**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #3**

**Due day: 3/3/2023**

**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 a 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 "\nEnter an integer: "

prompt1: .asciiz "\nResult by multiplying 10 is : "

.text

li $v0,4

la $a0,prompt #it will print prompt

syscall

li $v0,5

syscall #ask user input

move $t1,$v0 #save a to t1

sll $s0,$t1,3 #get 8x

sll $s1,$t1,1 #get 2x

add $s3,$s0,$s1 #get multiply by 10x

li $v0,4

la $a0,prompt1 #it will print prompt

syscall

move $a0,$s3

li $v0,1

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.**

A.

.data

x : .word 1

y : .word 2

z : .word 3

f : .word 0

.text

.globl main

main:

lw $s1, x

lw $s2, y

lw $s3, z #s3 = z

mul $s1, $s1, 5 # $s1 = 5x

mul $s2, $s2, 3 # $s2 = 3x

add $s0, $s1, $s2 # $s0 = 5x + 3y

add $s0, $s0, $s3 # s0 = 5x + 3y + z

la $t0, f #$t0 = &f

sw $s0, ($t0) # f = 5x + 3y + z

**B.**

.data

x: .word 0

y: .word 0

z: .word 0

result: .word 0

input\_x: .asciiz "Enter x: "

input\_y: .asciiz "Enter y: "

input\_z: .asciiz "Enter z: "

output: .asciiz "Result = "

.text

.globl main

main:

# Prompt user for input

li $v0, 4

la $a0, input\_x

syscall

# Read x from user input

li $v0, 5

syscall

move $s1, $v0

# Prompt user for input

li $v0, 4

la $a0, input\_y

syscall

# Read y from user input

li $v0, 5

syscall

move $s2, $v0

# Prompt user for input

li $v0, 4

la $a0, input\_z

syscall

# Read z from user input

li $v0, 5

syscall

move $s3, $v0

# Calculate the expression ((5x + 3y + z) / 2) \* 3 and store the result in memory

lw $t0, x

lw $t1, y

lw $t2, z

mul $t0, $s1, 5 # 5x

mul $t1, $s2, 3 # 3y

add $t0, $t0, $t1 # 5x + 3y

add $t0, $t0, $t2 # 5x + 3y + z

div $t0, $t0, 2 # (5x + 3y + z) / 2

mul $t0, $t0, 3 # ((5x + 3y + z) / 2) \* 3

sw $t0, result

# Display the result to the user

li $v0, 4

la $a0, output

syscall

lw $a0, result

li $v0, 1

syscall

# Exit the program

li $v0, 10

syscall

**C.**

.data

x: .word 0

result: .word 0

input: .asciiz "Enter x: "

output: .asciiz "Result = "

.text

.globl main

main:

# Prompt user for input

li $v0, 4

la $a0, input

syscall

# Read x from user input

li $v0, 5

syscall

move $s1, $v0

# Calculate the expression x^3 + 2x^2 + 3x + 4 and store the result in memory

lw $t0, x

mul $t1, $s1, $s1 # x^2

mul $t0, $t0, $s1 # x^3

add $t1, $t1, $t1 # 2x^2

add $t1, $t1, $s1 # 2x^2 + 3x

add $t0, $t0, $t1 # x^3 + 2x^2 + 3x

addi $t0, $t0, 4 # x^3 + 2x^2 + 3x + 4

sw $t0, result

# Display the result to the user

li $v0, 4

la $a0, output

syscall

lw $a0, result

li $v0, 1

syscall

# Exit the program

li $v0, 10

syscall

D.

.data

prompt\_x: .asciiz "Enter the value of x: "

prompt\_y: .asciiz "Enter the value of y: "

result\_msg: .asciiz "The result is: "

.text

main:

# Prompt the user for x

li $v0, 4

la $a0, prompt\_x

syscall

# Read in x

li $v0, 5

syscall

move $t0, $v0

# Prompt the user for y

li $v0, 4

la $a0, prompt\_y

syscall

# Read in y

li $v0, 5

syscall

move $t1, $v0

# Calculate (4x / 3) \* y and store in $t2

mult $t0, 4

mflo $t0

div $t0, 3

mflo $t0

mult $t0, $t1

mflo $t2

# Print the result

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t2

syscall

# Exit the program

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

msg1: .asciiz "Enter the first number\n"

msg2: .asciiz "Enter the second number\n"

msg3: .asciiz "First number after swaping \n"

msg4: .asciiz "\nSecond number after swaping\n"

.text

la $a0,msg1

li $v0,4

syscall #print the msg1

li $v0,5

syscall # read the first number and store in $v0.

move $t0,$v0 # copy the first number in $v0 to $t0.

la $a0,msg2

li $v0,4

syscall #print the msg2.

li $v0,5

syscall # read the second number.

move $t1,$v0 # copy the second number in $v0 to $t1.

xor $t0,$t0,$t1 # 3 xor instructions to swap the numbers in $t0 and $t1.

xor $t1,$t0,$t1

xor $t0,$t0,$t1

la $a0,msg3

li $v0,4

syscall # print the msg3.

li $v0,1

move $a0,$t0 # copy the number in $t0 to $a0 for print.

syscall # print the first number after swaping.

la $a0,msg4

li $v0,4

syscall # print the msg4.

li $v0,1

move $a0,$t1 # copy the number in $t1 to $a0 for print.

syscall # print the second number after swaping.

li $v0,10

syscall #end of program

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 a number to be checked: "

message: .asciiz "\nResult"

msgEven: .asciiz "\n0\nNumber is even"

msgOdd: .asciiz "\n1\nNumber is odd"

.text

main:

#prompt for input

li $v0,4

la $a0,prompt

syscall

#number input

li $v0,5

syscall

move $t0,$v0#move number from v0 to so

srl $s0,$t0,1#Shift right by 1 bit

sll $t1,$s0,1#shift left by 1 bit

#if number is even then original number is equal to the number after SLL and SRL

beq $t0,$t1, resultEven#if true then jump to result Even

bne $t0,$t1, resultOdd#if true then jump to result Odd

#END

li $v0,10

la $a0,message

syscall

resultEven:#Jump here when number is even

li $v0,4

la $a0,msgEven

syscall

li $v0,10

la $a0,message

syscall

resultOdd:#jump here when number is odd

li $v0,4

la $a0,msgOdd

syscall

li $v0,10

la $a0,message

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

msg1: .asciiz "Enter the first number:" # Store prompt messages on data segment

msg2: .asciiz "Enter the second number which is prime:"

.text

la $a0,msg1 # Load address of first prompt message to $a0

li $v0,4

syscall # Print string

li $v0,5 # Read the first input number to $v0

syscall

move $t0,$v0 # Move first input to $t0

la $a0,msg2 # Load address of second prompt message to $a0

li $v0,4

syscall # Print string

li $v0,5 # Read the second input number to $v0

syscall

move $t1,$v0 # Move second input to $t1

div $t0,$t1 # Divide first number with second prime number

mfhi $a0 # Move remainder to $a0

li $v0,1 # Output the remainder value

syscall

li $v0,10 # Terminate program

syscall