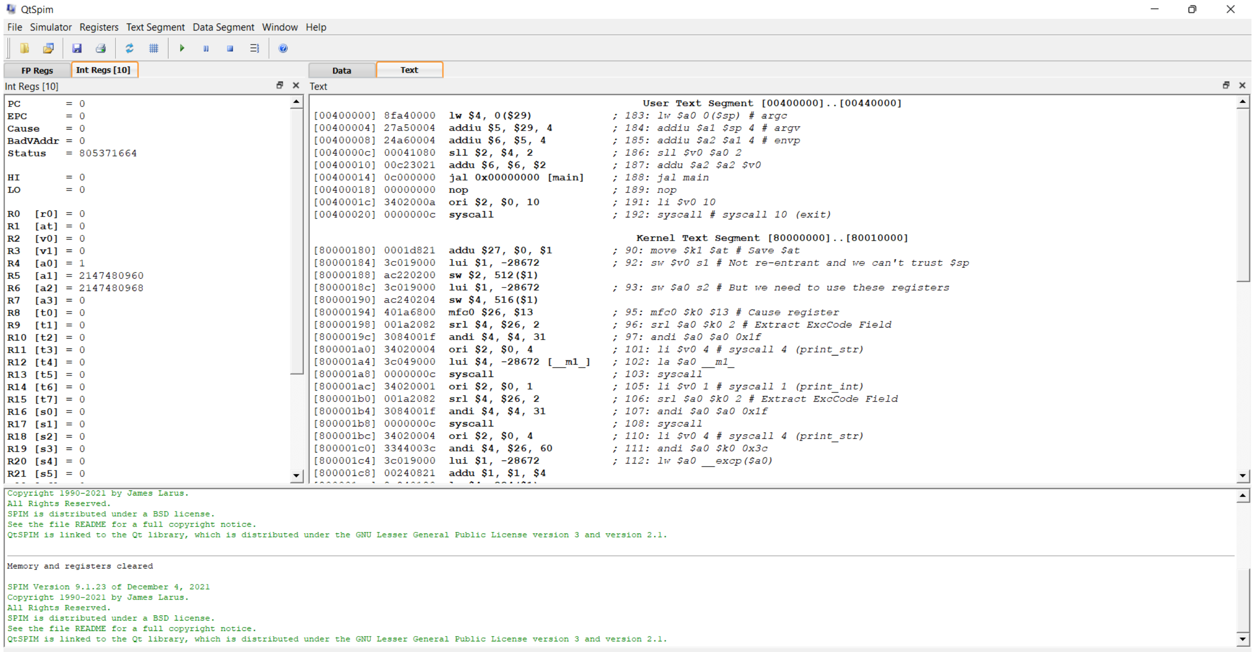
Lab5
CS211
Floating Point Arithmetic

Name-Pranav Tambe
Roll No-2106339


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Int Regs [10]
Data
Text
Text

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480960
R6  [a2] = 2147480968
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                          User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)

                                         Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                   ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Interface before loading any file

Q.1)Evaluate the expression 2.5x 2 +4.3x+5. Take x as input from the user. Display the result.

```
# This program prompts the user to enter a value for x, and calculates and
displays the result of a mathematical expression

    .data
string_one:   .asciiz   "Enter a value for x: "    # Prompt for user input
result:       .asciiz   "The result is: "          # Message to display the
result
two_point_five: .float 2.5                          # Initialize a
floating-point constant
```

```
four_point_three: .float 4.3
five:         .float    5.0
.text
.globl main

main:

    li $v0, 4
    la $a0, string_one
    syscall  # Print the prompt for user input

    li $v0, 6
    syscall
    mov.s $f0, $f0   # Read a floating-point value from the user and store
it in $f0

    mul.s  $f1, $f0, $f0     # Calculate $f1 = x^2
    l.s $f2, two_point_five    # Load the constant 2.5 into $f2
    mul.s $f3, $f1, $f2       # Calculate $f3 = 2.5 * x^2
    l.s $f4, four_point_three   # Load the constant 4.3 into $f4
    mul.s $f5, $f4, $f0    # Calculate $f5 = 4.3 * x
    add.s $f6, $f5, $f3      # Calculate $f6 = 4.3 * x + 2.5 * x^2
    l.s $f7, five     # Load the constant 5.0 into $f7
    add.s $f8, $f6, $f7    # Calculate the final result $f8 = 4.3 * x +
2.5 * x^2 + 5.0

    li $v0, 4
    la $a0, result
    syscall      # Print the message indicating the result will be printed
    li $v0, 2
    mov.s $f12, $f8
    syscall    # Print the final result value in $f8 using syscall 2
(print float)

    li $v0, 10
    syscall    # Terminate the program
```

Brief overview of the code section This program prompts the user to enter a value for x, reads the input value as a floating-point number, and then calculates the value of the expression 4.3 * x + 2.5 * x^2 + 5.0. The result is then printed to the console.

Here's how the program works:

The program first defines three floating-point constants: 2.5, 4.3, and 5.0.

The program then prompts the user to enter a value for x, which is read as a floating-point number using syscall 6 (read float).

The program then calculates the value of the expression 4.3 * x + 2.5 * x^2 + 5.0 using the following steps:

Multiply x by itself to get x^2 using mul.s.

Multiply x^2 by 2.5 to get 2.5 * x^2 using mul.s.

Load the constant 4.3 into $f4 using l.s.

Multiply x by 4.3 to get 4.3 * x using mul.s.

Add 4.3 * x and 2.5 * x^2 to get 4.3 * x + 2.5 * x^2 using add.s.

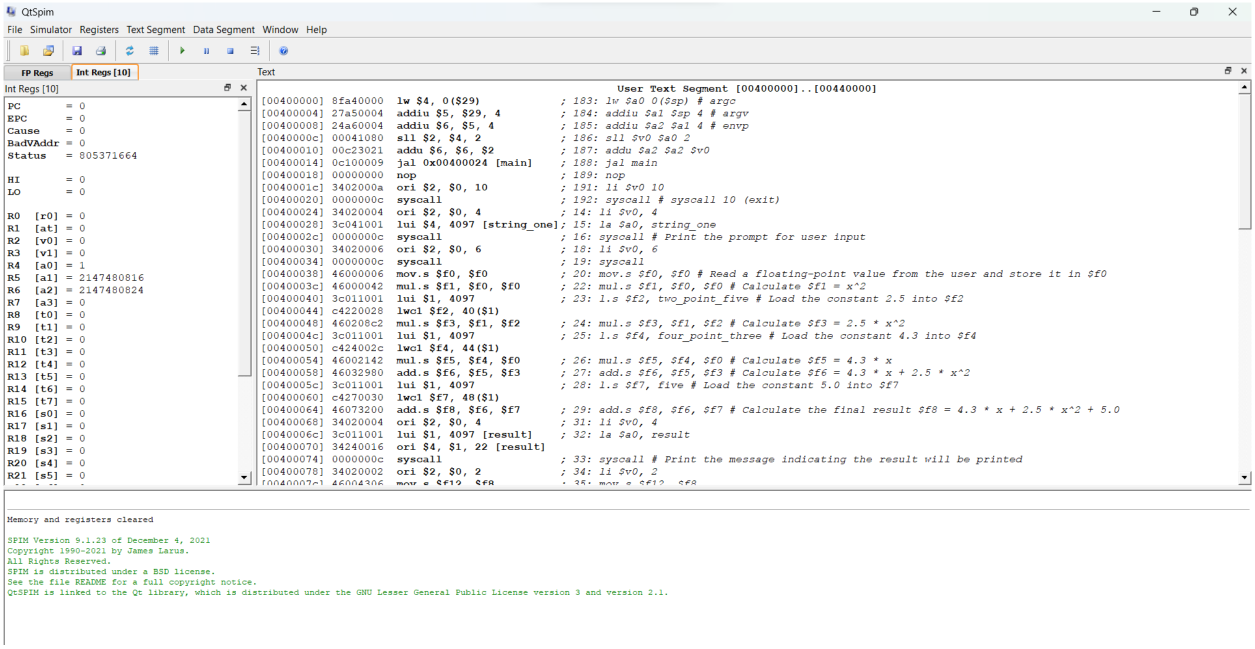
Load the constant 5.0 into $f7 using l.s.

Add 4.3 * x + 2.5 * x^2 and 5.0 to get the final result using add.s.

The program then prints the final result to the console using syscall 2 (print float).

Finally, the program terminates using syscall 10 (exit).

After loading the file

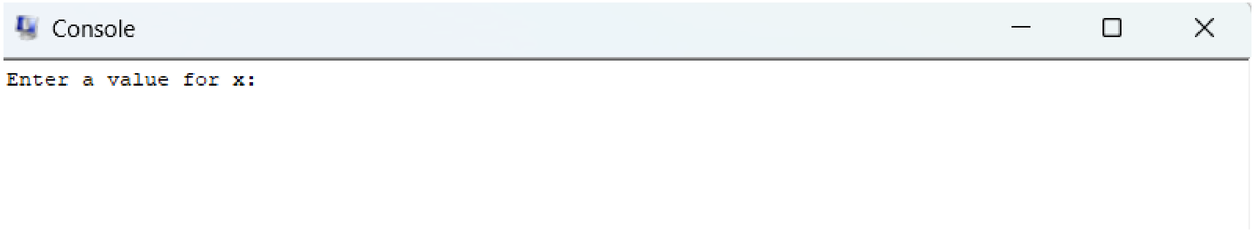

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text
Int Regs [10]

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480816
R6  [a2] = 2147480824
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

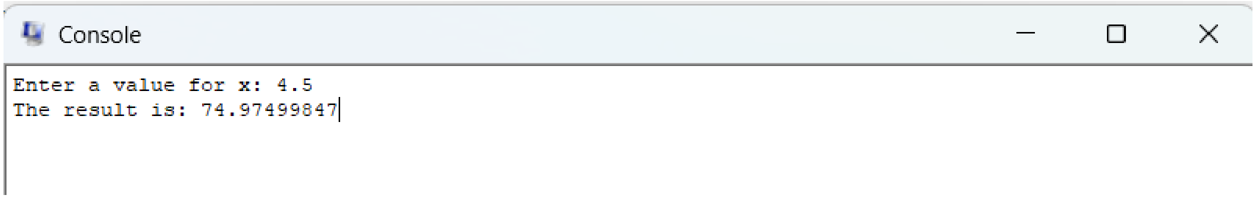
```
                              User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4            ; 14: li $v0, 4
[00400028] 3c041001  lui $4, 4097 [string_one]; 15: la $a0, string_one
[0040002c] 0000000c  syscall                  ; 16: syscall # Print the prompt for user input
[00400030] 34020006  ori $2, $0, 6            ; 18: li $v0, 6
[00400034] 0000000c  syscall                  ; 19: syscall
[00400038] 46000006  mov.s $f0, $f0           ; 20: mov.s $f0, $f0 # Read a floating-point value from the user and store it in $f0
[0040003c] 46000042  mul.s $f1, $f0, $f0      ; 22: mul.s $f1, $f0, $f0 # Calculate $f1 = x^2
[00400040] 3c011001  lui $1, 4097             ; 23: l.s $f2, two_point_five # Load the constant 2.5 into $f2
[00400044] c4220028  lwc1 $f2, 40($1)
[00400048] 460208c2  mul.s $f3, $f1, $f2      ; 24: mul.s $f3, $f1, $f2 # Calculate $f3 = 2.5 * x^2
[0040004c] 3c011001  lui $1, 4097             ; 25: l.s $f4, four_point_three # Load the constant 4.3 into $f4
[00400050] c424002c  lwc1 $f4, 44($1)
[00400054] 46002142  mul.s $f5, $f4, $f0      ; 26: mul.s $f5, $f4, $f0 # Calculate $f5 = 4.3 * x
[00400058] 46032980  add.s $f6, $f5, $f3      ; 27: add.s $f6, $f5, $f3 # Calculate $f6 = 4.3 * x + 2.5 * x^2
[0040005c] 3c011001  lui $1, 4097             ; 28: l.s $f7, five # Load the constant 5.0 into $f7
[00400060] c4270030  lwc1 $f7, 48($1)
[00400064] 46073200  add.s $f8, $f6, $f7      ; 29: add.s $f8, $f6, $f7 # Calculate the final result $f8 = 4.3 * x + 2.5 * x^2 + 5.0
[00400068] 34020004  ori $2, $0, 4            ; 31: li $v0, 4
[0040006c] 3c011001  lui $1, 4097 [result]    ; 32: la $a0, result
[00400070] 34240016  ori $4, $1, 22 [result]
[00400074] 0000000c  syscall                  ; 33: syscall # Print the message indicating the result will be printed
[00400078] 34020002  ori $2, $0, 2            ; 34: li $v0, 2
[0040007c] 46004306  mov.s $f12, $f8          ; 35: mov.s $f12, $f8
```

```
Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Console



```
Enter a value for x:
```

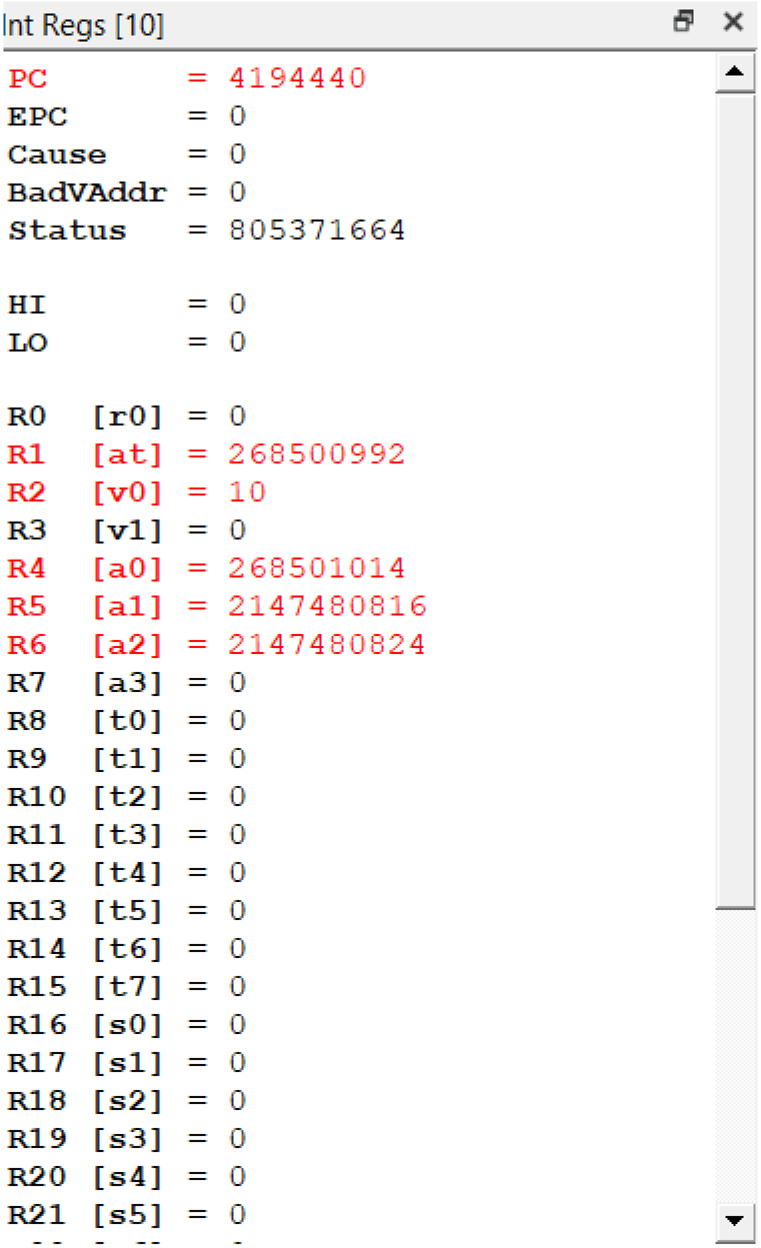


```
Enter a value for x: 4.5
The result is: 74.97499847
```

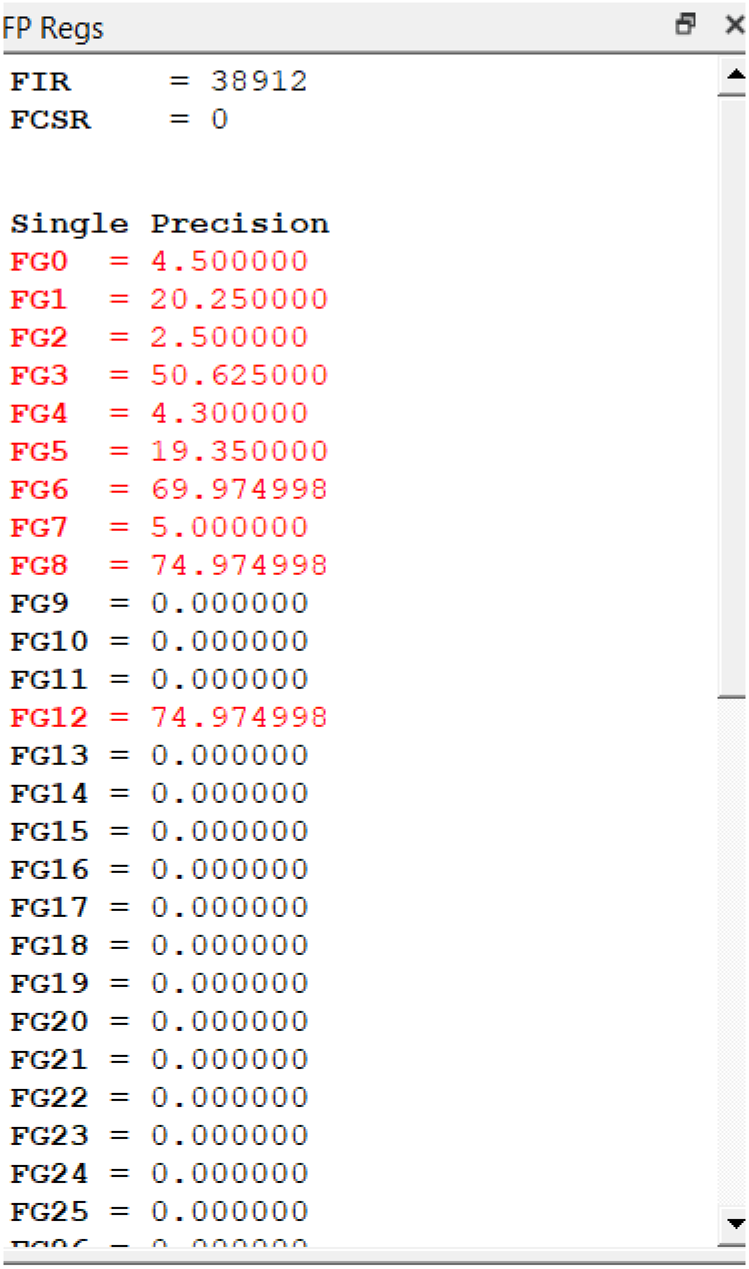
X=4.5

3.5*4.5*4.5+4.3*4.5+5=74.975

Registers after execution of the code



```
PC         = 4194440
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
LO         = 0

R0   [r0]  = 0
R1   [at]  = 268500992
R2   [v0]  = 10
R3   [v1]  = 0
R4   [a0]  = 268501014
R5   [a1]  = 2147480816
R6   [a2]  = 2147480824
R7   [a3]  = 0
R8   [t0]  = 0
R9   [t1]  = 0
R10  [t2]  = 0
R11  [t3]  = 0
R12  [t4]  = 0
R13  [t5]  = 0
R14  [t6]  = 0
R15  [t7]  = 0
R16  [s0]  = 0
R17  [s1]  = 0
R18  [s2]  = 0
R19  [s3]  = 0
R20  [s4]  = 0
R21  [s5]  = 0
```



```
FIR      = 38912
FCSR     = 0


Single Precision
FG0   = 4.500000
FG1   = 20.250000
FG2   = 2.500000
FG3   = 50.625000
FG4   = 4.300000
FG5   = 19.350000
FG6   = 69.974998
FG7   = 5.000000
FG8   = 74.974998
FG9   = 0.000000
FG10  = 0.000000
FG11  = 0.000000
FG12  = 74.974998
FG13  = 0.000000
FG14  = 0.000000
FG15  = 0.000000
FG16  = 0.000000
FG17  = 0.000000
FG18  = 0.000000
FG19  = 0.000000
FG20  = 0.000000
FG21  = 0.000000
FG22  = 0.000000
FG23  = 0.000000
FG24  = 0.000000
FG25  = 0.000000
FG26  = 0.000000
```

Q.2)Find the square root of a number entered by the user. Apply Newton's method to perform the calculations (Accurate up to 5 places of decimal).
Hint: Newton's method is a way to compute the square root of a number. Say that n is the number and that x is an approximation to the square root of n.
Then:x' =(1/2)(x + n/x)
x'; is an even better approximation to the square root.
If x reaches the exact value, it stays fixed at that value.

```
.data
prompt:  .asciiz "Enter a number: "  # Declares a string for user prompt
```

```
result: .asciiz "The square root is: "  # Declares a string for result output
one: .float 1.0  # Declares a float value of 1.0
half: .float 0.5  # Declares a float value of 0.5
.text
main:
  li $v0, 4  # Loads 4 into register $v0, which is the system call code for printing a string
  la $a0, prompt  # Loads the address of the prompt string into register $a0, which is the argument
  syscall  # Executes the system call to print the prompt string

  li $v0, 6  # Loads 6 into register $v0, which is the system call code for reading a float
  syscall  # Executes the system call to read a float from the user
  mov.s $f0, $f0  # Moves the float value read by the user from register $f0 to $f0

  l.s $f1, one  # Loads the float value 1.0 from memory into register $f1
  l.s $f7, half  # Loads the float value 0.5 from memory into register $f7

  add.s $f2, $f0, $f1  # Adds the float value read by the user and 1.0, and stores the result in register $f2
  mul.s $f3, $f2, $f7  # Multiplies the result in register $f2 by 0.5, and stores the result in register $f3

loop:
  div.s $f4, $f0, $f3  # Divides the float value read by the user by the result in register $f3, and stores the result in register $f4
  add.s $f5, $f3, $f4  # Adds the result in register $f3 and $f4, and stores the result in register $f5
  mul.s $f6, $f5, $f7  # Multiplies the result in register $f5 by 0.5, and stores the result in register $f6

  c.eq.s $f3, $f6  # Compares the results in registers $f3 and $f6 for equality
  bc1t done  # Branches to the label "done" if the result of the comparison was true
  mov.s $f3, $f6  # Moves the result in register $f6 to register $f3
  j loop  # Jumps to the label "loop" to repeat the loop
```

```
done:
  li $v0, 4  # Loads 4 into register $v0, which is the system call code
for printing a string
  la $a0, result  # Loads the address of the result string into register
$a0, which is the argument
  syscall  # Executes the system call to print the result string

  mov.s $f12, $f3  # Moves the result in register $f3 to register $f12
  li $v0, 2  # Loads 2 into register $v0, which is the system call code
for printing a float
  syscall  # Executes the system call to print the square root value

  li $v0, 10  # Loads 10 into
  syscall
```

Brief overview of the code section

This program prompts the user to enter a number, reads the input as a float, and then uses the Newton-Raphson method to find the square root of the input. The program then outputs the square root to the console.

The program starts by printing a prompt to the console, asking the user to enter a number. It then reads the user's input as a float and stores it in register $f0.

The program then initializes two float values: 1.0 and 0.5, and stores them in registers $f1 and $f7, respectively. These values are used in the Newton-Raphson method to calculate the square root.

The program then enters a loop where it performs the following steps:

Divides the input float value in register $f0 by the value in register $f3, which is initialized to 0.5. The result is stored in register $f4.

Adds the value in register $f3 to the value in register $f4 and stores the result in register $f5.

Multiplies the value in register $f5 by 0.5 and stores the result in register $f6.

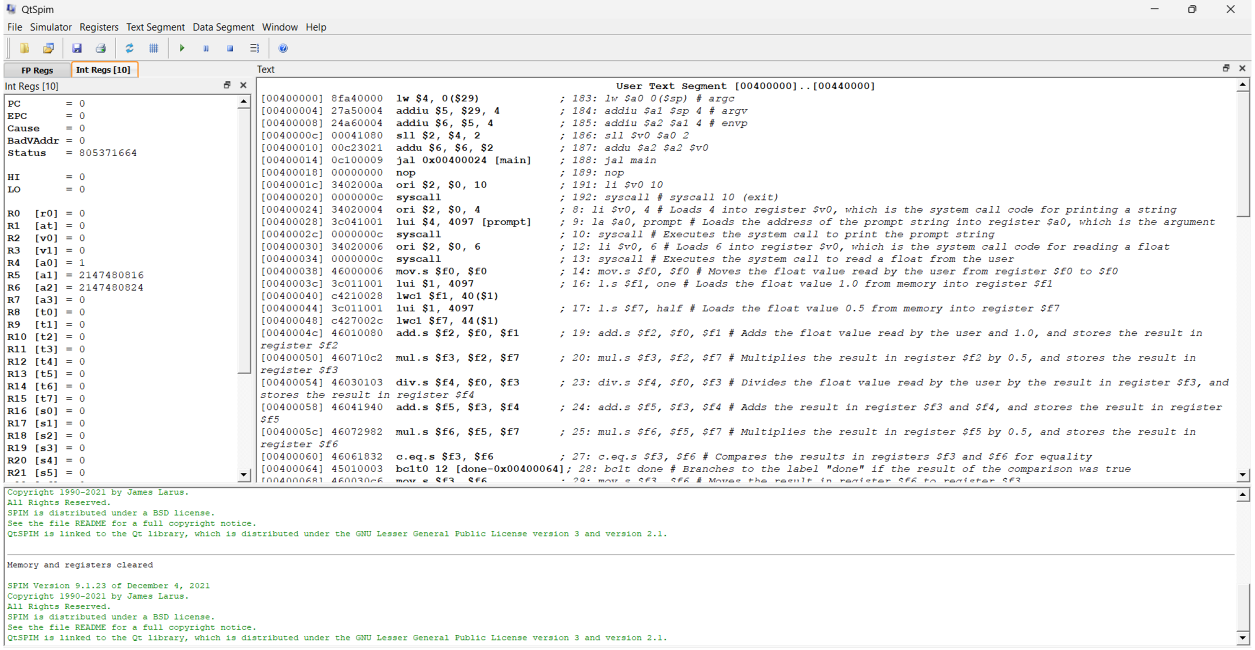
Compares the values in registers $f3 and $f6 for equality. If they are equal, the loop ends.

If the values in registers $f3 and $f6 are not equal, the value in register $f6 is moved to register $f3, and the loop repeats.

When the loop ends, the program prints a string indicating that the result is the square root, then outputs the value in register $f3 to the console as a float.

Finally, the program calls the system call to exit the program.

After loading the file

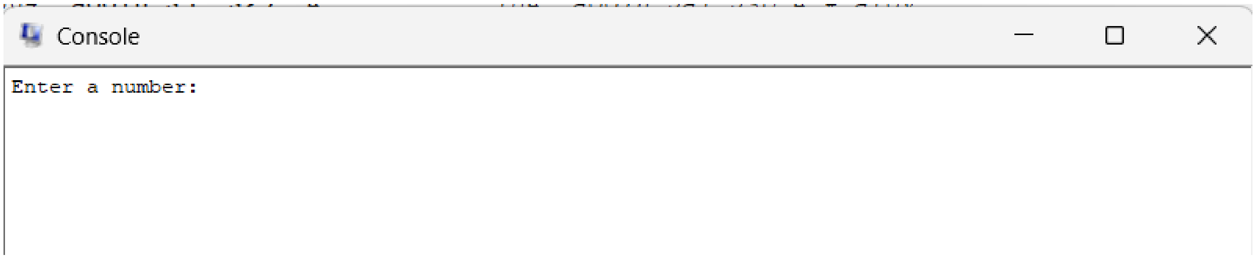

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text

```
Int Regs [10]
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480816
R6  [a2] = 2147480824
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

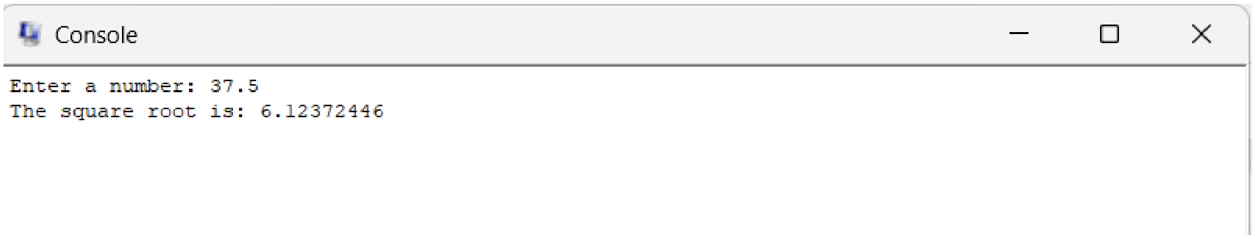
```
                                 User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4            ; 8: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400028] 3c041001  lui $4, 4097 [prompt]    ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                  ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6            ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                  ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0           ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097             ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097             ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1      ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7      ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3      ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4      ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7      ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6          ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6           ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

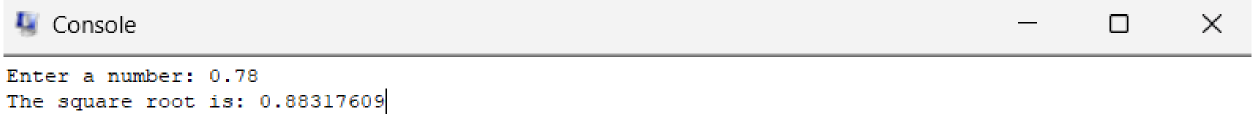
Console images



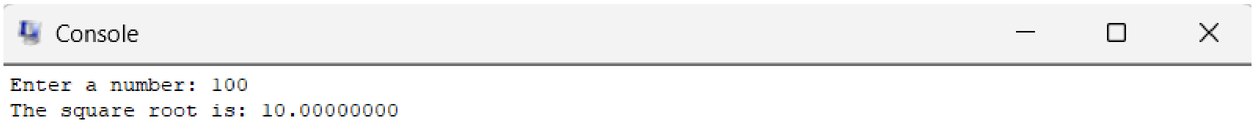
```
Enter a number:
```



```
Enter a number: 37.5
The square root is: 6.12372446
```

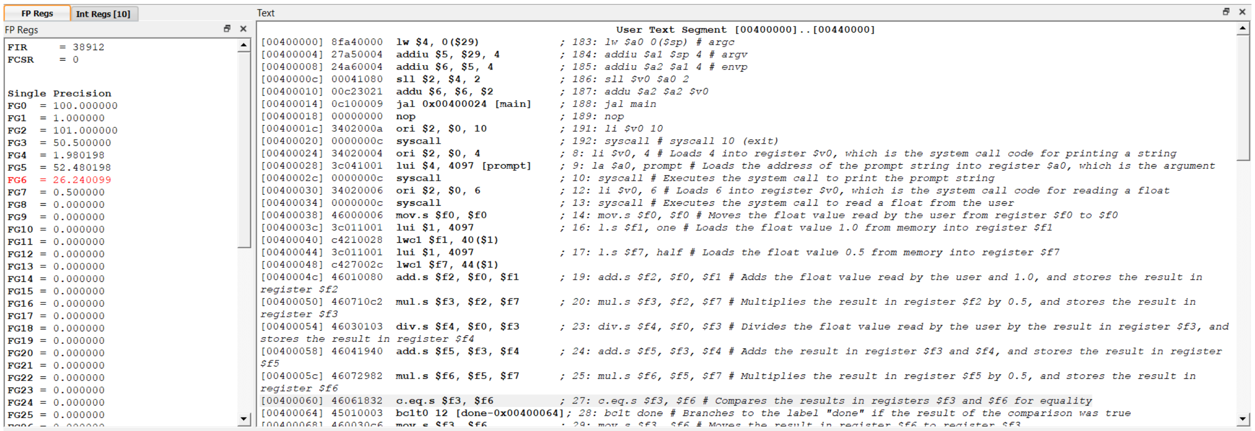


```
Enter a number: 0.78
The square root is: 0.88317609
```



```
Enter a number: 100
The square root is: 10.00000000
```

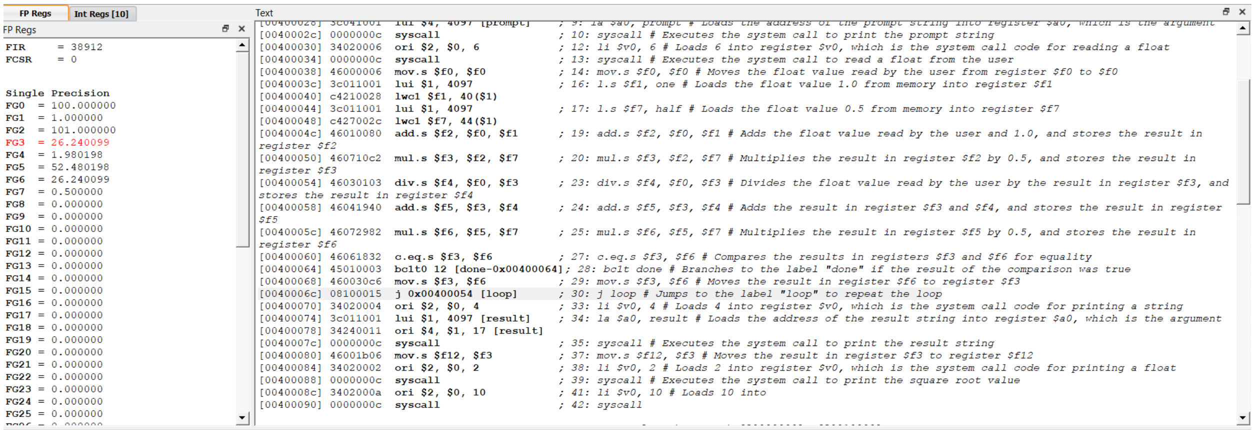
Changed approximation



```
FIR     = 38912
FCSR    = 0


Single Precision
FG0   = 100.000000
FG1   = 1.000000
FG2   = 101.000000
FG3   = 50.500000
FG4   = 1.980198
FG5   = 52.480198
FG6   = 26.240099
FG7   = 0.500000
FG8   = 0.000000
FG9   = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
FG26 = 0.000000
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4           ; 8: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400028] 3c041001  lui $4, 4097 [prompt]   ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                 ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6           ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                 ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0          ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097            ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097            ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1     ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7     ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3     ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4     ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7     ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6         ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6          ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
```

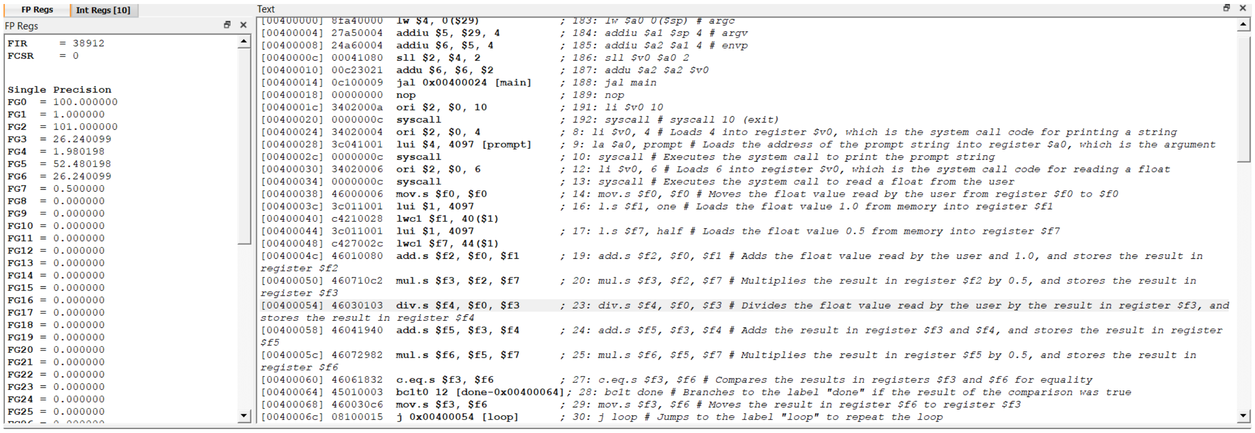
New approximation is stored



```
FIR     = 38912
FCSR    = 0


Single Precision
FG0   = 100.000000
FG1   = 1.000000
FG2   = 101.000000
FG3   = 26.240099
FG4   = 1.980198
FG5   = 52.480198
FG6   = 26.240099
FG7   = 0.500000
FG8   = 0.000000
FG9   = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
FG26 = 0.000000
```

```
[00400028] 3c041001  lui $4, 4097 [prompt]   ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                 ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6           ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                 ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0          ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097            ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097            ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1     ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7     ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3     ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4     ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7     ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6         ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6          ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
[0040006c] 08100015  j 0x00400054 [loop]     ; 30: j loop # Jumps to the label "loop" to repeat the loop
[00400070] 34020004  ori $2, $0, 4           ; 33: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400074] 3c011001  lui $1, 4097 [result]   ; 34: la $a0, result # Loads the address of the result string into register $a0, which is the argument
[00400078] 34240011  ori $4, $1, 17 [result]
[0040007c] 0000000c  syscall                 ; 35: syscall # Executes the system call to print the result string
[00400080] 46001b06  mov.s $f12, $f3         ; 37: mov.s $f12, $f3 # Moves the result in register $f3 to register $f12
[00400084] 34020002  ori $2, $0, 2           ; 38: li $v0, 2 # Loads 2 into register $v0, which is the system call code for printing a float
[00400088] 0000000c  syscall                 ; 39: syscall # Executes the system call to print the square root value
[0040008c] 3402000a  ori $2, $0, 10          ; 41: li $v0, 10 # Loads 10 into
[00400090] 0000000c  syscall                 ; 42: syscall
```

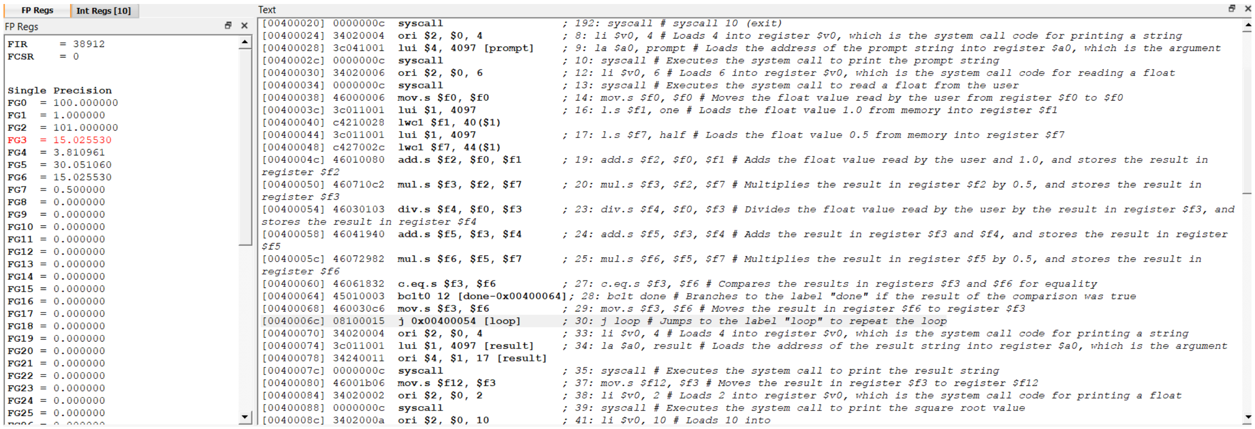
Moved back to loop



```
FIR     = 38912
FCSR    = 0


Single Precision
FG0   = 100.000000
FG1   = 1.000000
FG2   = 101.000000
FG3   = 26.240099
FG4   = 1.980198
FG5   = 52.480198
FG6   = 26.240099
FG7   = 0.500000
FG8   = 0.000000
FG9   = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
FG26 = 0.000000
```

```
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4           ; 8: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400028] 3c041001  lui $4, 4097 [prompt]   ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                 ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6           ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                 ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0          ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097            ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097            ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1     ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7     ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3     ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4     ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7     ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6         ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6          ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
[0040006c] 08100015  j 0x00400054 [loop]     ; 30: j loop # Jumps to the label "loop" to repeat the loop
```

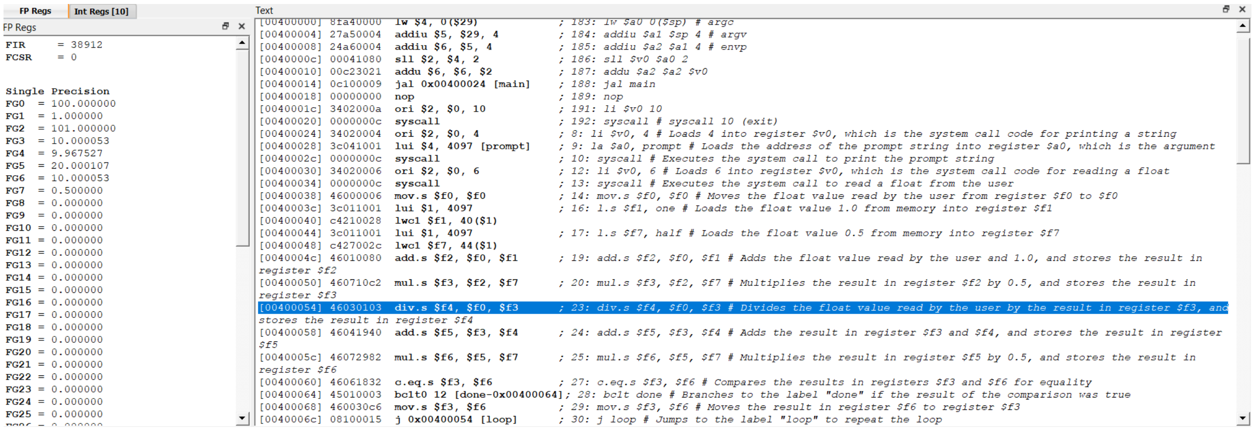
Move back to loop with approximation



```
FIR      = 38912
FCSR     = 0


Single Precision
FG0   = 100.000000
FG1   = 1.000000
FG2   = 101.000000
FG3   = 15.025530
FG4   = 3.810961
FG5   = 30.051060
FG6   = 15.025530
FG7   = 0.500000
FG8   = 0.000000
FG9   = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
```

```
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4             ; 8: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400028] 3c041001  lui $4, 4097 [prompt]     ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                   ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6             ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                   ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0            ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097              ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097              ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1       ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7       ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3       ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4       ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7       ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6           ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6            ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
[0040006c] 08100015  j 0x00400054 [loop]       ; 30: j loop # Jumps to the label "loop" to repeat the loop
[00400070] 34020004  ori $2, $0, 4             ; 33: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400074] 3c011001  lui $1, 4097 [result]     ; 34: la $a0, result # Loads the address of the result string into register $a0, which is the argument
[00400078] 34240011  ori $4, $1, 17 [result]
[0040007c] 0000000c  syscall                   ; 35: syscall # Executes the system call to print the result string
[00400080] 46001b06  mov.s $f12, $f3           ; 37: mov.s $f12, $f3 # Moves the result in register $f3 to register $f12
[00400084] 34020002  ori $2, $0, 2             ; 38: li $v0, 2 # Loads 2 into register $v0, which is the system call code for printing a float
[00400088] 0000000c  syscall                   ; 39: syscall # Executes the system call to print the square root value
[0040008c] 3402000a  ori $2, $0, 10            ; 41: li $v0, 10 # Loads 10 into
```

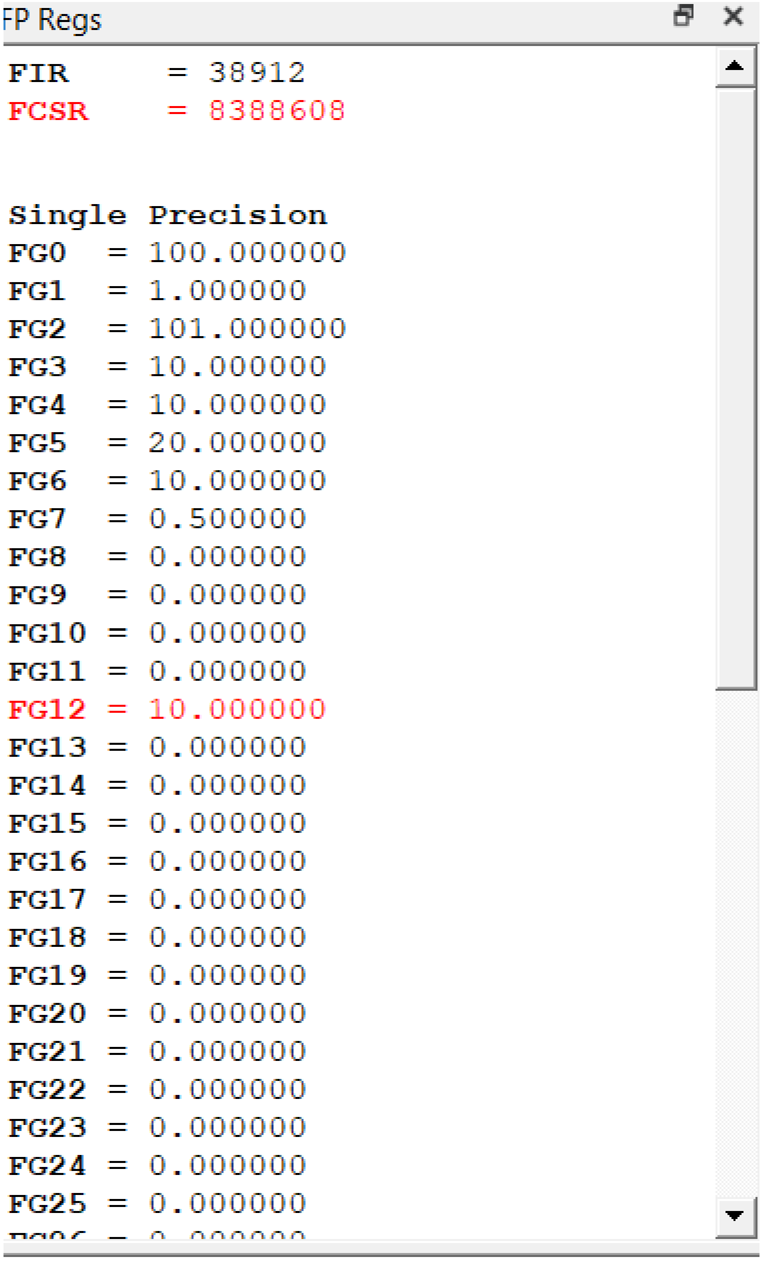
Back to loop with better approximation



```
FIR      = 38912
FCSR     = 0


Single Precision
FG0   = 100.000000
FG1   = 1.000000
FG2   = 101.000000
FG3   = 10.000053
FG4   = 9.967527
FG5   = 20.000107
FG6   = 10.000053
FG7   = 0.500000
FG8   = 0.000000
FG9   = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
```

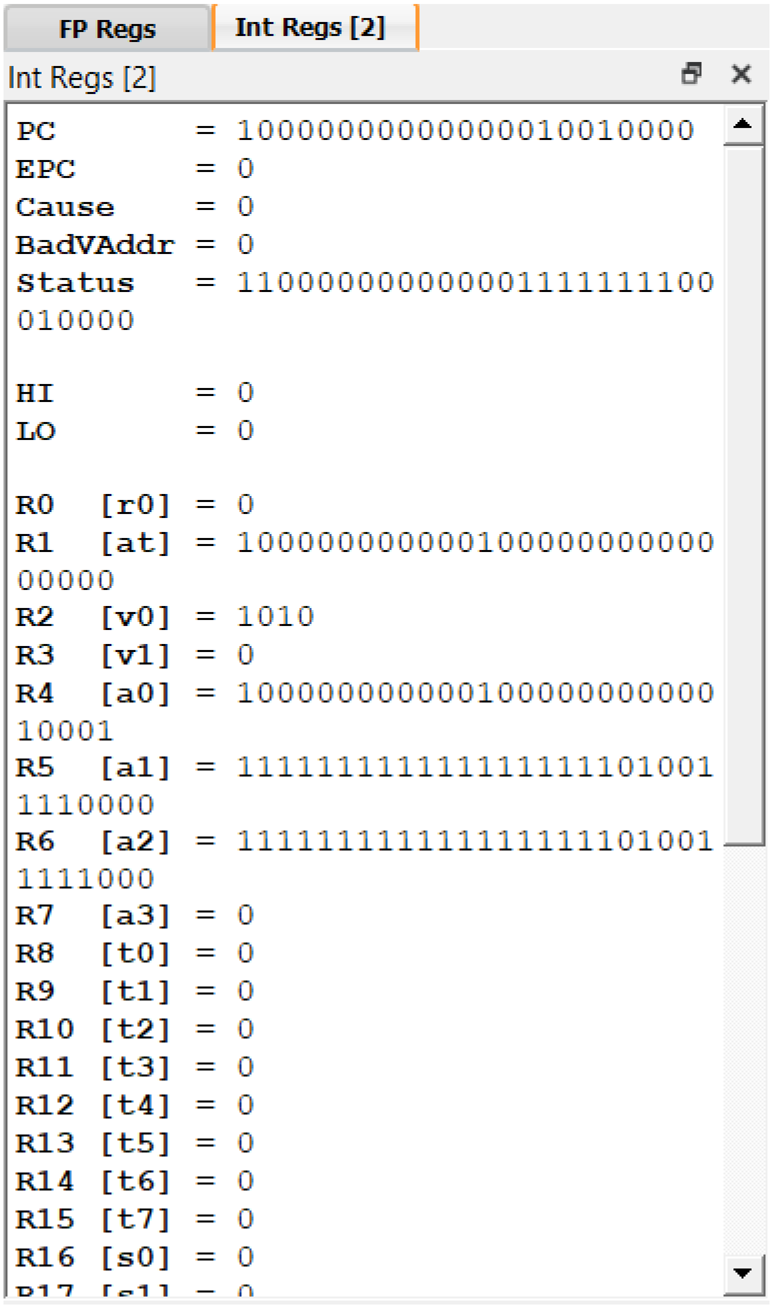
```
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4             ; 8: li $v0, 4 # Loads 4 into register $v0, which is the system call code for printing a string
[00400028] 3c041001  lui $4, 4097 [prompt]     ; 9: la $a0, prompt # Loads the address of the prompt string into register $a0, which is the argument
[0040002c] 0000000c  syscall                   ; 10: syscall # Executes the system call to print the prompt string
[00400030] 34020006  ori $2, $0, 6             ; 12: li $v0, 6 # Loads 6 into register $v0, which is the system call code for reading a float
[00400034] 0000000c  syscall                   ; 13: syscall # Executes the system call to read a float from the user
[00400038] 46000006  mov.s $f0, $f0            ; 14: mov.s $f0, $f0 # Moves the float value read by the user from register $f0 to $f0
[0040003c] 3c011001  lui $1, 4097              ; 16: l.s $f1, one # Loads the float value 1.0 from memory into register $f1
[00400040] c4210028  lwc1 $f1, 40($1)
[00400044] 3c011001  lui $1, 4097              ; 17: l.s $f7, half # Loads the float value 0.5 from memory into register $f7
[00400048] c427002c  lwc1 $f7, 44($1)
[0040004c] 46010080  add.s $f2, $f0, $f1       ; 19: add.s $f2, $f0, $f1 # Adds the float value read by the user and 1.0, and stores the result in
register $f2
[00400050] 460710c2  mul.s $f3, $f2, $f7       ; 20: mul.s $f3, $f2, $f7 # Multiplies the result in register $f2 by 0.5, and stores the result in
register $f3
[00400054] 46030103  div.s $f4, $f0, $f3       ; 23: div.s $f4, $f0, $f3 # Divides the float value read by the user by the result in register $f3, and
stores the result in register $f4
[00400058] 46041940  add.s $f5, $f3, $f4       ; 24: add.s $f5, $f3, $f4 # Adds the result in register $f3 and $f4, and stores the result in register
$f5
[0040005c] 46072982  mul.s $f6, $f5, $f7       ; 25: mul.s $f6, $f5, $f7 # Multiplies the result in register $f5 by 0.5, and stores the result in
register $f6
[00400060] 46061832  c.eq.s $f3, $f6           ; 27: c.eq.s $f3, $f6 # Compares the results in registers $f3 and $f6 for equality
[00400064] 45010003  bc1t0 12 [done-0x00400064]; 28: bc1t done # Branches to the label "done" if the result of the comparison was true
[00400068] 460030c6  mov.s $f3, $f6            ; 29: mov.s $f3, $f6 # Moves the result in register $f6 to register $f3
[0040006c] 08100015  j 0x00400054 [loop]       ; 30: j loop # Jumps to the label "loop" to repeat the loop
```

Registers after execution of the code



```
FIR      = 38912
FCSR     = 8388608


Single Precision
FG0  = 100.000000
FG1  = 1.000000
FG2  = 101.000000
FG3  = 10.000000
FG4  = 10.000000
FG5  = 20.000000
FG6  = 10.000000
FG7  = 0.500000
FG8  = 0.000000
FG9  = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 10.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
FG26 = 0.000000
```

Registers after execution of the code



```
PC       = 10000000000000010010000
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 110000000000001111111100
010000

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 100000000000100000000000
00000
R2  [v0] = 1010
R3  [v1] = 0
R4  [a0] = 100000000000100000000000
10001
R5  [a1] = 111111111111111111101001
1110000
R6  [a2] = 111111111111111111101001
1111000
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
```