**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #3**

**Due day: 7/2/2024**

**Instruction:**

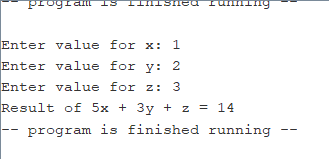
1. **Push the answer sheet to GitHub in word file**
2. **Overdue homework submission could not be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
4. Implement a program (MIPS Assembly) which multiplies user input by 10 using only bit shift operations 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 in a meaningful manner.

|  |
| --- |
| .data  prompt: .asciiz "Please enter a number to multiply by 10: "  resultStr: .asciiz "The result using bit shift and addition is: "  checkStr: .asciiz "\nThe result using mult and mflo operators is: "  .text  .globl main  main:  # Print the input prompt  li $v0, 4  la $a0, prompt  syscall  # Read user input  li $v0, 5  syscall  move $t0, $v0 # Store the input in $t0  # Multiply input by 10 using bit shift operations and addition  # The operation performed here is: result = input \* 8 + input \* 2  sll $t1, $t0, 3 # $t1 = input \* 8  sll $t2, $t0, 1 # $t2 = input \* 2  add $t3, $t1, $t2 # $t3 = input \* 10  # Print the result string  li $v0, 4  la $a0, resultStr  syscall  # Print the result  move $a0, $t3  li $v0, 1  syscall  # Verify the result using mult and mflo operators  li $t4, 10  mult $t0, $t4  mflo $t5  # Print the check string  li $v0, 4  la $a0, checkStr  syscall  # Print the checked result  move $a0, $t5  li $v0, 1  syscall  # Exit the program  li $v0, 10  syscall |

1. 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.**

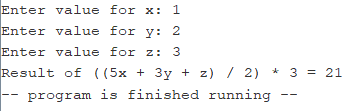
（1）For expression a: 5x + 3y + z

|  |
| --- |
| .data  prompt\_x: .asciiz "Enter value for x: "  prompt\_y: .asciiz "Enter value for y: "  prompt\_z: .asciiz "Enter value for z: "  result\_msg: .asciiz "Result of 5x + 3y + z = "  .text  .globl main  main:  # Prompt and read x  li $v0, 4  la $a0, prompt\_x  syscall  li $v0, 5  syscall  move $t0, $v0 # $t0 = x  # Prompt and read y  li $v0, 4  la $a0, prompt\_y  syscall  li $v0, 5  syscall  move $t1, $v0 # $t1 = y  # Prompt and read z  li $v0, 4  la $a0, prompt\_z  syscall  li $v0, 5  syscall  move $t2, $v0 # $t2 = z  # Calculate 5x  li $t3, 5  mul $t3, $t3, $t0 # $t3 = 5x  # Calculate 3y  li $t4, 3  mul $t4, $t4, $t1 # $t4 = 3y  # Sum up 5x + 3y + z  add $t5, $t3, $t4  add $t5, $t5, $t2 # $t5 = 5x + 3y + z  # Print result  li $v0, 4  la $a0, result\_msg  syscall  li $v0, 1  move $a0, $t5  syscall  # Exit  li $v0, 10  syscall |



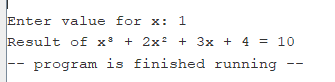
（2）For expression b: ((5x + 3y + z) / 2) \* 3

|  |
| --- |
| .data  prompt\_x: .asciiz "Enter value for x: "  prompt\_y: .asciiz "Enter value for y: "  prompt\_z: .asciiz "Enter value for z: "  result\_msg: .asciiz "Result of ((5x + 3y + z) / 2) \* 3 = "  .text  .globl main  main:  # Prompt and read x  li $v0, 4  la $a0, prompt\_x  syscall  li $v0, 5  syscall  move $t0, $v0 # $t0 = x  # Prompt and read y  li $v0, 4  la $a0, prompt\_y  syscall  li $v0, 5  syscall  move $t1, $v0 # $t1 = y  # Prompt and read z  li $v0, 4  la $a0, prompt\_z  syscall  li $v0, 5  syscall  move $t2, $v0 # $t2 = z  # Calculate 5x  li $t3, 5  mul $t3, $t3, $t0 # $t3 = 5x  # Calculate 3y  li $t4, 3  mul $t4, $t4, $t1 # $t4 = 3y  # Sum up 5x + 3y + z  add $t5, $t3, $t4  add $t5, $t5, $t2 # $t5 = 5x + 3y + z  # Divide by 2  li $t6, 2  div $t5, $t6  mflo $t5 # $t5 = (5x + 3y + z) / 2  # Multiply by 3  li $t6, 3  mul $t5, $t5, $t6 # $t5 = ((5x + 3y + z) / 2) \* 3  # Print result  li $v0, 4  la $a0, result\_msg  syscall  li $v0, 1  move $a0, $t5  syscall  # Exit  li $v0, 10  syscall |



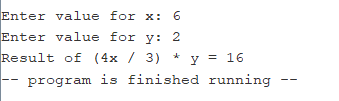
(3) For expression c: x³ + 2x² + 3x + 4

|  |
| --- |
| .data  prompt\_x: .asciiz "Enter value for x: "  result\_msg: .asciiz "Result of x³ + 2x² + 3x + 4 = "  .text  .globl main  main:  # Prompt and read x  li $v0, 4  la $a0, prompt\_x  syscall  li $v0, 5  syscall  move $t0, $v0 # $t0 = x  # Calculate x²  mul $t1, $t0, $t0 # $t1 = x²  # Calculate x³  mul $t2, $t1, $t0 # $t2 = x³  # Calculate 2x²  li $t3, 2  mul $t3, $t3, $t1 # $t3 = 2x²  # Calculate 3x  li $t4, 3  mul $t4, $t4, $t0 # $t4 = 3x  # Sum up x³ + 2x² + 3x + 4  add $t5, $t2, $t3  add $t5, $t5, $t4  addi $t5, $t5, 4 # $t5 = x³ + 2x² + 3x + 4  # Print result  li $v0, 4  la $a0, result\_msg  syscall  li $v0, 1  move $a0, $t5  syscall  # Exit  li $v0, 10  syscall |



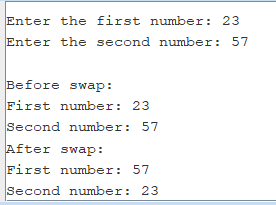
(4) (4x / 3) \* y

|  |
| --- |
| .data  prompt\_x: .asciiz "Enter value for x: "  prompt\_y: .asciiz "Enter value for y: "  result\_msg: .asciiz "Result of (4x / 3) \* y = "  .text  .globl main  main:  # Prompt and read x  li $v0, 4  la $a0, prompt\_x  syscall  li $v0, 5  syscall  move $t0, $v0 # $t0 = x  # Prompt and read y  li $v0, 4  la $a0, prompt\_y  syscall  li $v0, 5  syscall  move $t1, $v0 # $t1 = y  # Calculate 4x  li $t2, 4  mul $t2, $t2, $t0 # $t2 = 4x  # Divide 4x by 3  li $t3, 3  div $t2, $t3  mflo $t4 # $t4 = 4x / 3 (quotient)  # Multiply (4x / 3) by y  mul $t5, $t4, $t1 # $t5 = (4x / 3) \* y  # Print result  li $v0, 4  la $a0, result\_msg  syscall  li $v0, 1  move $a0, $t5  syscall  # Exit  li $v0, 10  syscall |



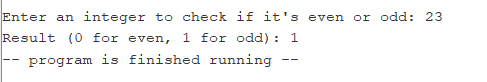
1. Write a program (MIPS Assembly) to retrieve two numbers from a user and swap those numbers using only the *XOR* operation. You should not use a temporary variable to store the numbers while swapping them. Your program should include a proper and useful prompt for input, and print the results in a meaningful manner.

|  |
| --- |
| .data  prompt1: .asciiz "Enter the first number: "  prompt2: .asciiz "Enter the second number: "  before\_swap: .asciiz "\nBefore swap:\n"  after\_swap: .asciiz "\nAfter swap:\n"  first\_num: .asciiz "First number: "  second\_num: .asciiz "\nSecond number: "  .text  .globl main  main:  # Prompt for first number  li $v0, 4  la $a0, prompt1  syscall  # Read first number  li $v0, 5  syscall  move $t0, $v0 # Store first number in $t0  # Prompt for second number  li $v0, 4  la $a0, prompt2  syscall  # Read second number  li $v0, 5  syscall  move $t1, $v0 # Store second number in $t1  # Print numbers before swap  li $v0, 4  la $a0, before\_swap  syscall  li $v0, 4  la $a0, first\_num  syscall  li $v0, 1  move $a0, $t0  syscall  li $v0, 4  la $a0, second\_num  syscall  li $v0, 1  move $a0, $t1  syscall  # Swap numbers using XOR  xor $t0, $t0, $t1 # $t0 = $t0 XOR $t1  xor $t1, $t0, $t1 # $t1 = $t0 XOR $t1 = ($t0 XOR $t1) XOR $t1 = original $t0  xor $t0, $t0, $t1 # $t0 = $t0 XOR $t1 = ($t0 XOR $t1) XOR (original $t0) = original $t1  # Print numbers after swap  li $v0, 4  la $a0, after\_swap  syscall  li $v0, 4  la $a0, first\_num  syscall  li $v0, 1  move $a0, $t0  syscall  li $v0, 4  la $a0, second\_num  syscall  li $v0, 1  move $a0, $t1  syscall  # Exit program  li $v0, 10  syscall |



1. 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.

|  |
| --- |
| .data  prompt: .asciiz "Enter an integer to check if it's even or odd: "  result\_msg: .asciiz "Result (0 for even, 1 for odd): "  .text  .globl main  main:  # Prompt user for input  li $v0, 4  la $a0, prompt  syscall  # Read integer from user  li $v0, 5  syscall  move $t0, $v0 # Store input in $t0  # Check if even or odd using only sll and srl  sll $t1, $t0, 31 # Shift left by 31 bits, leaving only the least significant bit  srl $t1, $t1, 31 # Shift right by 31 bits, moving the LSB back to the rightmost position  # Print result message  li $v0, 4  la $a0, result\_msg  syscall  # Print the result (0 for even, 1 for odd)  li $v0, 1  move $a0, $t1  syscall  # Exit program  li $v0, 10  syscall |



1. 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*.

|  |
| --- |
| .data  prompt1: .asciiz "Enter the first number: "  prompt2: .asciiz "Enter the second number (a prime number): "  result\_msg: .asciiz "Result (0 if prime factor, remainder otherwise): "  .text  .globl main  main:  # Prompt for first number  li $v0, 4  la $a0, prompt1  syscall  # Read first number  li $v0, 5  syscall  move $t0, $v0 # Store first number in $t0  # Prompt for second number  li $v0, 4  la $a0, prompt2  syscall  # Read second number  li $v0, 5  syscall  move $t1, $v0 # Store second number in $t1  # Perform division  div $t0, $t1  mfhi $t2 # Store remainder in $t2  # Print result message  li $v0, 4  la $a0, result\_msg  syscall  # Print the result (remainder)  li $v0, 1  move $a0, $t2  syscall  # Exit program  li $v0, 10  syscall |

