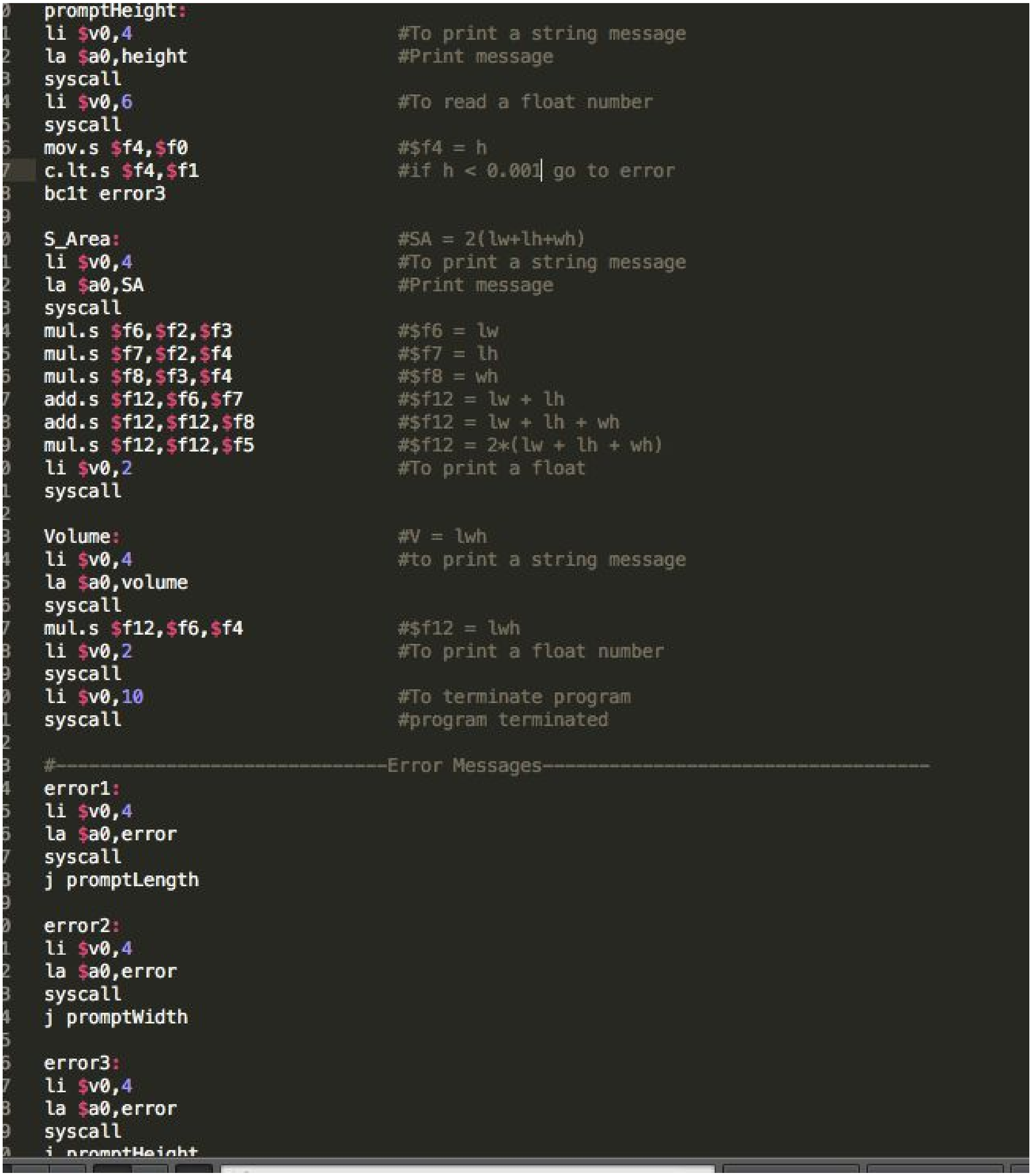
```
Prompt user to enter length, width and height
Input validation function(l,w,h)>0
$SA = 2(lw+lh+wh)$
$V = lwh$
print SA
print V
```

Conclusion:

Assignment calculations were a success, all values were correct and all input validations made sure of all values entered by user were correct. Floating point logic and registers were used in this assignment. All expected result were obtained.

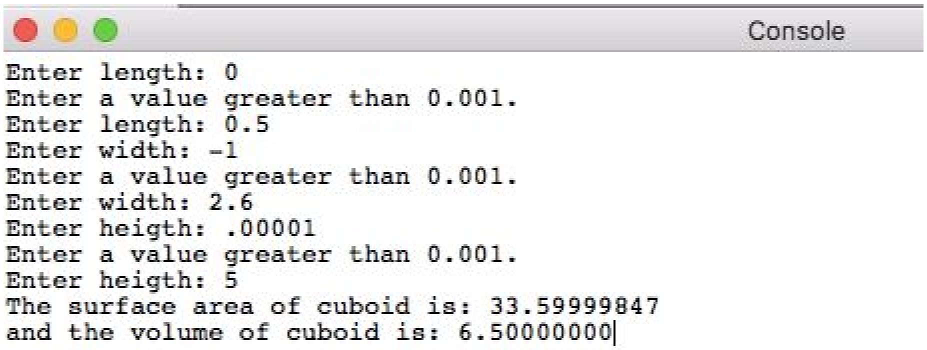


```
.data
lenght:         .asciiz "Enter length: "
width:          .asciiz "Enter width: "
height:         .asciiz "Enter heigth: "
SA:             .asciiz "The surface area of cuboid is: "
volume:         .asciiz "\nand the volume of cuboid is: "
error:          .asciiz "Enter a value greater than 0.001.\n"

.text
.globl main

main:
li.s $f1,0.001                  #$f1 = system failure smallest value
li.s $f5,2.0                    #$f5 = 2.0(multiplicand)

promptLength:
li $v0,4                        #To print a string message
la $a0,lenght                   #Print message
syscall
li $v0,6                        #To read a float number
syscall
mov.s $f2,$f0                   #$f2 = l
c.lt.s $f2,$f1                  #if l < 0.001 go to error
bc1t error1

promptWidth:
li $v0,4                        #To print a string message
la $a0,width                    #Print message
syscall
li $v0,6                        #To read a float number
syscall
mov.s $f3,$f0                   #$f3 = w
c.lt.s $f3,$f1                  #if w < 0.001 go to error
bc1t error2
```



```
promptHeight:
li $v0,4                        #To print a string message
la $a0,height                   #Print message
syscall
li $v0,6                        #To read a float number
syscall
mov.s $f4,$f0                   #$f4 = h
c.lt.s $f4,$f1                  #if h < 0.001 go to error
bc1t error3

S_Area:                         #SA = 2(lw+lh+wh)
li $v0,4                        #To print a string message
la $a0,SA                       #Print message
syscall
mul.s $f6,$f2,$f3               #$f6 = lw
mul.s $f7,$f2,$f4               #$f7 = lh
mul.s $f8,$f3,$f4               #$f8 = wh
add.s $f12,$f6,$f7              #$f12 = lw + lh
add.s $f12,$f12,$f8             #$f12 = lw + lh + wh
mul.s $f12,$f12,$f5             #$f12 = 2*(lw + lh + wh)
li $v0,2                        #To print a float
syscall

Volume:                         #V = lwh
li $v0,4                        #to print a string message
la $a0,volume
syscall
mul.s $f12,$f6,$f4              #$f12 = lwh
li $v0,2                        #To print a float number
syscall
li $v0,10                       #To terminate program
syscall                         #program terminated

#-------------------------------Error Messages-----------------------------------
error1:
li $v0,4
la $a0,error
syscall
j promptLength

error2:
li $v0,4
la $a0,error
syscall
j promptWidth

error3:
li $v0,4
la $a0,error
syscall
j promptHeight
```

Output:



```
Enter length: 0
Enter a value greater than 0.001.
Enter length: 0.5
Enter width: -1
Enter a value greater than 0.001.
Enter width: 2.6
Enter heigth: .00001
Enter a value greater than 0.001.
Enter heigth: 5
The surface area of cuboid is: 33.59999847
and the volume of cuboid is: 6.50000000
```

Assignment 6:

In this assignment, the mips program will calculate the sum of the square of all numbers less than or equal to the number entered by the user. If the number entered by user is greater than the numbers in the array then no sum will be performed.

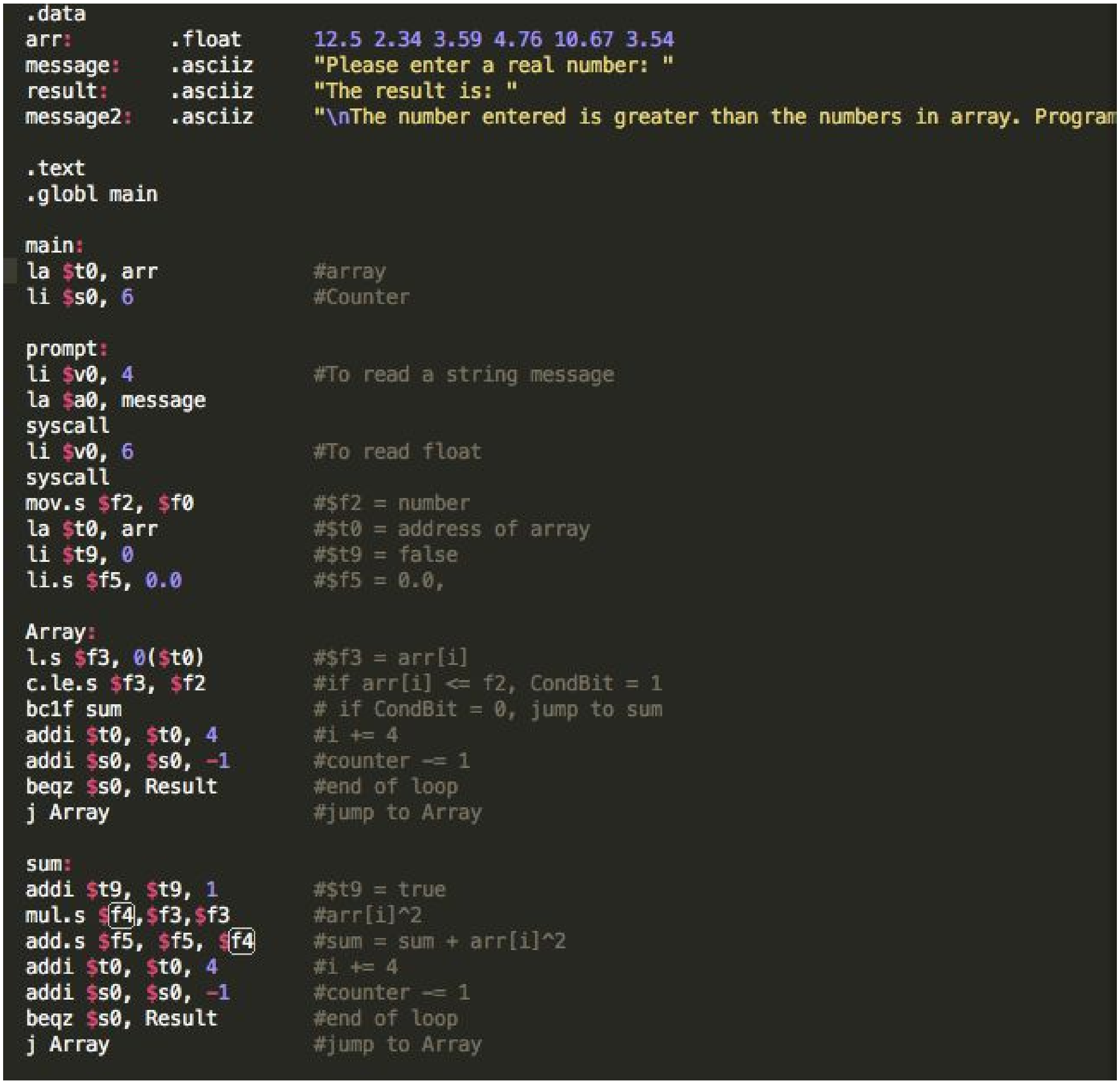
Pseudocode:

```
Prompt uses to enter a real float number
for(i = 0; i <= array size; i++){
    if(arr[i] < arr[i+1] ){
        minimum number = arr[i]
    }
    if(number <= minimum number){
        sum += a[i]^2;
    }
print sum;
```

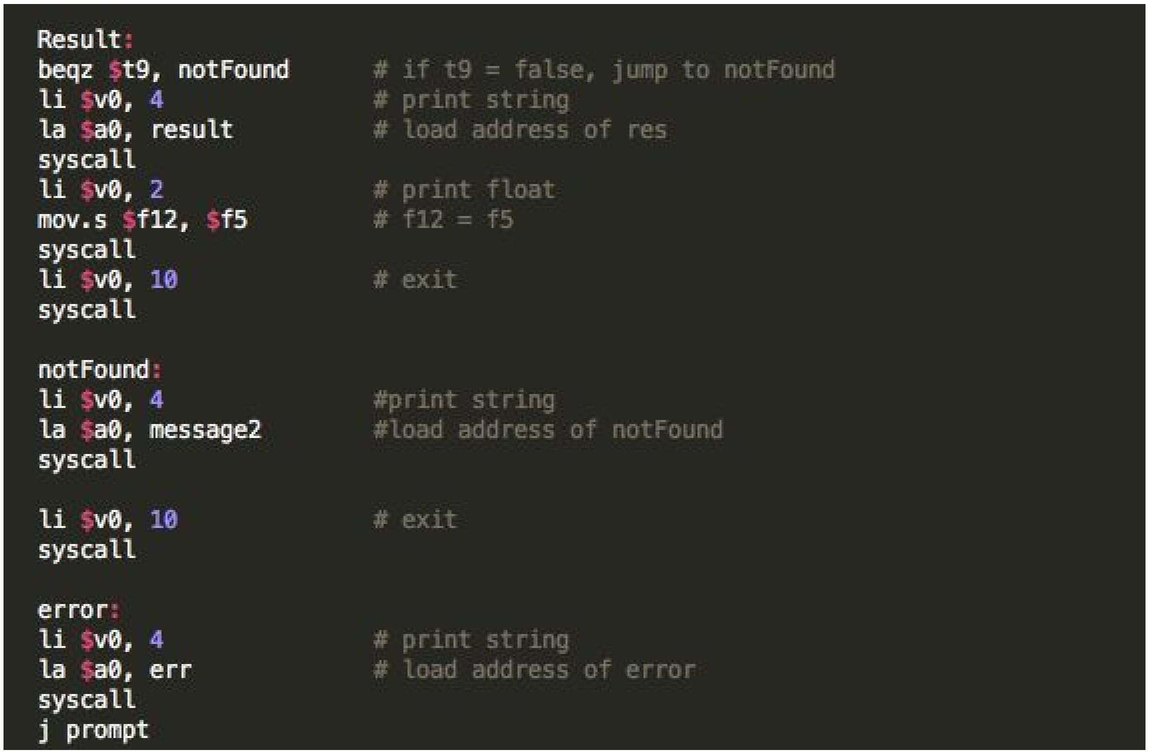
Conclusion:

Assignment calculations were a success, all values were correct and all input validations made sure of all values entered by user were correct. Floating point logic and registers were used in this assignment. All expected result were obtained.

Output

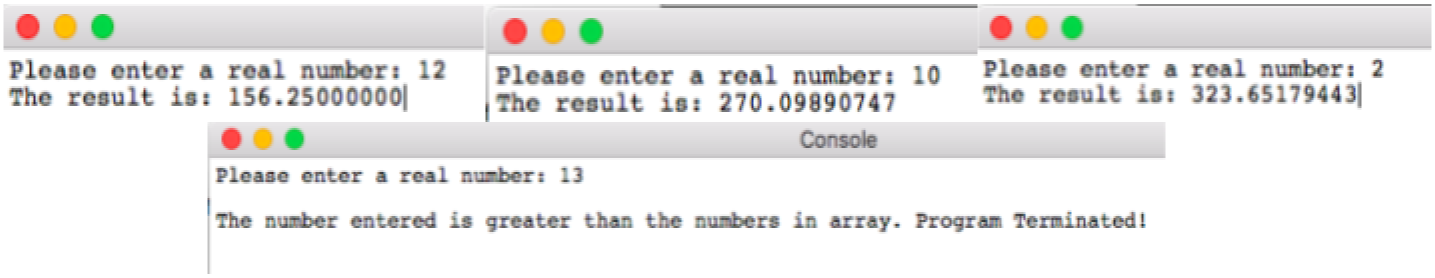


```
.data
arr:        .float      12.5 2.34 3.59 4.76 10.67 3.54
message:    .asciiz     "Please enter a real number: "
result:     .asciiz     "The result is: "
message2:   .asciiz     "\nThe number entered is greater than the numbers in array. Program

.text
.globl main

main:
la $t0, arr             #array
li $s0, 6               #Counter

prompt:
li $v0, 4               #To read a string message
la $a0, message
syscall
li $v0, 6               #To read float
syscall
mov.s $f2, $f0          #$f2 = number
la $t0, arr             #$t0 = address of array
li $t9, 0               #$t9 = false
li.s $f5, 0.0           #$f5 = 0.0,

Array:
l.s $f3, 0($t0)         #$f3 = arr[i]
c.le.s $f3, $f2         #if arr[i] <= f2, CondBit = 1
bc1f sum                # if CondBit = 0, jump to sum
addi $t0, $t0, 4        #i += 4
addi $s0, $s0, -1       #counter -= 1
beqz $s0, Result        #end of loop
j Array                 #jump to Array

sum:
addi $t9, $t9, 1        #$t9 = true
mul.s $f4,$f3,$f3       #arr[i]^2
add.s $f5, $f5, $f4     #sum = sum + arr[i]^2
addi $t0, $t0, 4        #i += 4
addi $s0, $s0, -1       #counter -= 1
beqz $s0, Result        #end of loop
j Array                 #jump to Array
```



```
Result:
beqz $t9, notFound       # if t9 = false, jump to notFound
li $v0, 4                # print string
la $a0, result           # load address of res
syscall
li $v0, 2                # print float
mov.s $f12, $f5          # f12 = f5
syscall
li $v0, 10               # exit
syscall

notFound:
li $v0, 4                #print string
la $a0, message2         #load address of notFound
syscall

li $v0, 10               # exit
syscall

error:
li $v0, 4                # print string
la $a0, err              # load address of error
syscall
j prompt
```

Output:



```
Please enter a real number: 12
The result is: 156.25000000
```

```
Please enter a real number: 10
The result is: 270.09890747
```

```
Please enter a real number: 2
The result is: 323.65179443
```

```
Please enter a real number: 13

The number entered is greater than the numbers in array. Program Terminated!
```