San Francisco Bay University

EE488 - Computer Architecture
Homework Assignment #3

Due day:
3/20/2025

Instruction:

1. Implement a program (MIPS Assembly) which multiplies user input by 10 using only bit shift operations and addition. Check to see if your program is correct by using the *mult* and *mflo* operators. Your program should include a proper and useful prompt for input, and print the results meaningfully.

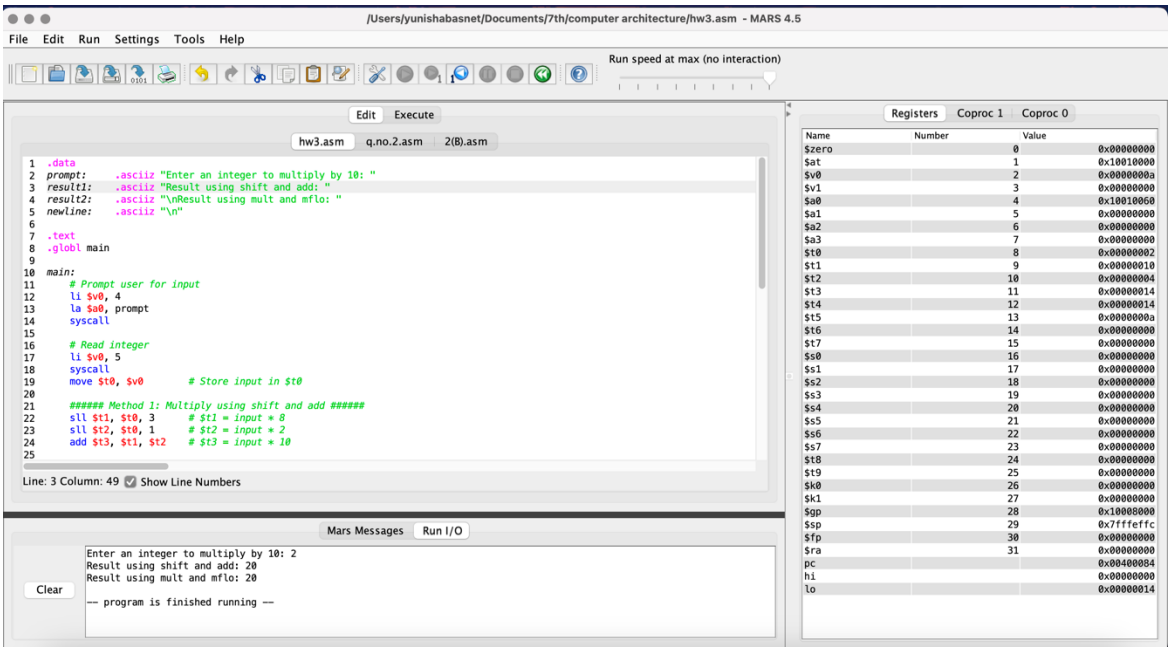
⇨ Answer:

⇨ The code:

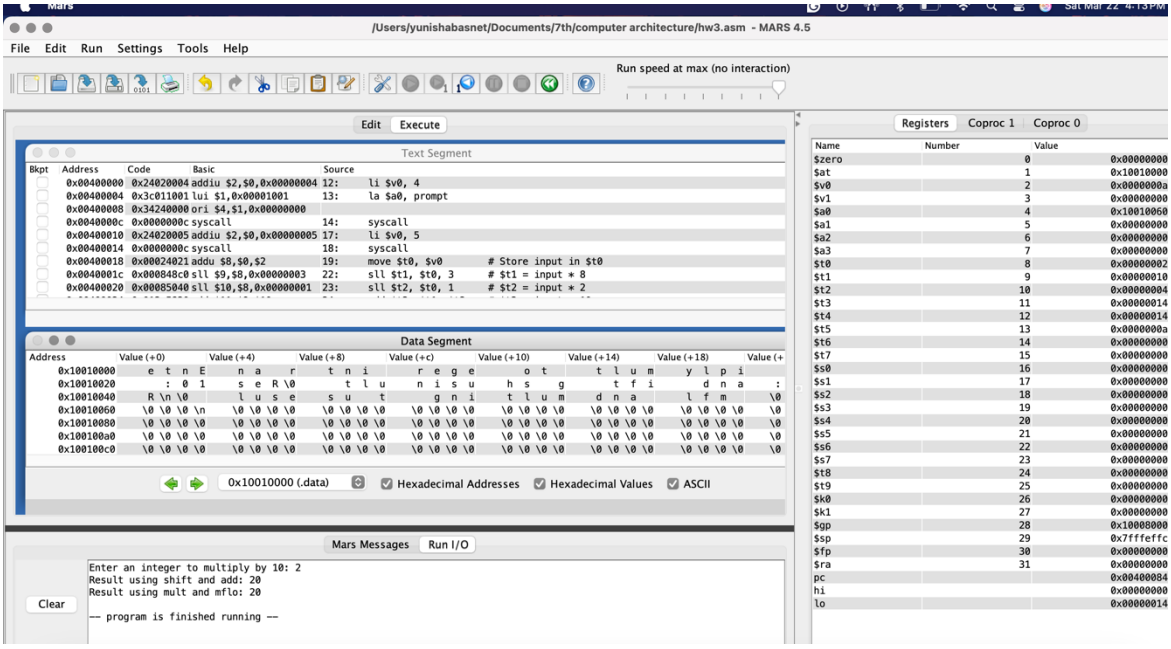
```
.data
prompt:       .asciiz "Enter an integer to multiply by 10: "
result1:      .asciiz "Result using shift and add: "
result2:      .asciiz "\nResult using mult and mflo: "
newline:      .asciiz "\n"

.text
.globl main

main:
    # Prompt user for input
    li $v0, 4
    la $a0, prompt
```

```
    syscall

    # Read integer
    li $v0, 5
    syscall
    move $t0, $v0         # Store input
in $t0

    ######  Method  1:  Multiply  using
shift and add ######
    sll $t1, $t0, 3       # $t1 = input
* 8
    sll $t2, $t0, 1       # $t2 = input
* 2
    add $t3, $t1, $t2     # $t3 = input
* 10

    # Print result1
    li $v0, 4
    la $a0, result1
    syscall

    li $v0, 1
    move $a0, $t3
    syscall

    ######  Method  2:  Multiply  using
mult and mflo ######
    li $t5, 10           # Load 10 into
register
    mult  $t0,  $t5           # Multiply
input * 10
    mflo $t4             # Store result
in $t4

    # Print result2
    li $v0, 4
    la $a0, result2
    syscall
```

```
li $v0, 1
move $a0, $t4
syscall

# Print newline
li $v0, 4
la $a0, newline
syscall

# Exit
li $v0, 10
syscall
```



⇨ The output:



2. Write programs (MIPS Assembly) to evaluate the following expressions. The user should enter the variables, and the program should print back an answer. Prompt the user for all variables in the expression, and print the results in a meaningful manner. **The results should be as accurate as possible.**

a. $5x + 3y + z$

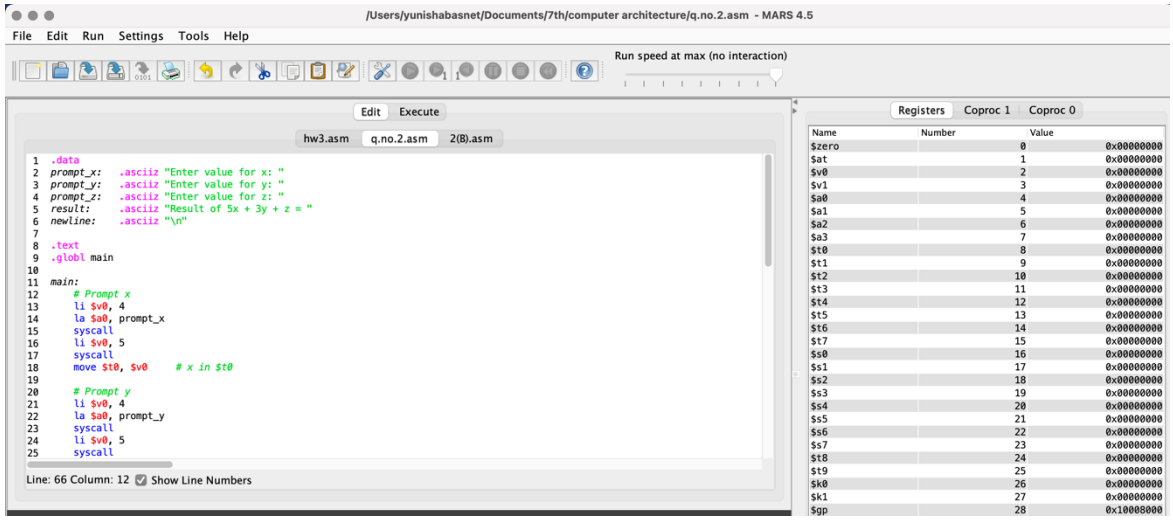
⇨ The answer:

⇨ The code:

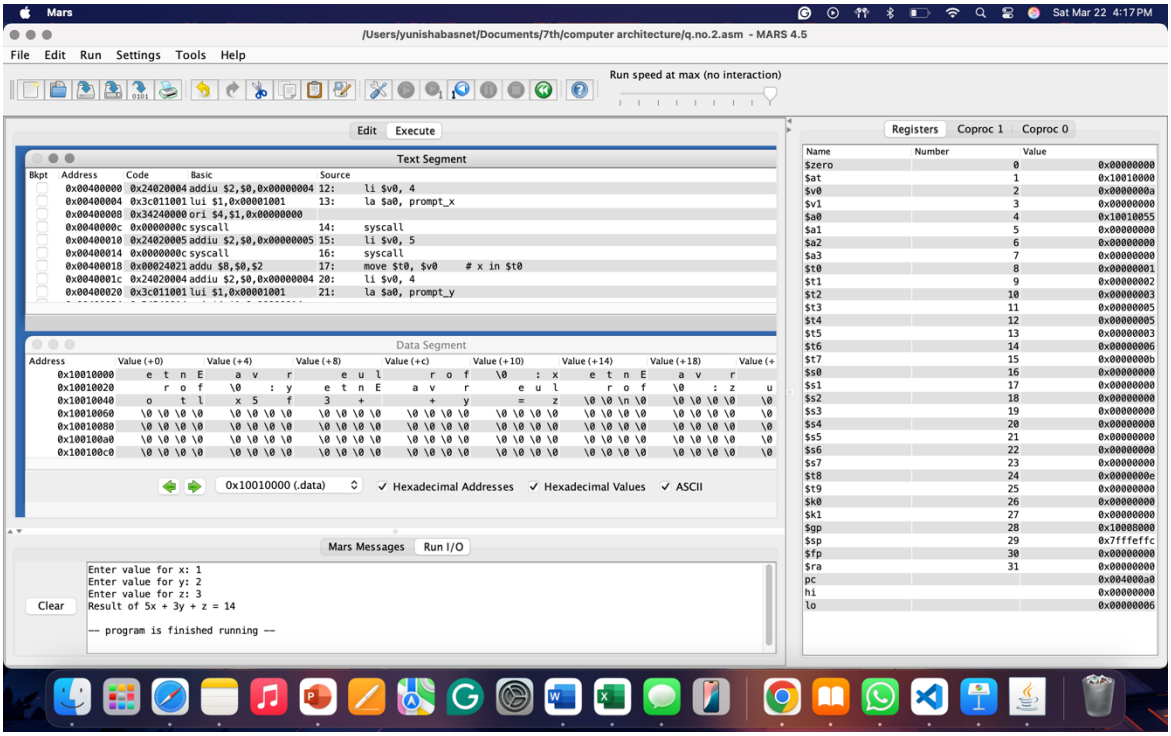
```
.data
prompt_x:    .asciiz "Enter value for x:
"
prompt_y:    .asciiz "Enter value for y:
"
prompt_z:    .asciiz "Enter value for z:
"
result:       .asciiz "Result of 5x + 3y
+ z = "
newline:     .asciiz "\n"

.text
```

```
.globl main
main:
    # Prompt x
    li $v0, 4
    la $a0, prompt_x
    syscall
    li $v0, 5
    syscall
    move $t0, $v0     # x in $t0

    # Prompt y
    li $v0, 4
    la $a0, prompt_y
    syscall
    li $v0, 5
    syscall
    move $t1, $v0     # y in $t1

    # Prompt z
    li $v0, 4
    la $a0, prompt_z
    syscall
    li $v0, 5
    syscall
    move $t2, $v0     # z in $t2

    # Compute 5x → 5 * $t0
    li $t3, 5
    mul $t4, $t3, $t0  # $t4 = 5x

    # Compute 3y → 3 * $t1
    li $t5, 3
    mul $t6, $t5, $t1  # $t6 = 3y

    # Add: 5x + 3y
    add $t7, $t4, $t6

    # Add z: 5x + 3y + z
```

```
    add $t8, $t7, $t2

    # Print result
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $t8
    syscall

    # Print newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```



⇨ The output:



b. $\left(\frac{5x+3y+z}{2}\right) * 3$

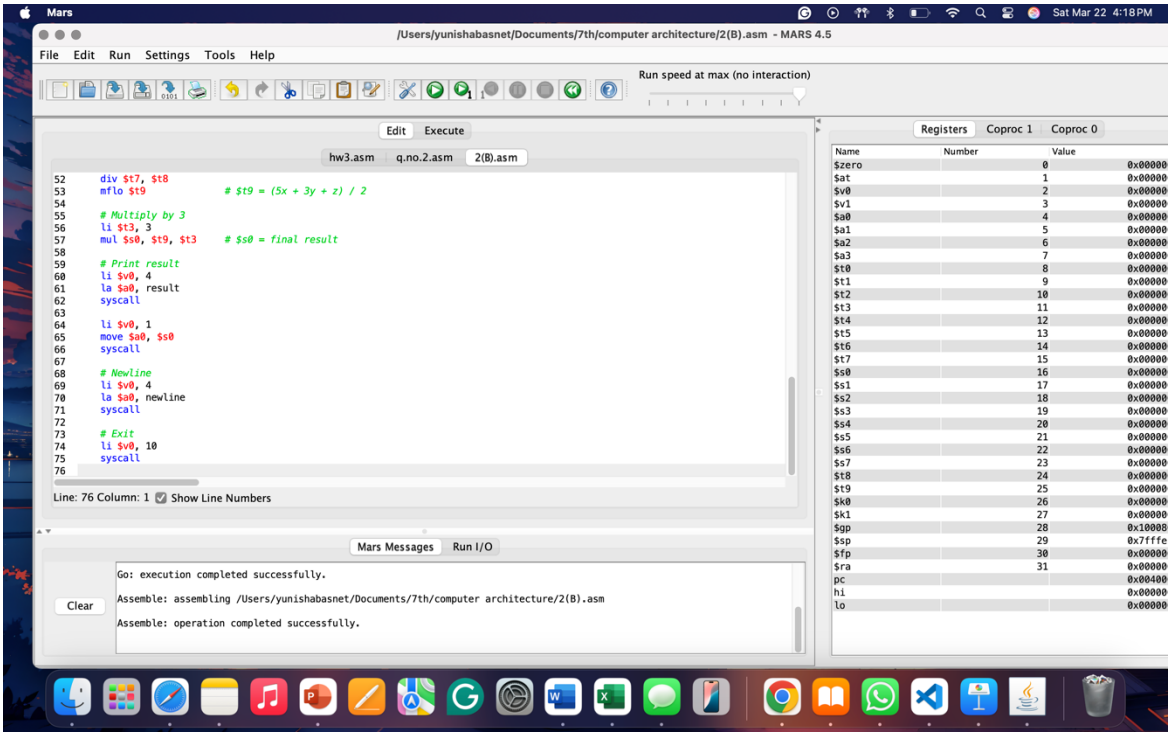
⇨ The answer:

⇨ The code:

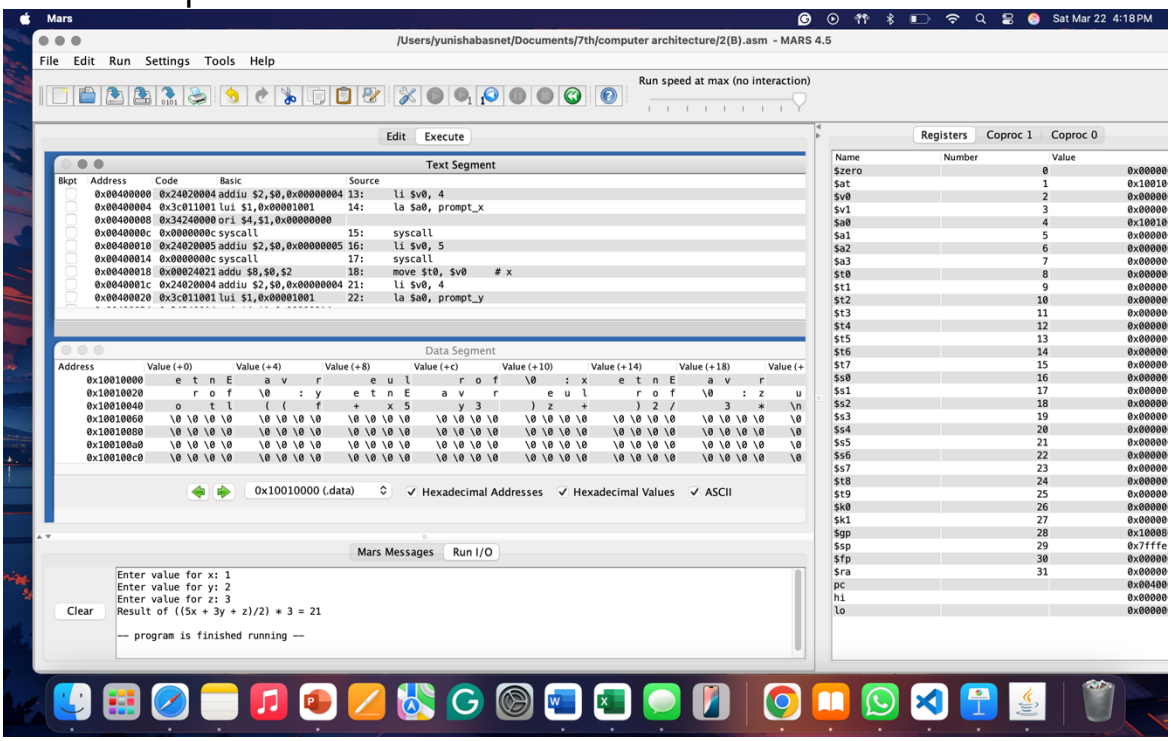
```
.data
prompt_x:    .asciiz "Enter value for x:
"
prompt_y:    .asciiz "Enter value for y:
"
prompt_z:    .asciiz "Enter value for z:
"
result:       .asciiz "Result of ((5x +
3y + z)/2) * 3 = "
newline:     .asciiz "\n"

.text
.globl main

main:
    # Prompt x
    li $v0, 4
    la $a0, prompt_x
    syscall
```

```
    li $v0, 5
    syscall
    move $t0, $v0     # x

    # Prompt y
    li $v0, 4
    la $a0, prompt_y
    syscall
    li $v0, 5
    syscall
    move $t1, $v0     # y

    # Prompt z
    li $v0, 4
    la $a0, prompt_z
    syscall
    li $v0, 5
    syscall
    move $t2, $v0     # z

    # 5x
    li $t3, 5
    mul $t4, $t3, $t0     # $t4 = 5x

    # 3y
    li $t5, 3
    mul $t6, $t5, $t1     # $t6 = 3y

    # sum = 5x + 3y
    add $t7, $t4, $t6

    # sum = sum + z
    add $t7, $t7, $t2      # $t7 = 5x +
3y + z

    # Divide by 2
    li $t8, 2
    div $t7, $t8
```

```
    mflo $t9                # $t9 = (5x + 3y + z) / 2

    # Multiply by 3
    li $t3, 3
    mul $s0, $t9, $t3       # $s0 = final result

    # Print result
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $s0
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```



⇨ The output:



c. $x^3 + 2x^2 + 3x + 4$

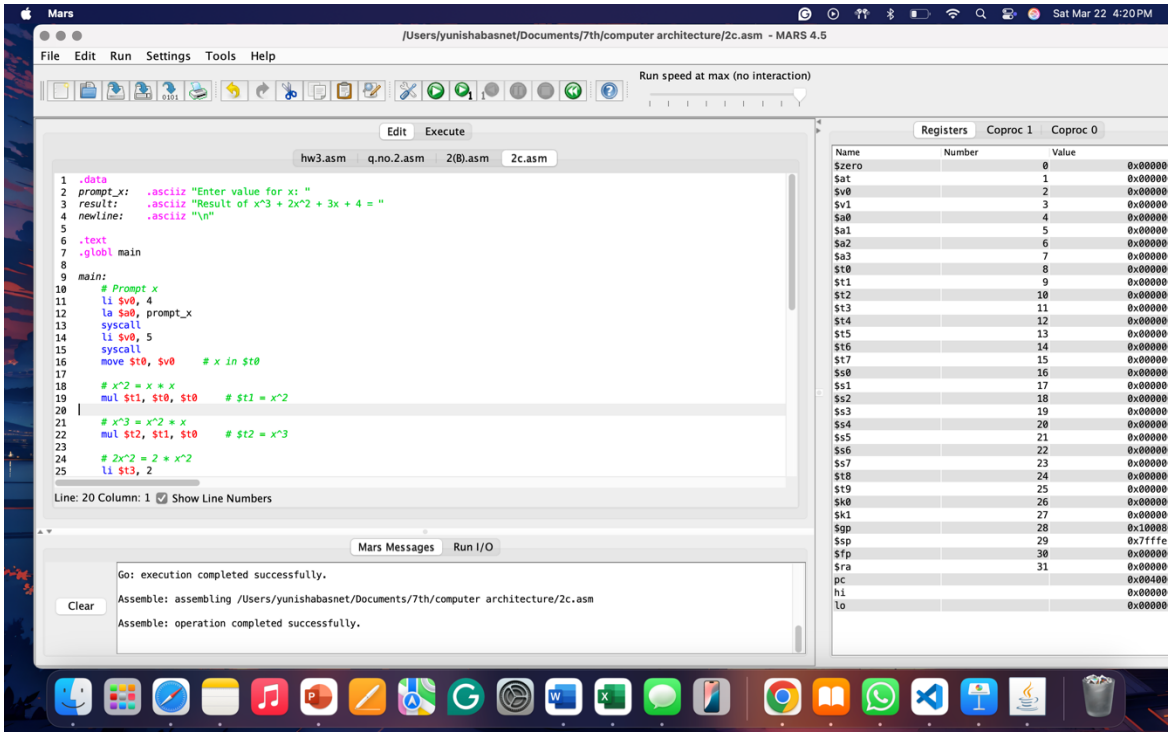
⇨ The answer:

⇨ The code:

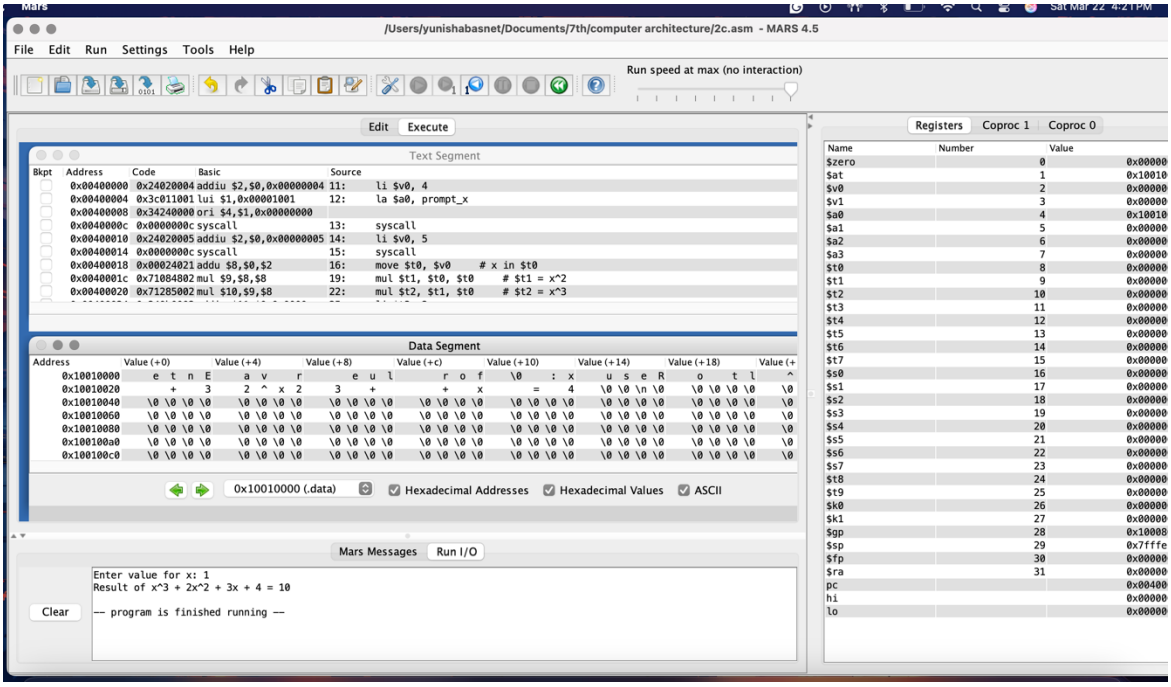
```
.data
```

```
prompt_x:   .asciiz "Enter value for x:
"
result:       .asciiz "Result of x^3 +
2x^2 + 3x + 4 = "
newline:    .asciiz "\n"

.text
.globl main

main:
    # Prompt x
    li $v0, 4
    la $a0, prompt_x
    syscall
    li $v0, 5
    syscall
    move $t0, $v0     # x in $t0

    # x^2 = x * x
    mul $t1, $t0, $t0     # $t1 = x^2

    # x^3 = x^2 * x
    mul $t2, $t1, $t0     # $t2 = x^3

    # 2x^2 = 2 * x^2
    li $t3, 2
    mul $t4, $t3, $t1     # $t4 = 2x^2

    # 3x = 3 * x
    li $t5, 3
    mul $t6, $t5, $t0     # $t6 = 3x

    # Add all: x^3 + 2x^2 + 3x + 4
    add $t7, $t2, $t4     # x^3 + 2x^2
    add $t7, $t7, $t6     # + 3x
    li $t8, 4
    add $t9, $t7, $t8     # + 4 → final
result
```

```
# Print result
li $v0, 4
la $a0, result
syscall

li $v0, 1
move $a0, $t9
syscall

# Newline
li $v0, 4
la $a0, newline
syscall

# Exit
li $v0, 10
syscall
```



⇨ The output:



d. $\left(\frac{4x}{3}\right) * y$

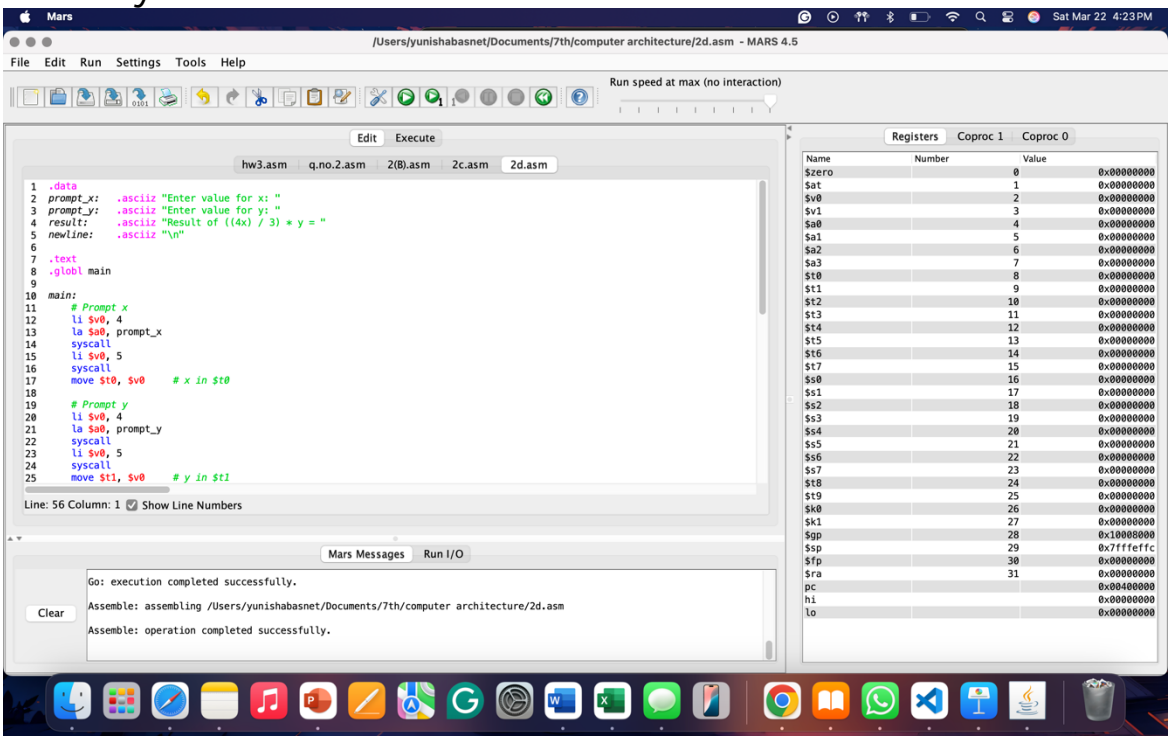
⇨ The answer:

⇨ The code:

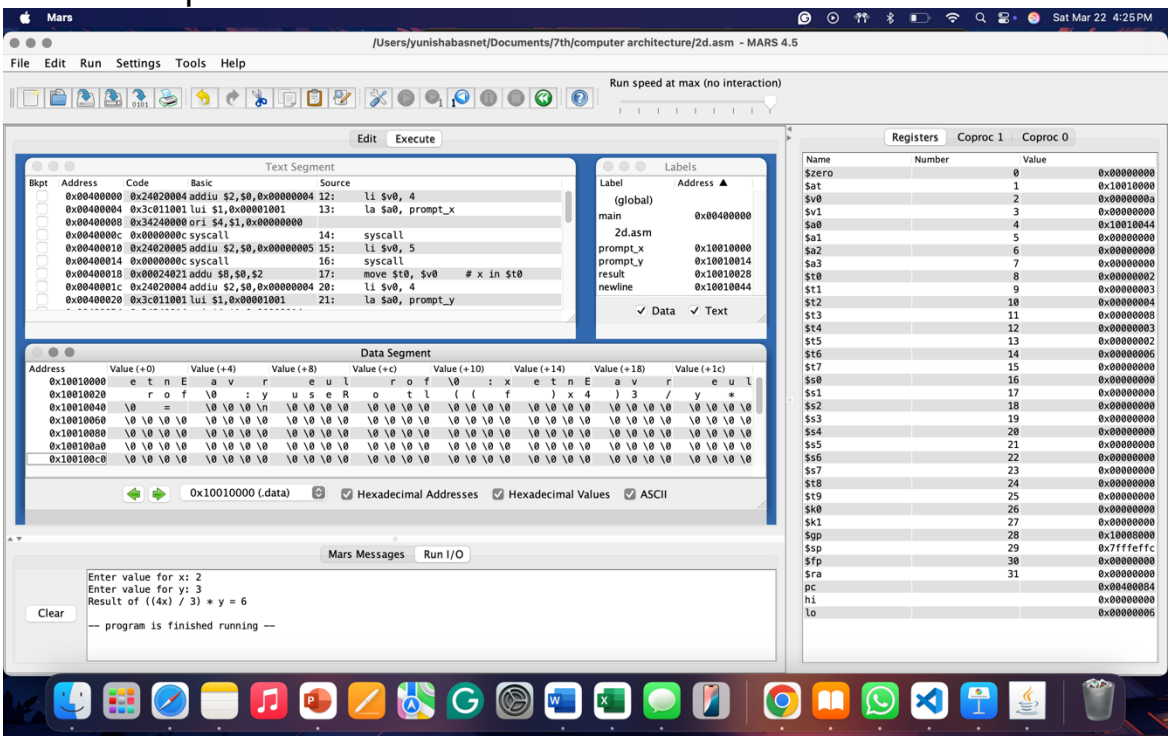
```
.data
prompt_x:    .asciiz "Enter value for x:
"
prompt_y:    .asciiz "Enter value for y:
"
result:       .asciiz "Result of ((4x) /
3) * y = "
newline:     .asciiz "\n"

.text
.globl main

main:
    # Prompt x
    li $v0, 4
    la $a0, prompt_x
    syscall
    li $v0, 5
    syscall
    move $t0, $v0      # x in $t0
```

```
    # Prompt y
    li $v0, 4
    la $a0, prompt_y
    syscall
    li $v0, 5
    syscall
    move $t1, $v0     # y in $t1

    # 4x
    li $t2, 4
    mul $t3, $t2, $t0     # $t3 = 4x

    # (4x) / 3
    li $t4, 3
    div $t3, $t4
    mflo $t5                # $t5 = (4x)
/ 3

    # * y
    mul $t6, $t5, $t1      # $t6 = ((4x)
/ 3) * y

    # Print result
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $t6
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
```

syscall



⇨ The output:



3. Write a program (MIPS Assembly) to retrieve two numbers from a user and swap those numbers using only the *XOR* operation. You should not use a temporary

variable to store the numbers while swapping them. Your program should include a proper and useful prompt for input, and print the results meaningfully.

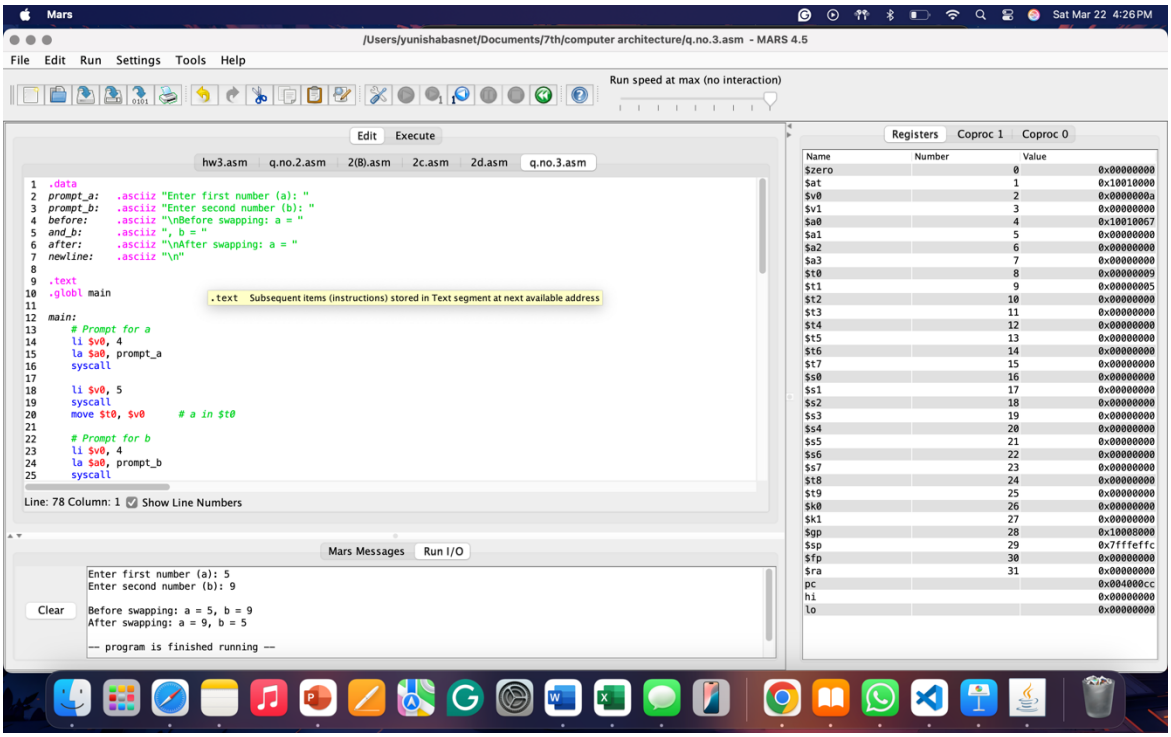
⇨ The answer:

⇨ The code:

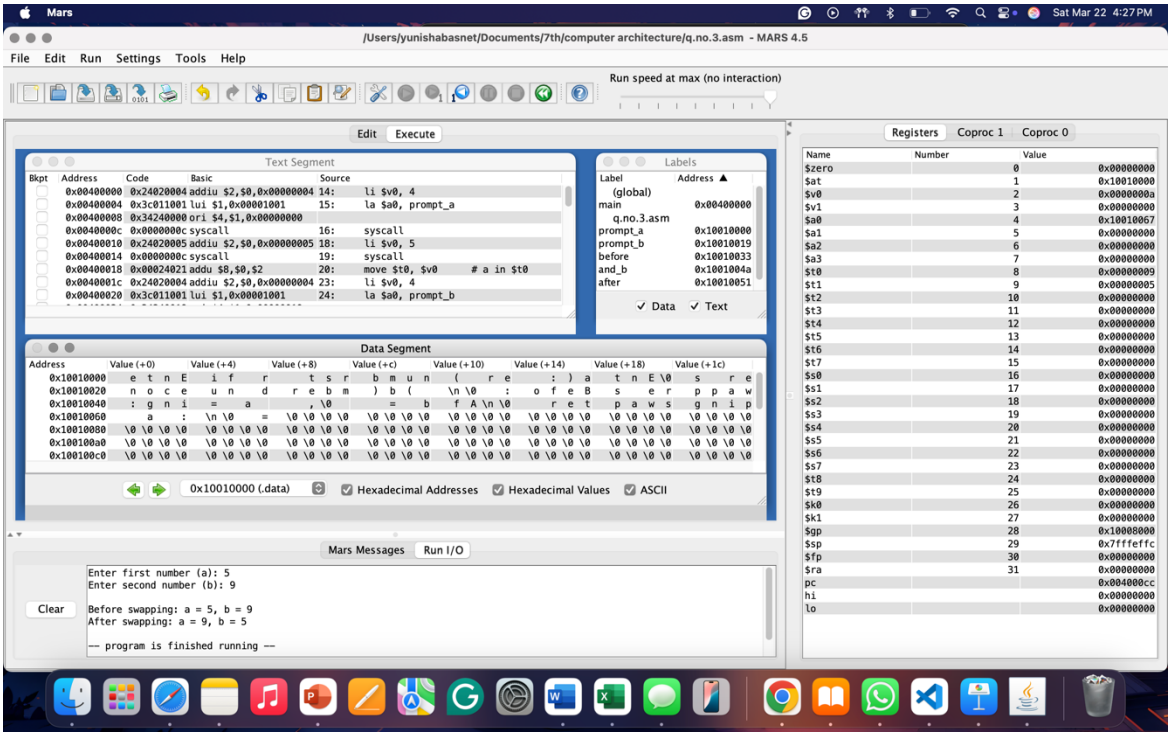
```
  .data
  prompt_a:    .asciiz "Enter first number
  (a): "
  prompt_b:        .asciiz  "Enter  second
  number (b): "
  before:      .asciiz "\nBefore swapping:
  a = "
  and_b:       .asciiz ", b = "
  after:        .asciiz "\nAfter swapping:
  a = "
  newline:     .asciiz "\n"

  .text
  .globl main

  main:
      # Prompt for a
      li $v0, 4
      la $a0, prompt_a
      syscall

      li $v0, 5
      syscall
      move $t0, $v0      # a in $t0

      # Prompt for b
      li $v0, 4
      la $a0, prompt_b
      syscall

      li $v0, 5
      syscall
      move $t1, $v0      # b in $t1
```

```
    ###### Print BEFORE swap ######
    li $v0, 4
    la $a0, before
    syscall

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 4
    la $a0, and_b
    syscall

    li $v0, 1
    move $a0, $t1
    syscall

    ###### XOR Swap ######
    xor $t0, $t0, $t1   # a = a ^ b
    xor $t1, $t0, $t1    # b = a ^ b → original a
    xor $t0, $t0, $t1    # a = a ^ b → original b

    ###### Print AFTER swap ######
    li $v0, 4
    la $a0, after
    syscall

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 4
    la $a0, and_b
    syscall
```

```
    li $v0, 1
    move $a0, $t1
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```



⇨ The output:



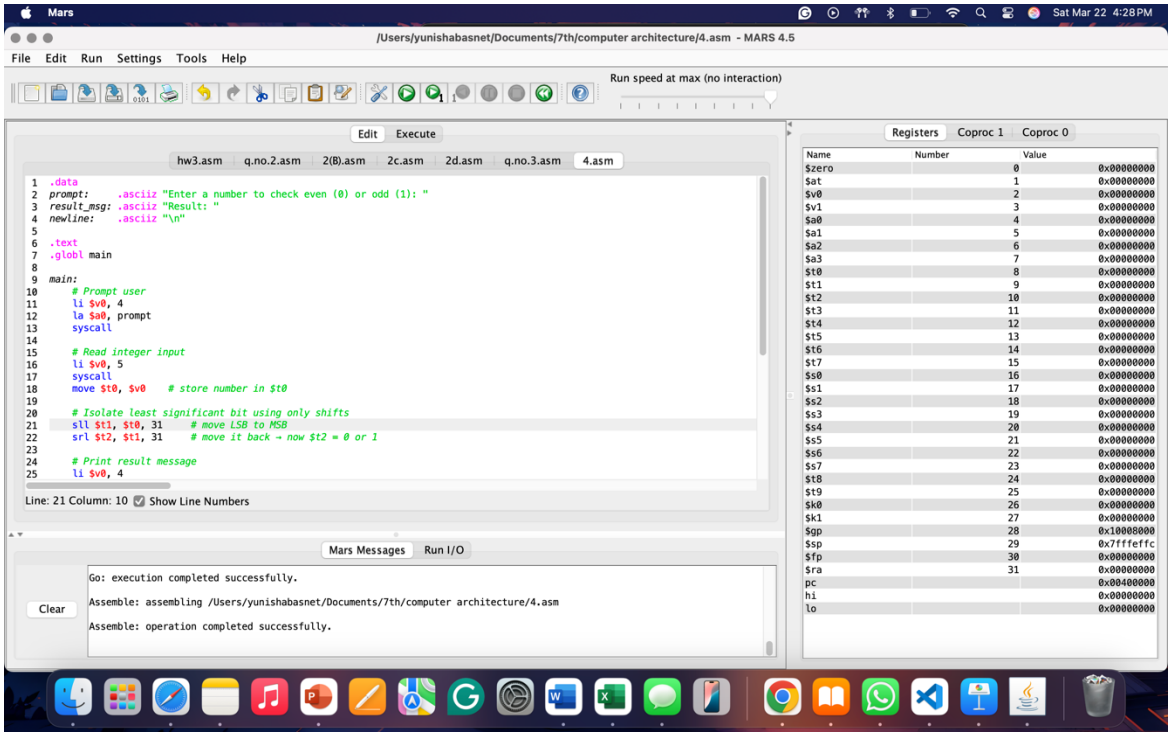
4. Using only *sll* and *srl*, implement a program to check if a user input value is even or odd. The result should print out *0* if the number is even or *1* if the number is odd. Your program (MIPS Assembly) should include a proper and useful prompt for input, and print the results in a meaningful manner.

⇨ The answer:

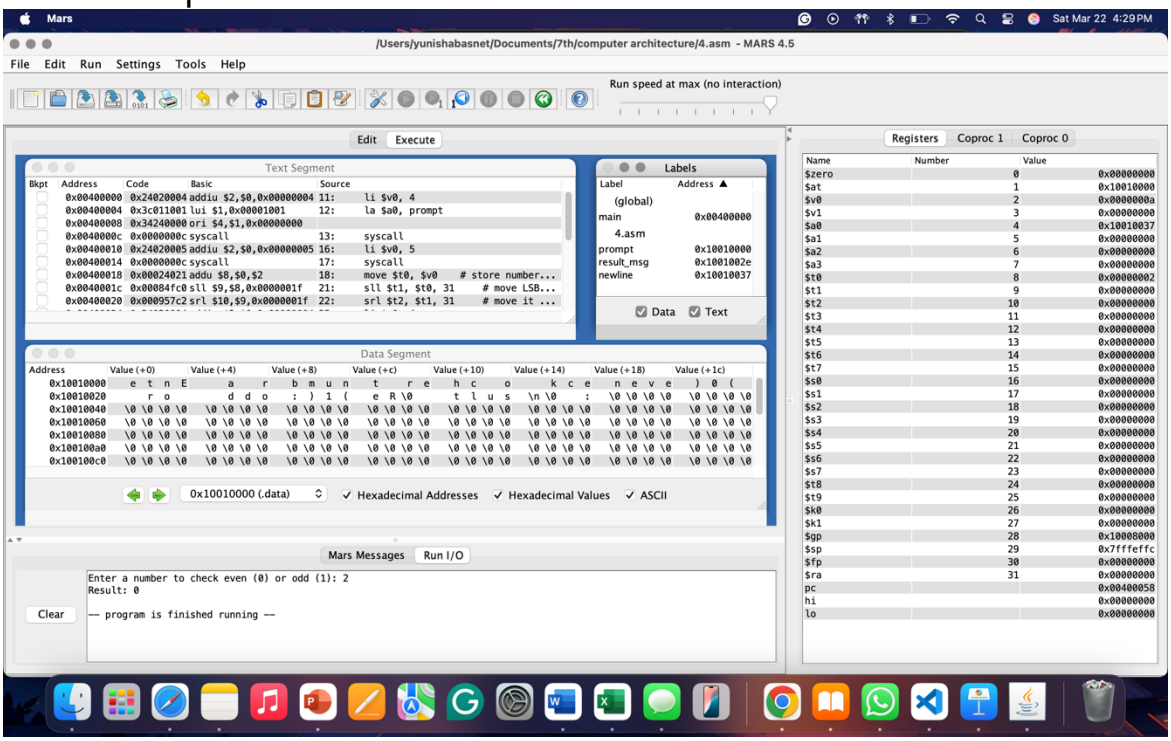
⇨ The code:

```
.data
prompt:      .asciiz "Enter a number to
check even (0) or odd (1): "
result_msg: .asciiz "Result: "
newline:    .asciiz "\n"

.text
.globl main

main:
    # Prompt user
```

```
    li $v0, 4
    la $a0, prompt
    syscall

    # Read integer input
    li $v0, 5
    syscall
    move $t0, $v0     # store number in
$t0

    #  Isolate  least  significant  bit
using only shifts
    sll $t1, $t0, 31      # move LSB to
MSB
    srl $t2, $t1, 31      # move it back
→ now $t2 = 0 or 1

    # Print result message
    li $v0, 4
    la $a0, result_msg
    syscall

    # Print 0 (even) or 1 (odd)
    li $v0, 1
    move $a0, $t2
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```



⇨ The output:



5. Implement a program (MIPS Assembly) to prompt the user for two numbers, the first being any number and the second a prime number. Return to the user a *0* if the second number is a prime factor for

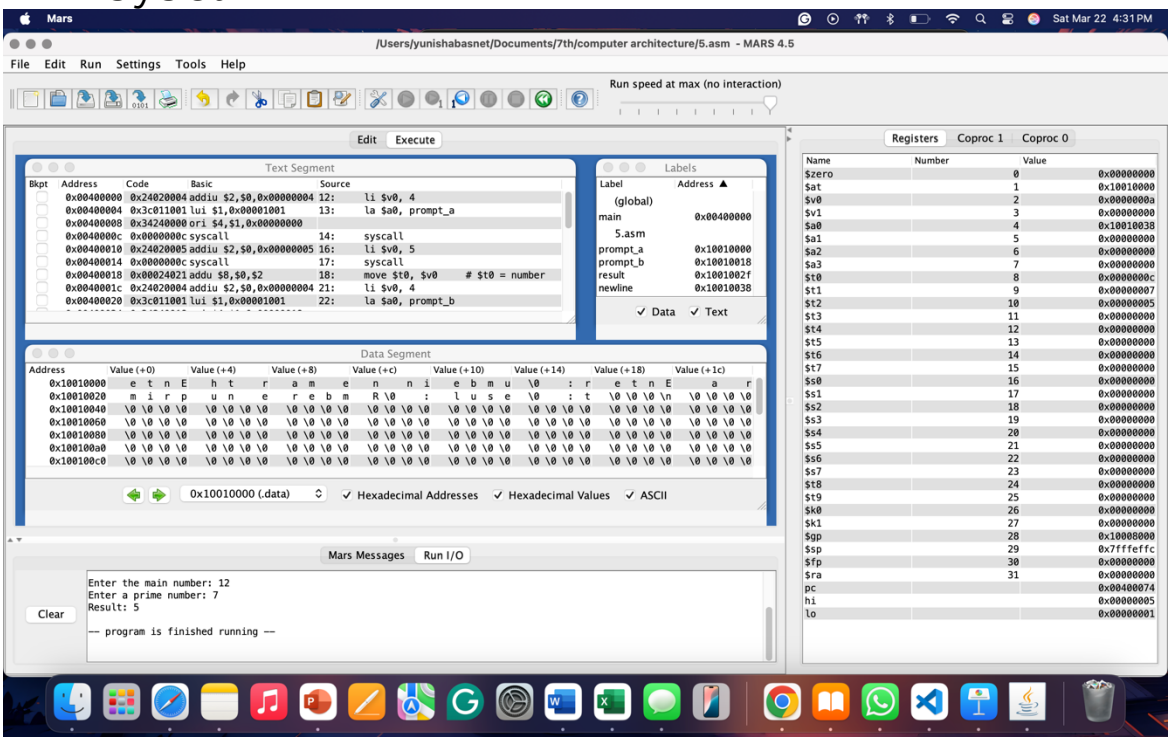
the first one, otherwise any number if it is not. For example, if the user enters *60* and *5*, the program returns *0*. If the user enters *62* and *5*, the program returns *2*.

⇨ The answer:

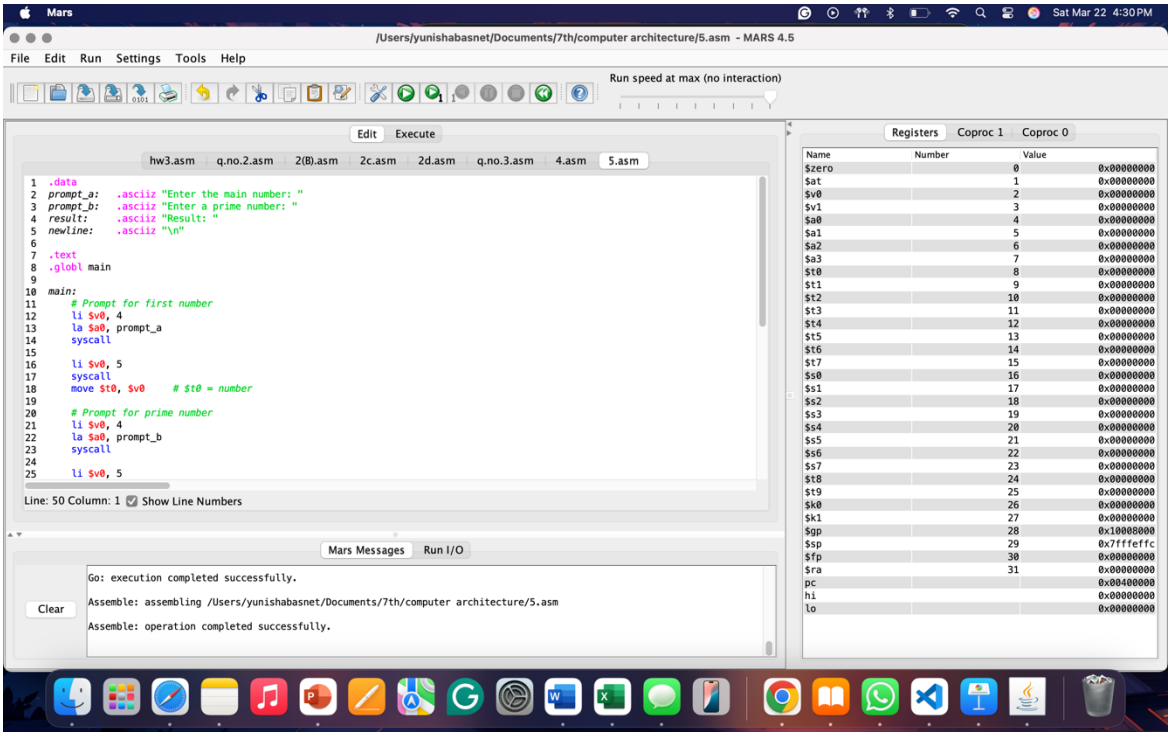
⇨ The code:

```
  .data
  prompt_a:      .asciiz  "Enter  the  main
  number: "
  prompt_b:       .asciiz  "Enter  a  prime
  number: "
  result:      .asciiz "Result: "
  newline:     .asciiz "\n"

  .text
  .globl main

  main:
      # Prompt for first number
      li $v0, 4
      la $a0, prompt_a
      syscall

      li $v0, 5
      syscall
      move $t0, $v0      # $t0 = number

      # Prompt for prime number
      li $v0, 4
      la $a0, prompt_b
      syscall

      li $v0, 5
      syscall
      move $t1, $v0      # $t1 = supposed
  prime

      # Divide number by prime
```

```
    div $t0, $t1
    mfhi $t2                # remainder in
$t2

    # Print result
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $t2       # if remainder is
0 → it's a factor
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```



⇨ The output:


Mars
Sat Mar 22 4:30 PM
/Users/yunishabasnet/Documents/7th/computer architecture/5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
hw3.asm q.no.2.asm 2(B).asm 2c.asm 2d.asm q.no.3.asm 4.asm 5.asm
1 .data
2 prompt_a: .asciiz "Enter the main number: "
3 prompt_b: .asciiz "Enter a prime number: "
4 result: .asciiz "Result: "
5 newline: .asciiz "\n"
6
7 .text
8 .globl main
9
10 main:
11 # Prompt for first number
12 li $v0, 4
13 la $a0, prompt_a
14 syscall
15
16 li $v0, 5
17 syscall
18 move $t0, $v0 # $t0 = number
19
20 # Prompt for prime number
21 li $v0, 4
22 la $a0, prompt_b
23 syscall
24
25 li $v0, 5
Line: 50 Column: 1 Show Line Numbers
Mars Messages Run I/O
Clear
Go: execution completed successfully.
Assemble: assembling /Users/yunishabasnet/Documents/7th/computer architecture/5.asm
Assemble: operation completed successfully.
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
$v1 3 0x00000000
$a0 4 0x00000000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000