Lab Tasks

1. Convert by hand the number **-123456789** into its 32-bit single-precision binary representation, and Show your work for a full mark.

123 456 789 ÷ 2 = 61 728 394 + 1;
61 728 394 ÷ 2 = 30 864 197 + 0;
30 864 197 ÷ 2 = 15 432 098 + 1;
15 432 098 ÷ 2 = 7 716 049 + 0;
7 716 049 ÷ 2 = 3 858 024 + 1;
3 858 024 ÷ 2 = 1 929 012 + 0;
1 929 012 ÷ 2 = 964 506 + 0;
964 506 ÷ 2 = 482 253 + 0;
482 253 ÷ 2 = 241 126 + 1;
241 126 ÷ 2 = 120 563 + 0;
120 563 ÷ 2 = 60 281 + 1;
60 281 ÷ 2 = 30 140 + 1;
30 140 ÷ 2 = 15 070 + 0;
15 070 ÷ 2 = 7 535 + 0;
7 535 ÷ 2 = 3 767 + 1;
3 767 ÷ 2 = 1 883 + 1;
1 883 ÷ 2 = 941 + 1;
941 ÷ 2 = 470 + 1;
470 ÷ 2 = 235 + 0;
235 ÷ 2 = 117 + 1;
117 ÷ 2 = 58 + 1;
58 ÷ 2 = 29 + 0;
29 ÷ 2 = 14 + 1;
14 ÷ 2 = 7 + 0;
7 ÷ 2 = 3 + 1;
3 ÷ 2 = 1 + 1;
1 ÷ 2 = 0 + 1;
123 456 789 (decimal) = 111 0101 1011 1100 1101 0001 0101 (binary)

2. Convert by hand the floating-point number **1 10010100 10011000001100000000000** (shown in binary) into its corresponding decimal value. Show your work for a full mark.

First. we subtract 127 from the exponent to get the true exponent: 148 – 127 = 21

Second. we convert the mantissa to decimal, by multiplying by 2N of number:

1 × 2-1 + 0 × 2-2 + 0 × 2-3 + 1 × 2-4 + 1 × 2-5 + 0 × 2-6 + 0 × 2-7 + 0 × 2-8 + 0 × 2-9 + 0 × 2-10 + 1 ×

2

-11 + 1 × 2-12 + 0 × 2-13 + 0 × 2-14 + 0 × 2-15 + 0 × 2-16 + 0 × 2-17 + 0 × 2-18 + 0 × 2-19 + 0 × 2-20 + 0

× 2-21 + 0 × 2-22 + 0 × 2-23 = 0.594482421875 (decimal)

Then we will add 1 to the mantissa: 0.594482421875 (decimal) + 1 = 1.594482421875

(decimal)

Finally, the floating-point value is then equal to −1 × 2

21 × 1.594482421875 = −3343872

3. Trace the following program by hand to determine the values of registers **$f0** thru **$f9**. Notice that **array1** and **array2** have the same elements, but in a different order. Comment on the sums of **array1** and **array2** elements computed in registers **$f4** and **$f9**, respectively. Now use the QTSPIM tool to trace the execution of the program and verify your results. What conclusion can be made from this exercise?

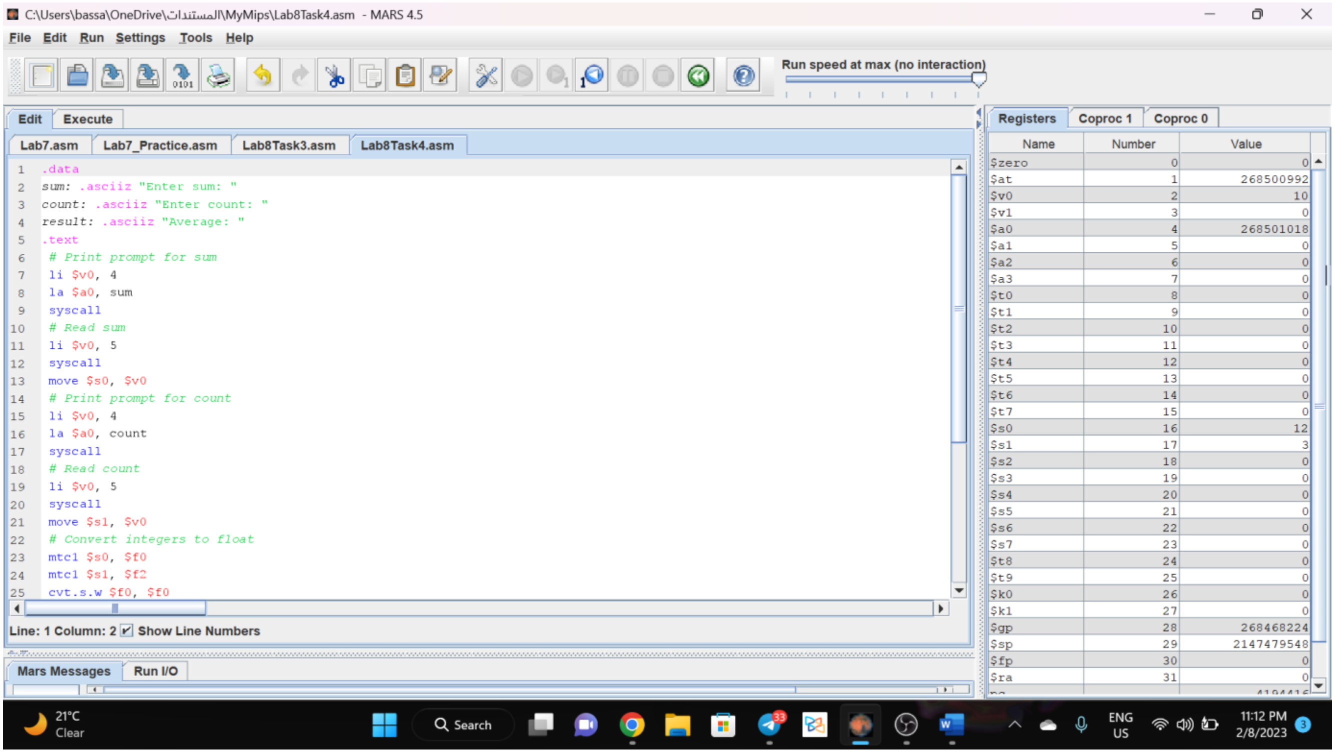
```
.data
array1:  .float  5.6e+20,  -5.6e+20,  1.2
array2:  .float  1.2,  5.6e+20,  -5.6e+20
.text
.globl main

main:

la      $t0,  array1
lwc1    $f0,  0($t0)
lwc1    $f1,  4($t0)
lwc1    $f2,  8($t0)
add.s   $f3,  $f0,  $f1
add.s   $f4,  $f2,  $f3
la      $t1,  array2
lwc1    $f5,  0($t1)
lwc1    $f6,  4($t1)
lwc1    $f7,  8($t1)
```

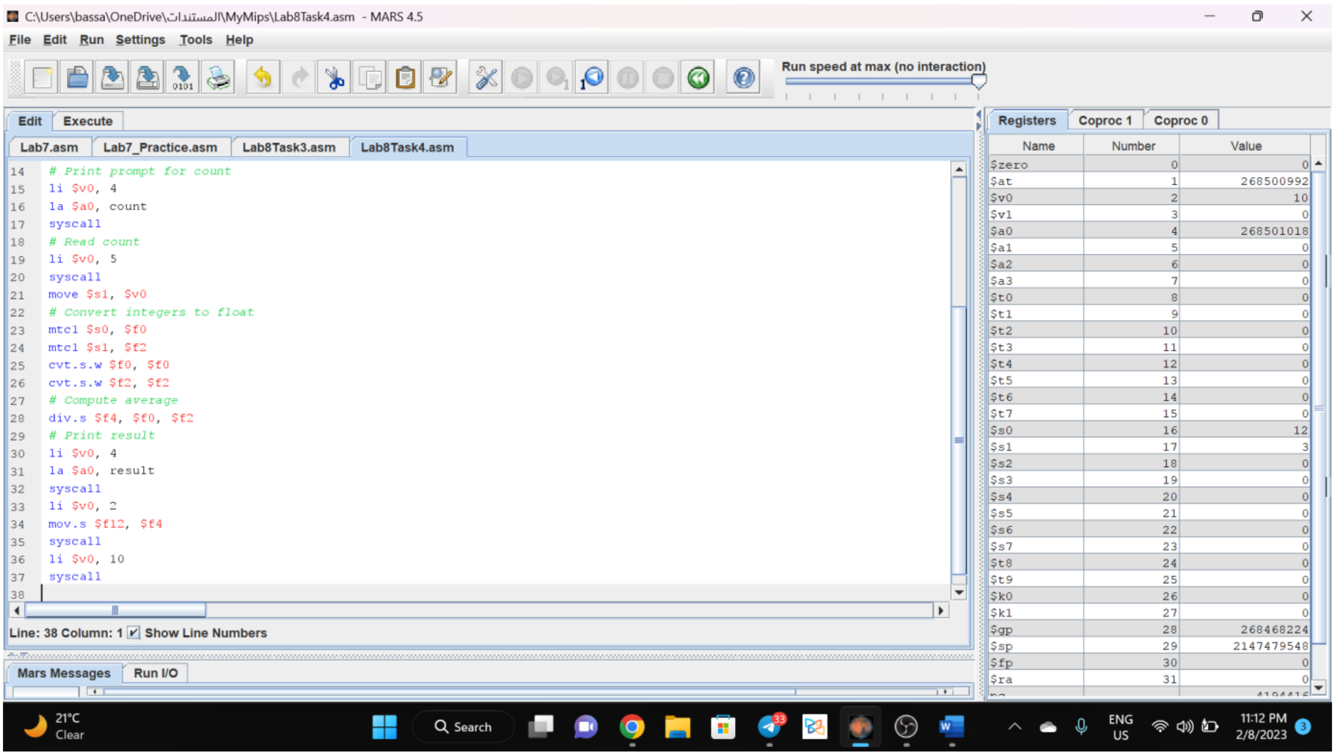
```
add.s   $f8,  $f5,  $f6
add.s   $f9,  $f7,  $f8

li $v0, 10          # To terminate the program
syscall

.end main
```

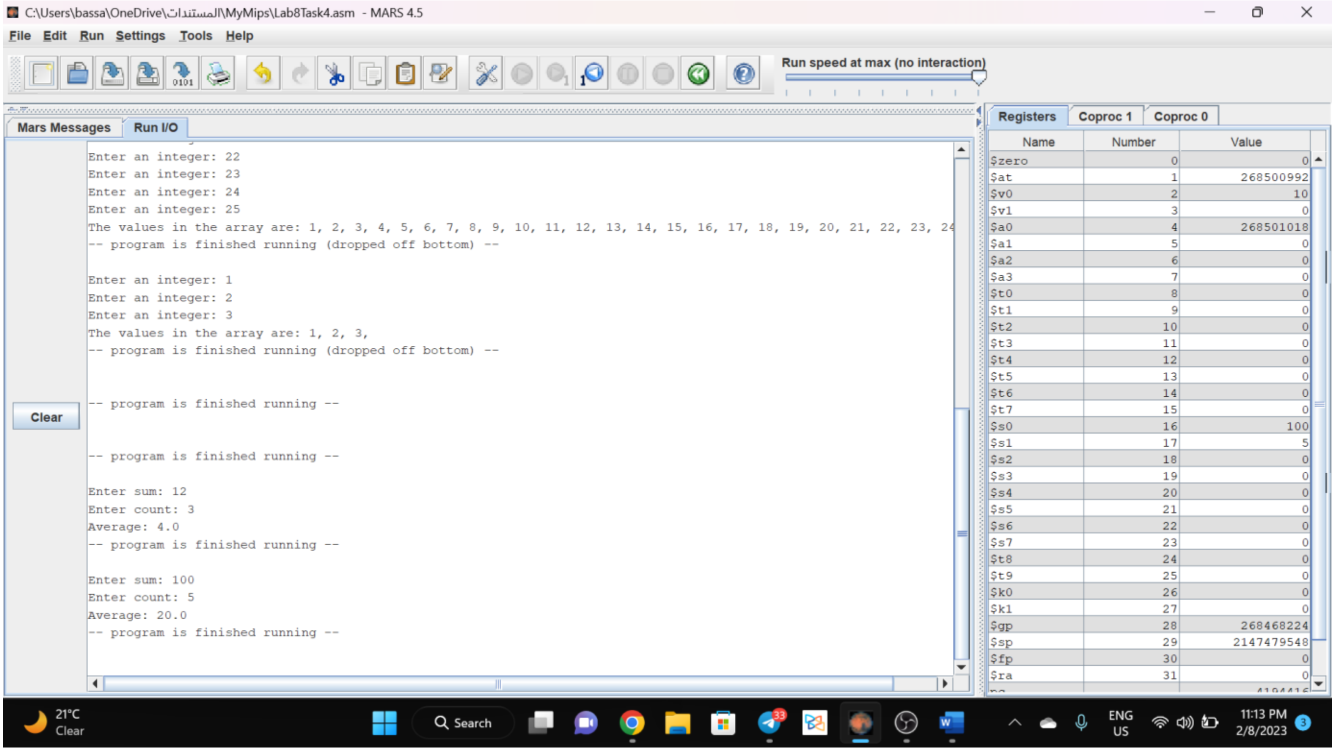
$f0 = 5.6e+20
$f1 = -5.6e+20
$f2 = 1.2
$f3 = $f0 + $f1 = 0
$f4 = $f2 + $f3 = 1.2
$f5 = 1.2
$f6 = 5.6e+20
$f7 = -5.6e+20
$f8 = $f5 + $f6 = 5.6e+20
$f9 = $f7 + $f8 = 0

The sum of elements of "array1" is 1.2 and the sum of elements of "array2" is 0. This shows that the order of elements in a floating-point addition operation can affect the result due to the limited precision of floating-point numbers.

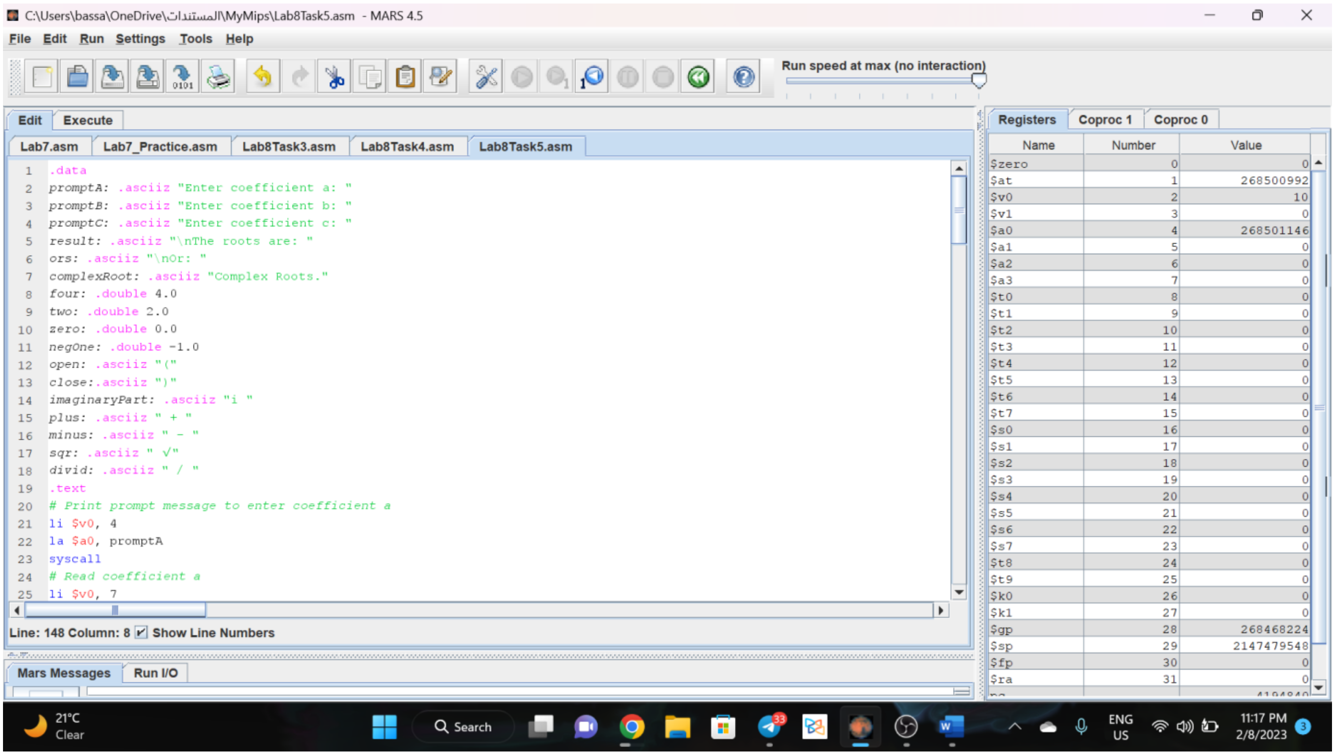
4. Write an interactive program that inputs an integer **sum** and an integer **count**, computes, and displays the **average = (float) sum / (float) count** as a single-precision floating-point number. Hint: use the proper convert instruction to convert **sum** and **count** from integer word into single-precision float.

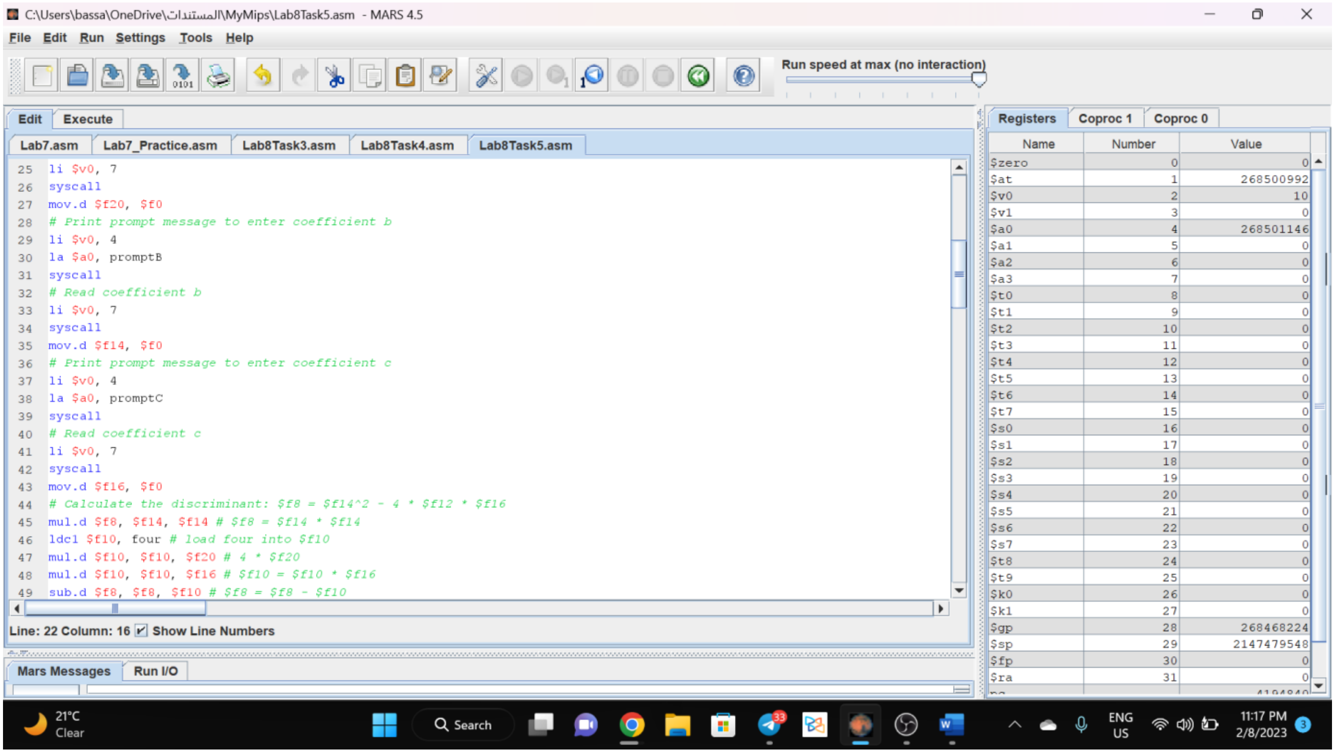


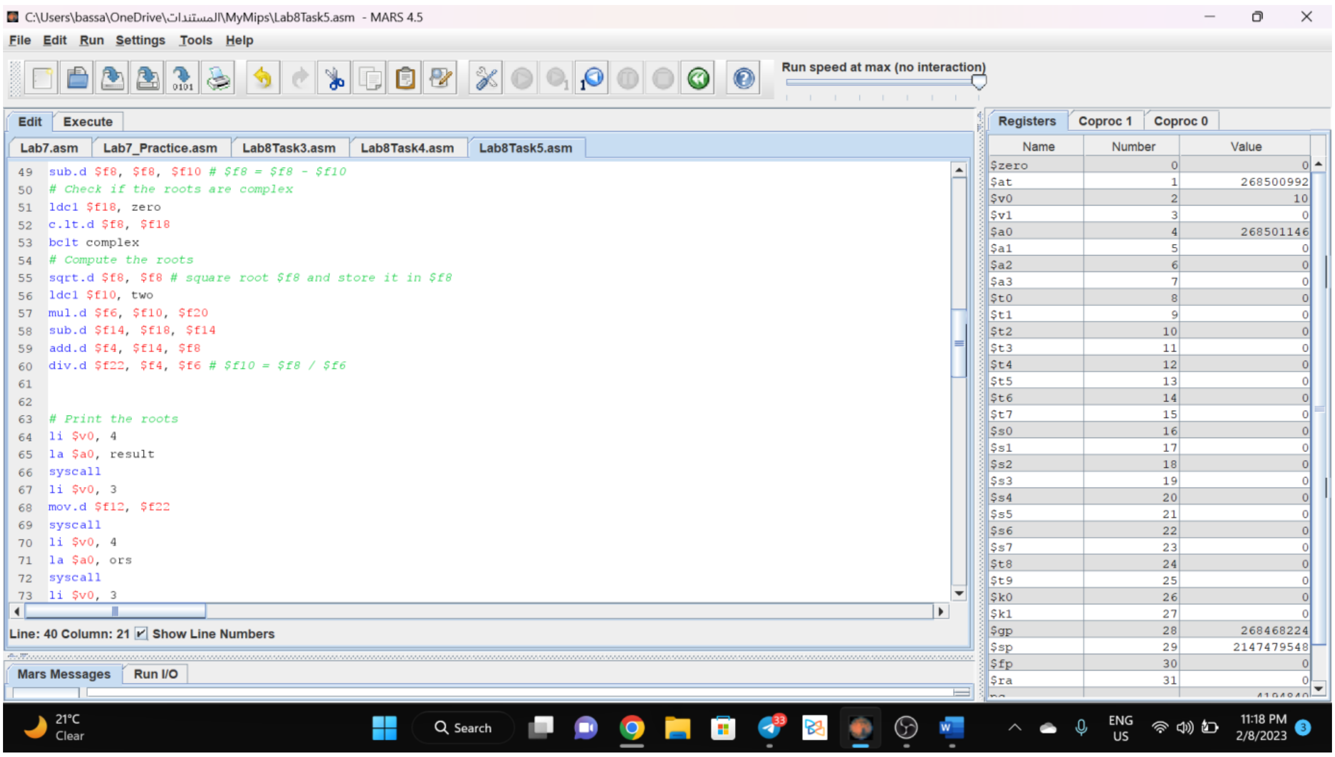


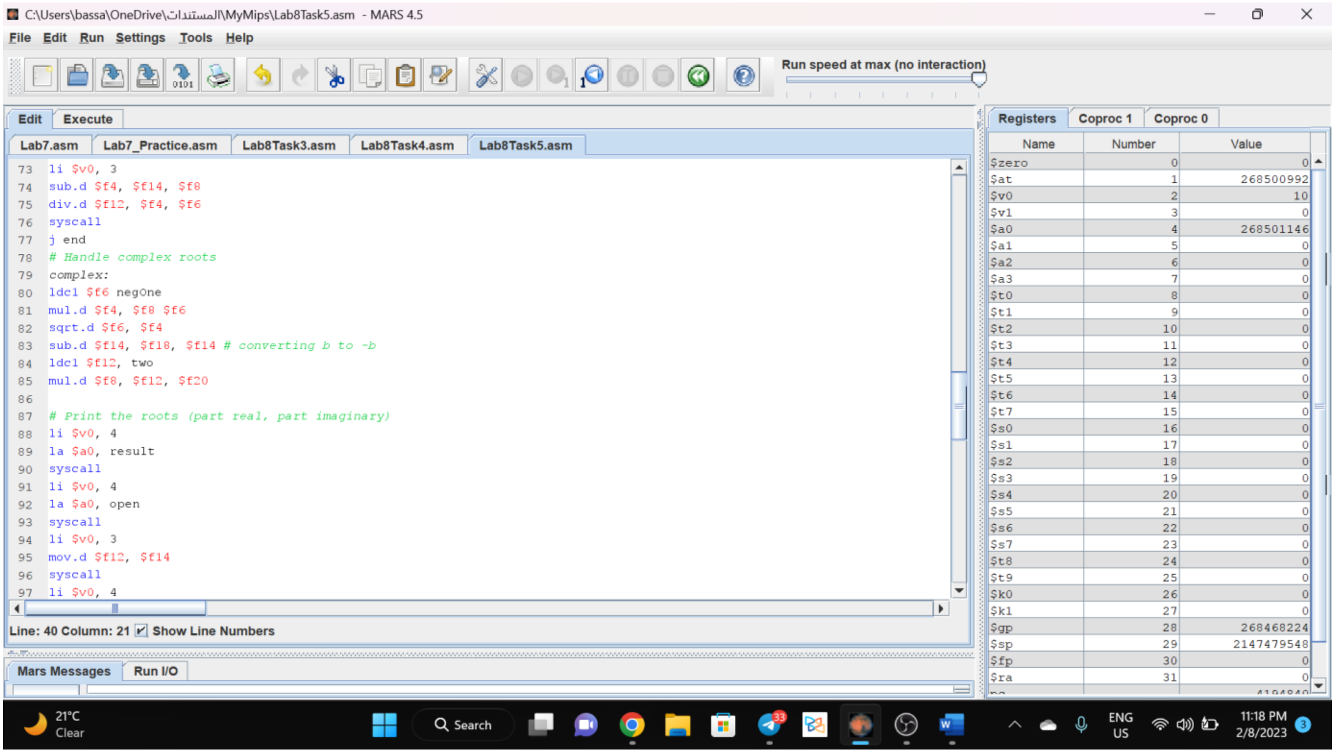


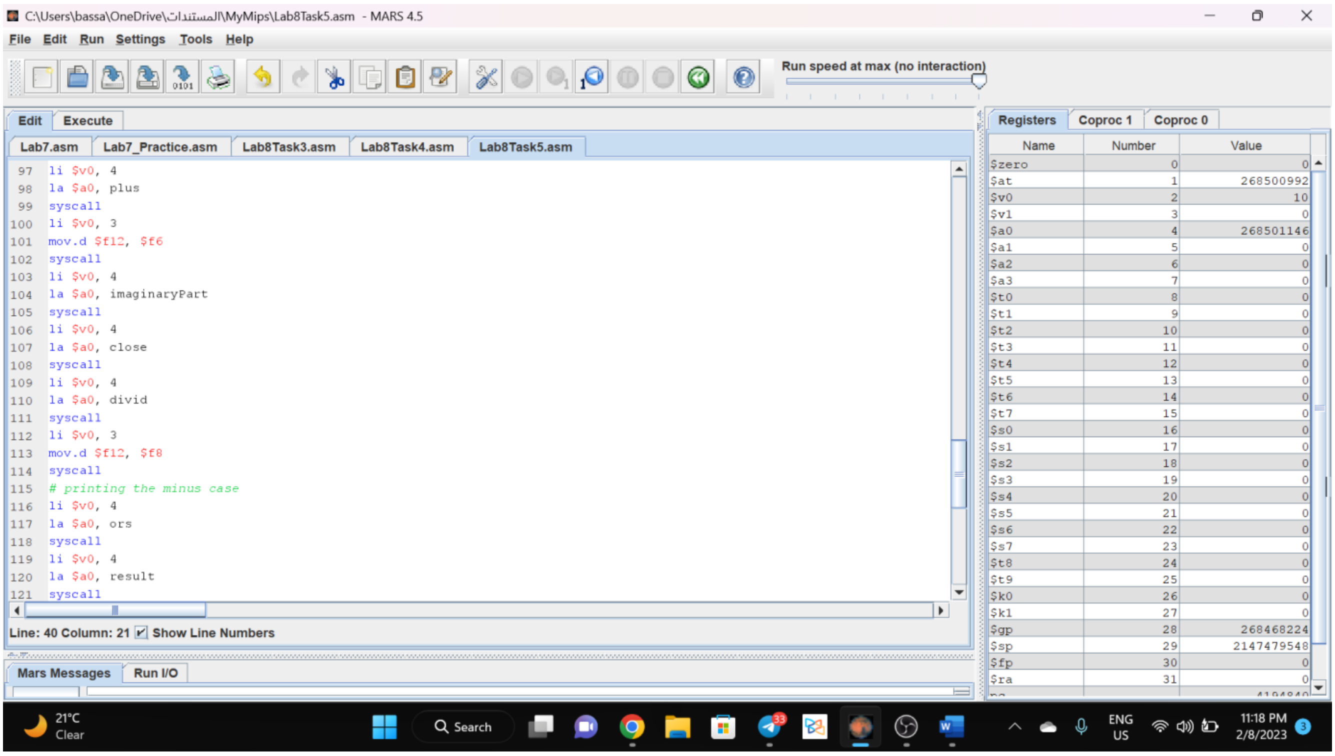
5. Write an interactive program that inputs the coefficient of a quadratic equation, computes, and displays the roots of the quadratic equation. All input, computation, and output should be done using double-precision floating-point instructions and registers. The program should handle the case of complex roots and displays the results properly.

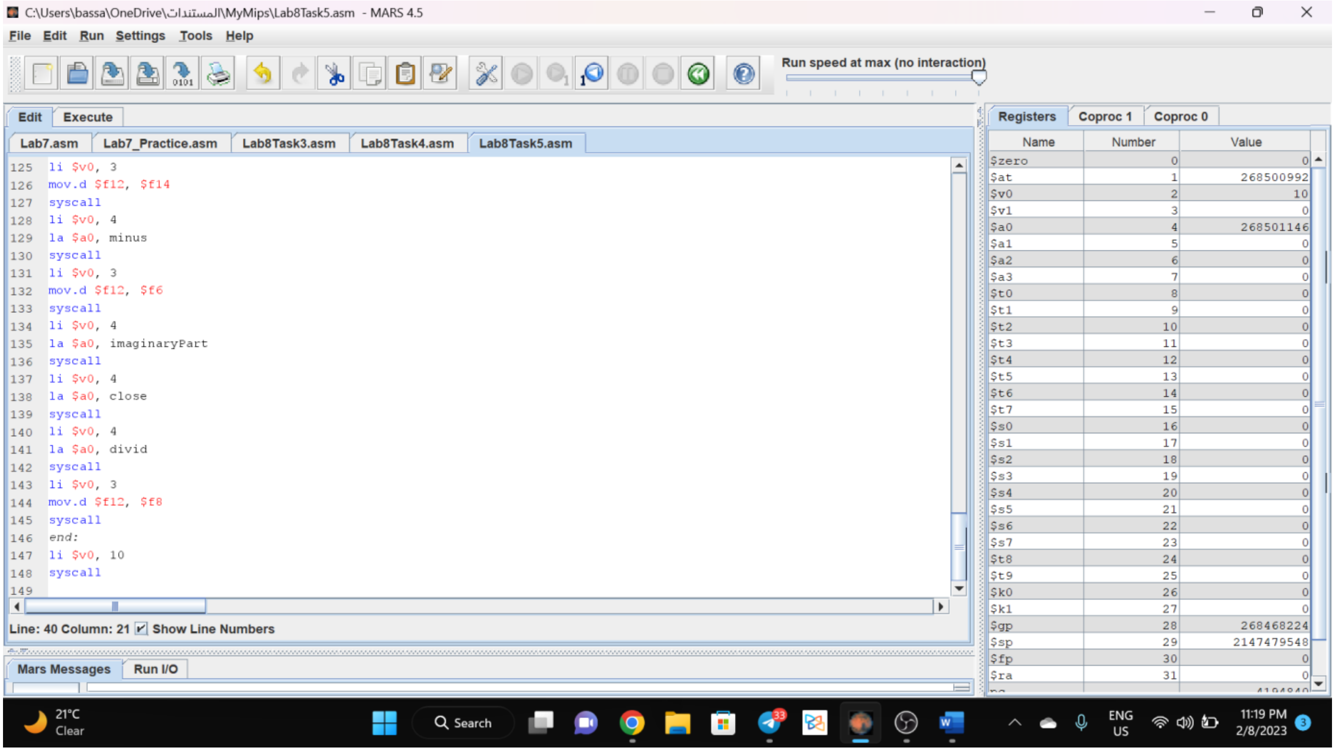



C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm
li $v0, 7
syscall
mov.d $f20, $f0
Print prompt message to enter coefficient b
li $v0, 4
la $a0, promptB
syscall
Read coefficient b
li $v0, 7
syscall
mov.d $f14, $f0
Print prompt message to enter coefficient c
li $v0, 4
la $a0, promptC
syscall
Read coefficient c
li $v0, 7
syscall
mov.d $f16, $f0
Calculate the discriminant: $f8 = $f14^2 - 4 * $f12 * $f16
mul.d $f8, $f14, $f14 # $f8 = $f14 * $f14
ldc1 $f10, four # load four into $f10
mul.d $f10, $f10, $f20 # 4 * $f20
mul.d $f10, $f10, $f16 # $f10 = $f10 * $f16
sub.d $f8, $f8, $f10 # $f8 = $f8 - $f10
Line: 22 Column: 16 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
11:17 PM
2/8/2023


C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm
sub.d $f8, $f8, $f10 # $f8 = $f8 - $f10
Check if the roots are complex
ldc1 $f18, zero
c.lt.d $f8, $f18
bc1t complex
Compute the roots
sqrt.d $f8, $f8 # square root $f8 and store it in $f8
ldc1 $f10, two
mul.d $f6, $f10, $f20
sub.d $f14, $f18, $f14
add.d $f4, $f14, $f8
div.d $f22, $f4, $f6 # $f10 = $f8 / $f6
Print the roots
li $v0, 4
la $a0, result
syscall
li $v0, 3
mov.d $f12, $f22
syscall
li $v0, 4
la $a0, ors
syscall
li $v0, 3
Line: 40 Column: 21 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
11:18 PM
2/8/2023


C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm
li $v0, 3
sub.d $f4, $f14, $f8
div.d $f12, $f4, $f6
syscall
j end
Handle complex roots
complex:
ldc1 $f6 negOne
mul.d $f4, $f8 $f6
sqrt.d $f6, $f4
sub.d $f14, $f18, $f14 # converting b to -b
ldc1 $f12, two
mul.d $f8, $f12, $f20
Print the roots (part real, part imaginary)
li $v0, 4
la $a0, result
syscall
li $v0, 4
la $a0, open
syscall
li $v0, 3
mov.d $f12, $f14
syscall
li $v0, 4
Line: 40 Column: 21 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
21°C Clear
Search
ENG US
11:18 PM 2/8/2023

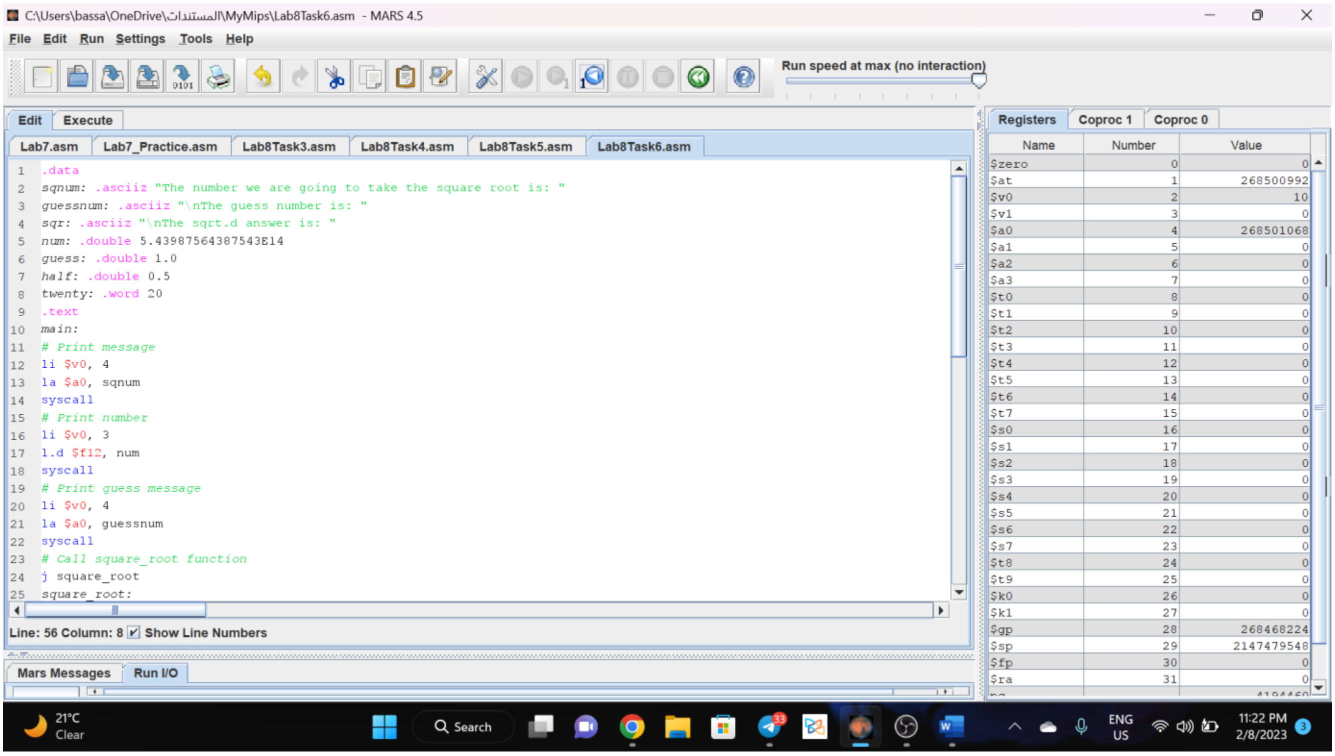

C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm
li $v0, 4
la $a0, plus
syscall
li $v0, 3
mov.d $f12, $f6
syscall
li $v0, 4
la $a0, imaginaryPart
syscall
li $v0, 4
la $a0, close
syscall
li $v0, 4
la $a0, divid
syscall
li $v0, 3
mov.d $f12, $f8
syscall
printing the minus case
li $v0, 4
la $a0, ors
syscall
li $v0, 4
la $a0, result
syscall
Line: 40 Column: 21 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
21°C Clear
Search
ENG US
11:18 PM 2/8/2023

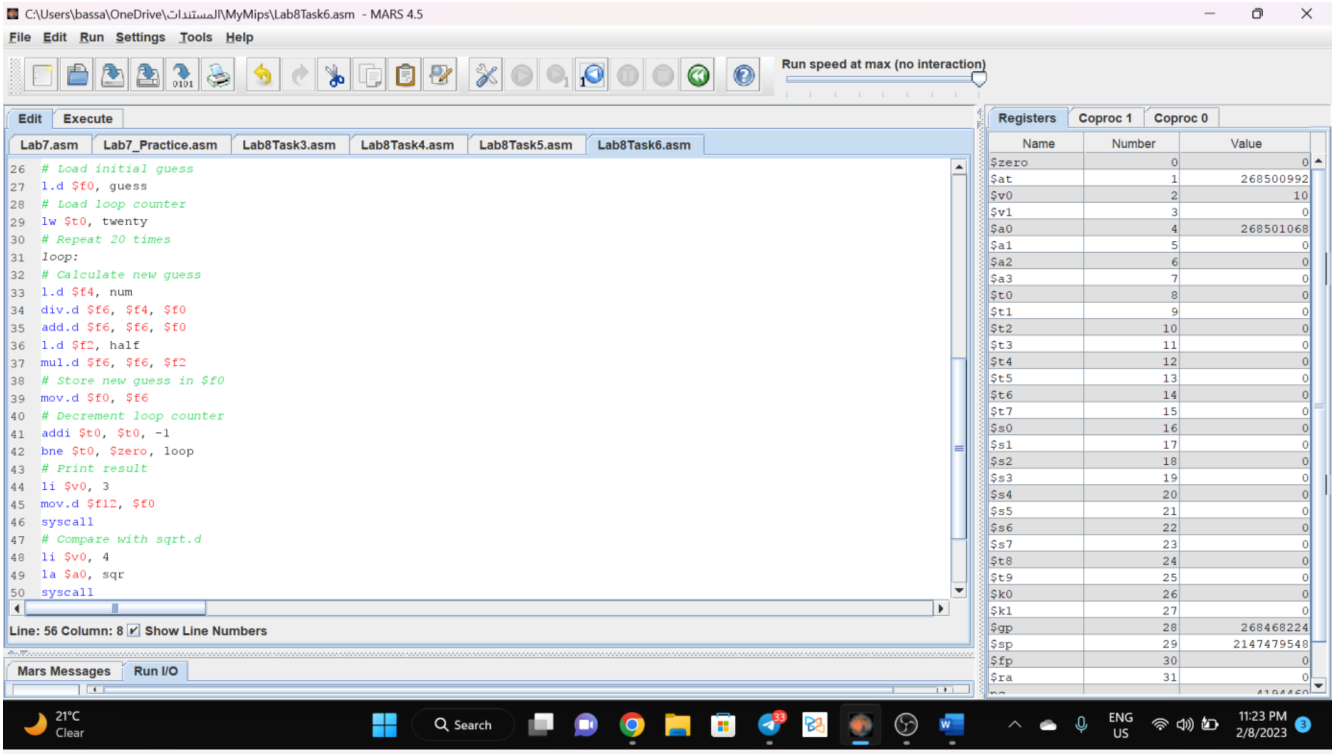

C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm
li $v0, 3
mov.d $f12, $f14
syscall
li $v0, 4
la $a0, minus
syscall
li $v0, 3
mov.d $f12, $f6
syscall
li $v0, 4
la $a0, imaginaryPart
syscall
li $v0, 4
la $a0, close
syscall
li $v0, 4
la $a0, divid
syscall
li $v0, 3
mov.d $f12, $f8
syscall
end:
li $v0, 10
syscall
Line: 40 Column: 21 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0
$at 1 268500992
$v0 2 10
$v1 3 0
$a0 4 268501146
$a1 5 0
$a2 6 0
$a3 7 0
$t0 8 0
$t1 9 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 0
$s1 17 0
$s2 18 0
$s3 19 0
$s4 20 0
$s5 21 0
$s6 22 0
$s7 23 0
$t8 24 0
$t9 25 0
$k0 26 0
$k1 27 0
$gp 28 268468224
$sp 29 2147479548
$fp 30 0
$ra 31 0
21°C Clear
Search
ENG US
11:19 PM 2/8/2023

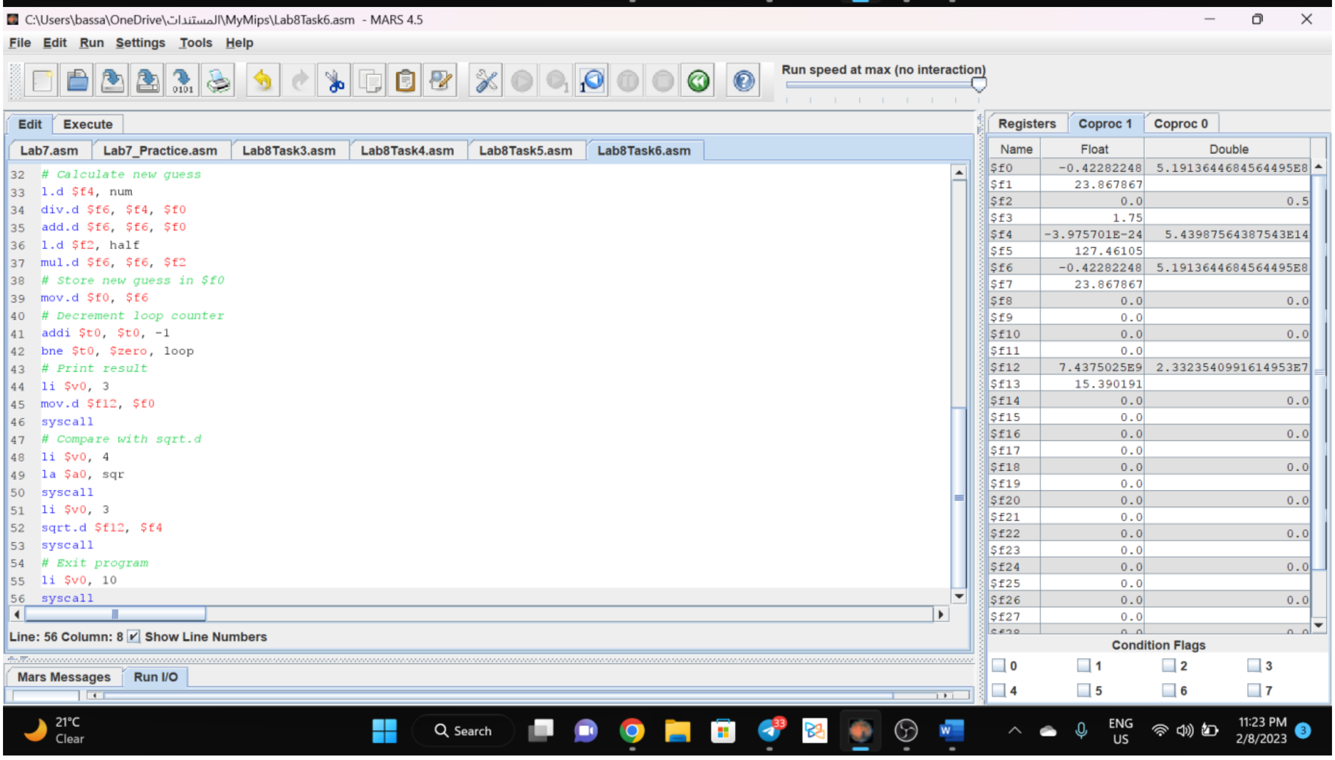
6. Square Root Calculation: Newton's iterative method can be used to approximate the square root of a number **x**. Let the initial **guess** be **1**. Then each new **guess** can be computed as follows:

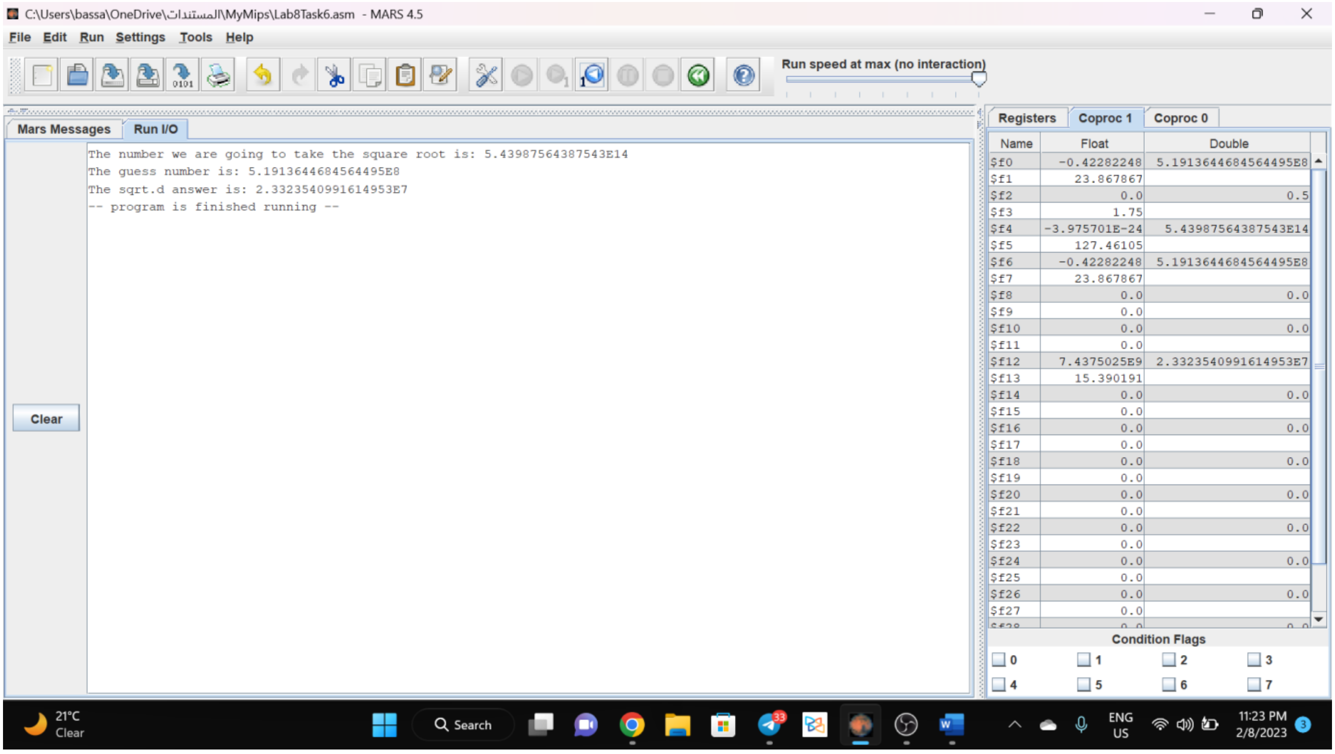
`guess = ((x/guess) + guess) / 2;`

Write a function called **square_root** that receives a double-precision parameter **x**, computes, and returns the approximated value of the square root of **x**. Write a loop that repeats 20 times and computes 20 **guess** values, then returns the final **guess** after 20 iterations. Use the MIPS floating-point register convention to pass the parameter **x** and to return the function result. All computation should be done using double-precision floating-point instructions and registers. Compare the result of the **sqrt.d** instruction against the result of your **square_root** function. What is the error in absolute value?




C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task6.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm Lab8Task6.asm
Load initial guess
l.d $f0, guess
# Load loop counter
lw $t0, twenty
# Repeat 20 times
loop:
Calculate new guess
l.d $f4, num
div.d $f6, $f4, $f0
add.d $f6, $f6, $f0
l.d $f2, half
mul.d $f6, $f6, $f2
Store new guess in $f0
mov.d $f0, $f6
# Decrement loop counter
addi $t0, $t0, -1
bne $t0, $zero, loop
Print result
li $v0, 3
mov.d $f12, $f0
syscall
Compare with sqrt.d
li $v0, 4
la $a0, sqr
syscall
Line: 56 Column: 8 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0
$at 1 268500992
$v0 2 10
$v1 3 0
$a0 4 268501068
$a1 5 0
$a2 6 0
$a3 7 0
$t0 8 0
$t1 9 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 0
$s1 17 0
$s2 18 0
$s3 19 0
$s4 20 0
$s5 21 0
$s6 22 0
$s7 23 0
$t8 24 0
$t9 25 0
$k0 26 0
$k1 27 0
$gp 28 268468224
$sp 29 2147479548
$fp 30 0
$ra 31 0
21°C Clear
Search
ENG US
11:23 PM 2/8/2023


C:\Users\bassa\OneDrive\المستندات\MyMips\Lab8Task6.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Lab7.asm Lab7_Practice.asm Lab8Task3.asm Lab8Task4.asm Lab8Task5.asm Lab8Task6.asm
Calculate new guess
l.d $f4, num
div.d $f6, $f4, $f0
add.d $f6, $f6, $f0
l.d $f2, half
mul.d $f6, $f6, $f2
Store new guess in $f0
mov.d $f0, $f6
# Decrement loop counter
addi $t0, $t0, -1
bne $t0, $zero, loop
Print result
li $v0, 3
mov.d $f12, $f0
syscall
Compare with sqrt.d
li $v0, 4
la $a0, sqr
syscall
li $v0, 3
sqrt.d $f12, $f4
syscall
Exit program
li $v0, 10
syscall
Line: 56 Column: 8 Show Line Numbers
Mars Messages Run I/O
Registers Coproc 1 Coproc 0
Name Float Double
$f0 -0.42282248 5.191364468456449 5E8
$f1 23.867867
$f2 0.0 0.5
$f3 1.75
$f4 -3.975701E-24 5.43987564387543E14
$f5 127.46105
$f6 -0.42282248 5.191364468456449 5E8
$f7 23.867867
$f8 0.0 0.0
$f9 0.0
$f10 0.0 0.0
$f11 0.0
$f12 7.4375025E9 2.3323540991614953E7
$f13 15.390191
$f14 0.0 0.0
$f15 0.0
$f16 0.0 0.0
$f17 0.0
$f18 0.0 0.0
$f19 0.0
$f20 0.0 0.0
$f21 0.0
$f22 0.0 0.0
$f23 0.0
$f24 0.0 0.0
$f25 0.0
$f26 0.0 0.0
$f27 0.0
Condition Flags
0 1 2 3
4 5 6 7
21°C Clear
Search
ENG US
11:23 PM 2/8/2023



The difference gets bigger when the number is big. We can fix that by increasing the loops from 20 to something bigger.