Peter Luo PL399 157004870

Report. Pseudocode and screen print of results included

Assignment 1.

Ask for int

Read int $v0 =5.

X3

Stores A, B, C in $s0, $s1, $s2

$t1 = A OR C

$t1 = NOT $t1

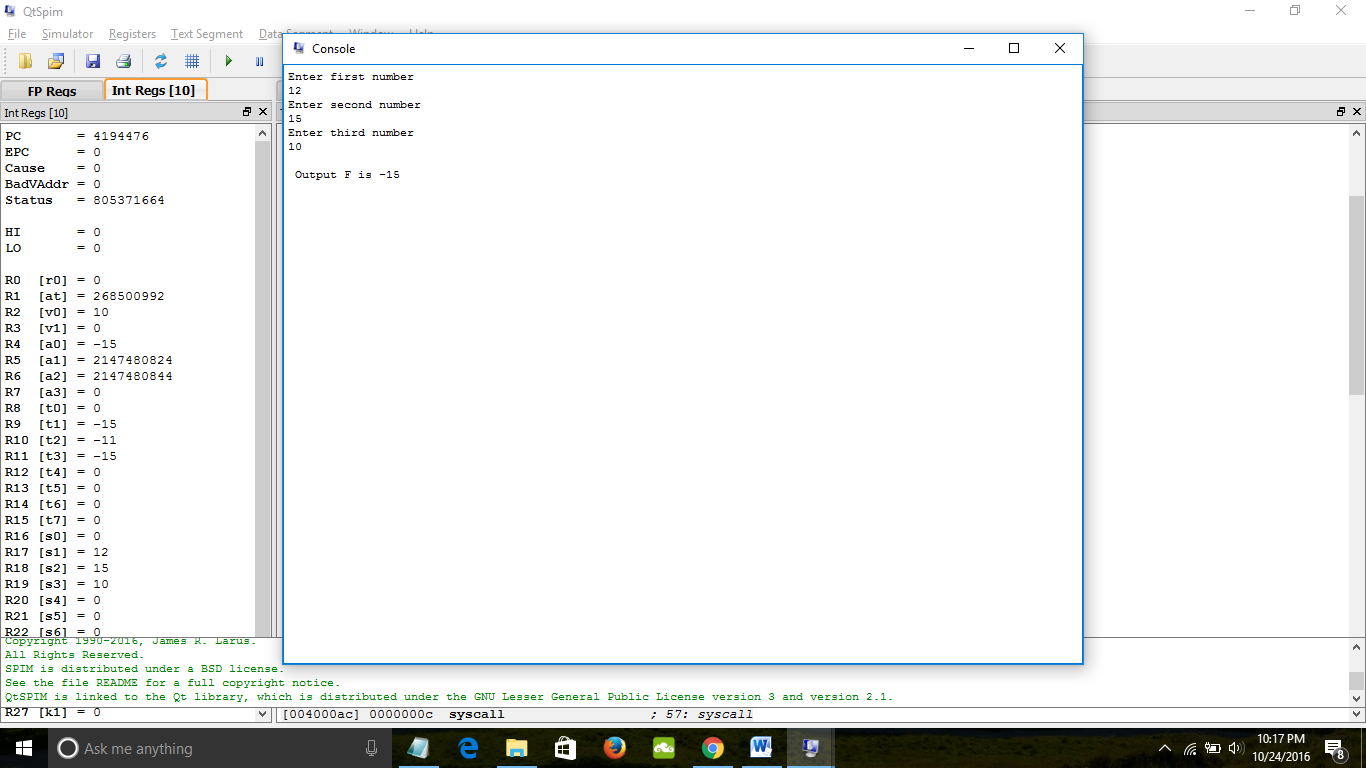
$t2 = B AND C

$t2 = NOT $t2

$t3 = $t2 and $t1

Print Message

Return $t3



Assignment 2.

Asks for two integers into $s0, $s1. $s0 = multiplicand $s1 = multiplier. $s2 = product

while $s1 > 0{ #link

if multiplier0 = 1{ (by using AND with 1)

product = product + multiplicand

multiplier shift right

}

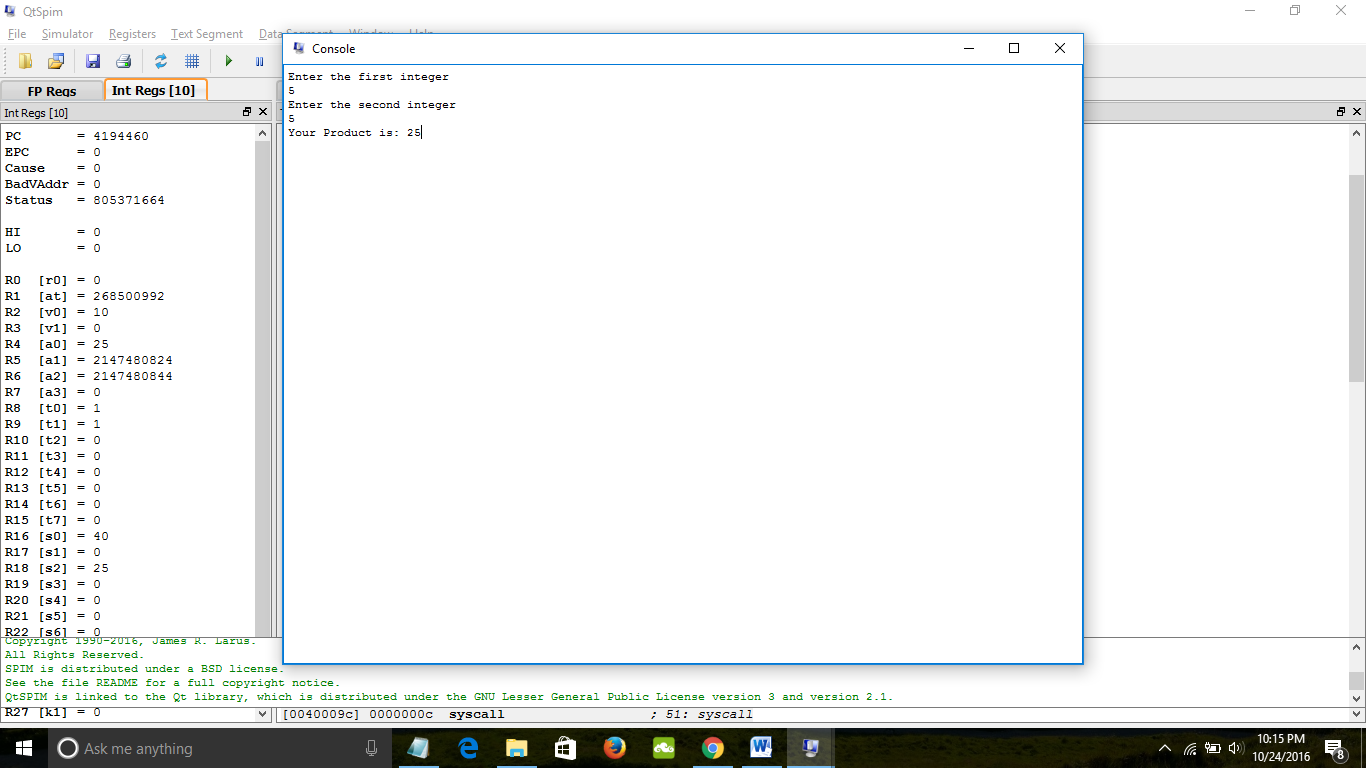
Else{

Multiplier shift right

}

Multiplicand shift left

} #goes back to while if $s1>0



Assignment 3:

Since the sign bit is 1, then the number is negative. The exponent 8 bit is equal to 30. Therefore the exponent of 2 is -97. The fraction is the sum of all the 2 to the negative exponent similar to the way positive is found.

Fraction = 2^{-3,-4,-6,-8,-9,-10,-11,-12,-15,-16,-19,-20} = 0.21074199676513672

Therefore the number is about -1.21074199676513672 \*(2^-97)

B.

Since the number is a positive, the sign bit is 0. The number 7.56 in binary is

111.1000 1111 0101 1100 0010 1000 1111 0101 1100 001010001111010111

To shift it over to start with one point something, we divide by 2^2. Therefore, we need to multiply that number by 2^2. Thus the exponent bits must be be 127+2, since it will have 127 subtracted from it.

Exponent bits must therefore be 129. In binary it is 10000001.

The fraction bits section only takes the first 23.

23-Fractional bits 1110 0011 1101 0111 0000 101

|  |  |  |
| --- | --- | --- |
| Sign(1) | Exponent (8) | Fraction(23) |
| 0 | 10000001 | 1110 0011 1101 0111 0000 101 |

Assignment 4.

# $f1 holds x­­i

# $f2 holds xi+1

# $f9 holds N

# $f6 holds 2.0

# $f5 holds 2\*xi

# $f7 holds N/Xi^2

# $f8 holds xi+1

Set $f3 as the small number 0.00001 with li.s $f3, 0.00001

Set li.s $f6 as 2.0

Set li.s $f1 as 1 #the initial guess

Asks for the positive number N

Move from $f0 into $f9

Label for while loop:

Mul.s $f5, $f1, $f6 # 2\*xi

Set $f7 as zero

Mul.s $f1 with $f1 and store in $f7

Add $f7 and $f5 and store into $f8

Divide $f7 by 3

Move $f7 into $$f2

Check to see if less than $s0

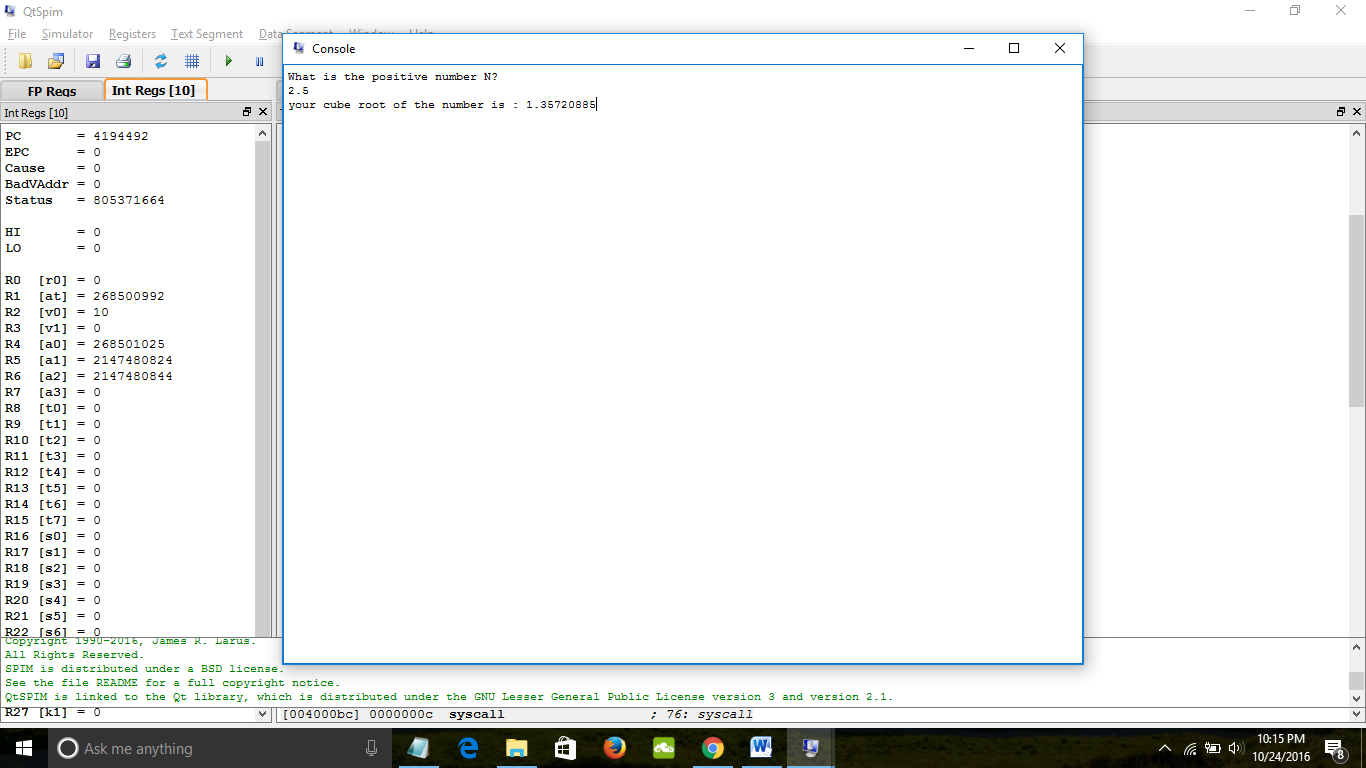
Subtract $f2 by $f1 into $f4

Abs.s of $f4

If $f4 less than $f3, then jump to endProgram

Otherwise jump to label for while loop

endProgram:



Assignment 5.

# $f1 = height

# $f2 = Radius

# $f3 = 0.333333333333333333

# $f4 = PI

# $f5 stores volume

li.s $f3 = 0.333333333333333

li.s $f4 = 3.14159

Asks for height in float point

Move $f0 into $f1

Asks for Radius in float point

Move $f0 into $f2

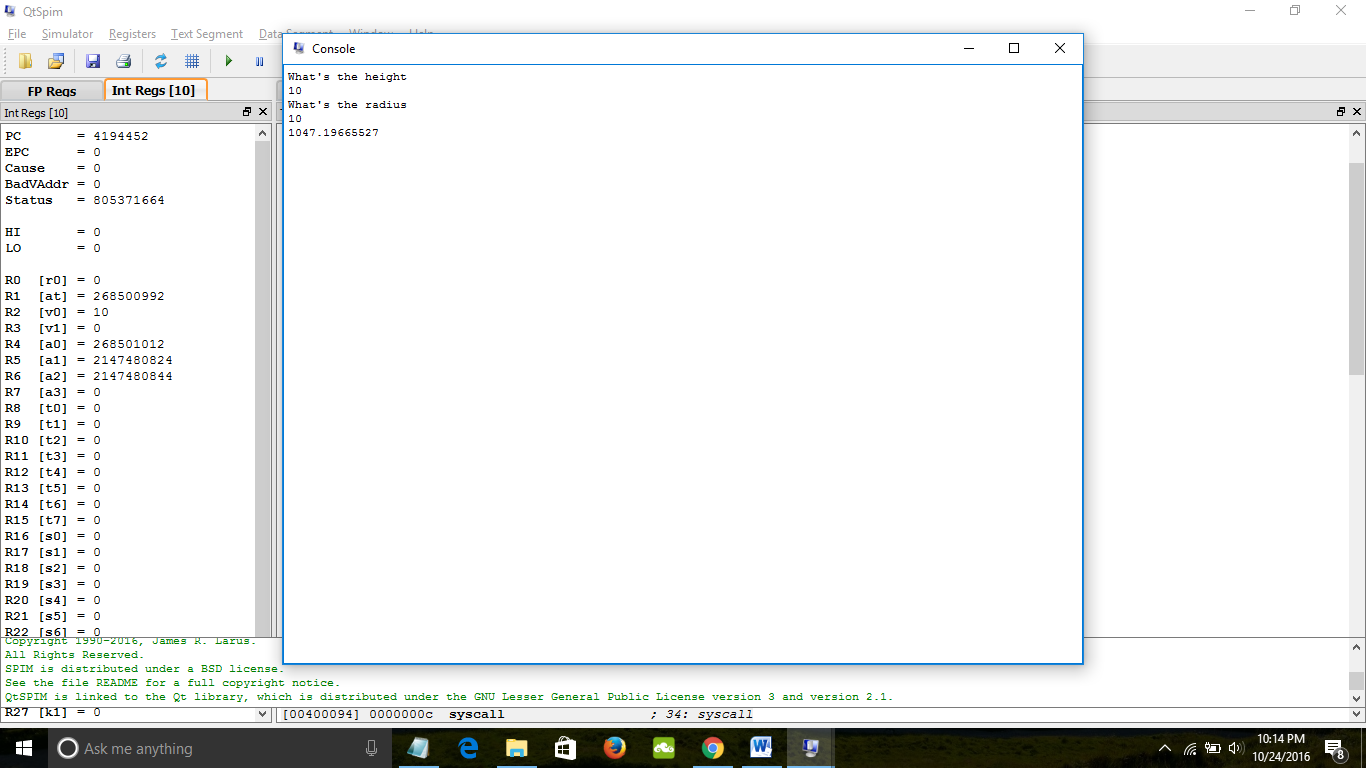
Mul.s $f2 by $f2 and store into $f2.

Mul.s $f3 and $f4 and store into $f5

Mul.s $f5 and $f2 and stores into $f5

Mul.s $f5 and $f1 and stores into $f5

Outputs $f5 as the volume



Assignment 6.

#$f1 holds user input

# $f2 holds sum

Asks for float

Move $f0 into $f1

Load address of array into $s0

Load 4 into $s1 #4 bits each single float point

loop:

store $counter \* $s4 into $t1

load float point from ($t1)(array) into $f5

check user input less than $f5

branch to skip if less than

branch to skip if equal to

add $f5 to $f2 and store in $f2

skip:

counter ++

counter less than 10

.data

Array: .float 1.35, 2.67, 3.566, 4.56, 5.98, 9.43, 12.34, 15.54, 23.87, 34.33

