**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)**

**Github link:** [**https://github.com/boktiar19795/EE488-Assginment-3-17th-Mar**](https://github.com/boktiar19795/EE488-Assginment-3-17th-Mar)

1. 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.

Answer:  **# Multiply input by 10 using shift and add operations**

#Declations

.data

message: .asciiz "Please enter a number: "

output: .asciiz "The output is: "

.text

main:

# user input prompt

li $v0, 4

la $a0, message

syscall

# Read input value

li $v0, 5

syscall

move $s0, $v0 # move $sv0 to $s0 to save value

# Multiply input by 10 using shift and add operations

sll $s1, $s0, 2 # left shift by 2 bits to multiply by 4

add $s1, $s1, $s0 # add input to the result to get final multiplication of 10

sll $s1, $s1, 1 # left shift by 1 bit to multiply by 2

# Print output

li $v0, 4

la $a0, output

syscall

# Print the output in $s1

li $v0, 1

move $a0, $s1

syscall

# Close the program

li $v0, 10

syscall

**# Multiply input by 10 using *mult* and *mflo* operators**

#Declations

.data

message: .asciiz "Please Enter a number: "

output: .asciiz "The outputis: "

.text

main:

# User input prompt

li $v0, 4

la $a0, message

syscall

# Read input value

li $v0, 5

syscall

move $s0, $v0 # move from $v0 to $s0

# Deploy mult and mflo operators to multiply by 10

li $t0, 10

mult $s0, $t0

mflo $s1

# Print output

li $v0, 4

la $a0, output

syscall

# Print the output in $s1

li $v0, 1

move $a0, $s1

syscall

# Close 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.**

**Code:**

#Declations

.data

x: .word 0

y: .word 0

z: .word 0

# User input prompt

.text

main:

# Prompt for x

li $v0, 4

la $a0, x\_prompt

syscall

# Read x

li $v0, 5

syscall

sw $v0, x

# Prompt for y

li $v0, 4

la $a0, y\_prompt

syscall

# Read y

li $v0, 5

syscall

sw $v0, y

# Prompt for z

li $v0, 4

la $a0, z\_prompt

syscall

# Read z

li $v0, 5

syscall

sw $v0, z

# Calculateexpression

lw $t0, x

lw $t1, y

lw $t2, z

add $t3, $zero, $zero # $t3 = 0

addi $t3, $t3, 5 # $t3 = 5

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

addi $t3, $zero, 3 # $t3 = 3

mul $t1, $t1, $t3 # $t1 = 3y

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

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

# Print output

li $v0, 4

la $a0, output

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 10

syscall

# Strings

.data

x\_prompt: .asciiz "Enter x: "

y\_prompt: .asciiz "Enter y: "

z\_prompt: .asciiz "Enter z: "

output: .asciiz "Result: "

**Code:**

#Declations

.data

x: .word 0

y: .word 0

z: .word 0

# User input prompt

.text

main:

# Prompt for x

li $v0, 4

la $a0, x\_prompt

syscall

# Read x

li $v0, 5

syscall

sw $v0, x

# Prompt for y

li $v0, 4

la $a0, y\_prompt

syscall

# Read y

li $v0, 5

syscall

sw $v0, y

# Prompt for z

li $v0, 4

la $a0, z\_prompt

syscall

# Read z

li $v0, 5

syscall

sw $v0, z

# Calculate value

lw $t0, x

lw $t1, y

lw $t2, z

add $t3, $zero, $zero # $t3 = 0

addi $t3, $t3, 5 # $t3 = 5

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

addi $t3, $zero, 3 # $t3 = 3

mul $t1, $t1, $t3 # $t1 = 3y

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

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

srl $t0, $t0, 1 # $t0 = (5x + 3y + z) / 2

addi $t3, $zero, 3 # $t3 = 3

mul $t0, $t0, $t3 # $t0 = ((5x + 3y + z) / 2) \* 3

# Print output

li $v0, 4

la $a0, output

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 10

syscall

# Strings

.data

x\_prompt: .asciiz "input value for x: "

y\_prompt: .asciiz "Enput value for y: "

z\_prompt: .asciiz "nput value for z: "

output: .asciiz "Result: "



**Code:**

#Declations

.data

x: .word 0

# User prompt for input

.text

main:

# Prompt for x

li $v0, 4

la $a0, x\_prompt

syscall

# Read x

li $v0, 5

syscall

sw $v0, x

# calculate the value

lw $t0, x

# x^3

mul $t1, $t0, $t0

mul $t0, $t1, $t0

# 2x^2

add $t1, $zero, $zero # $t1 = 0

addi $t1, $t1, 2 # $t1 = 2

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

# 3x

add $t1, $zero, $zero # $t1 = 0

addi $t1, $t1, 3 # $t1 = 3

mul $t3, $t0, $t1 # $t3 = 3x

# 4

addi $t1, $zero, 4 # $t1 = 4

# x^3 + 2x^2 + 3x + 4

add $t0, $t0, $t2 # $t0 = x^3 + 2x^2

add $t0, $t0, $t3 # $t0 = x^3 + 2x^2 + 3x

add $t0, $t0, $t1 # $t0 = x^3 + 2x^2 + 3x + 4

# Print result

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 10

syscall

# Strings

.data

x\_prompt: .asciiz "Enter x: "

result: .asciiz "Result: "

**Code:**

#Declations

.data

x: .word 0

y: .word 0

# User input prompt

.text

main:

# Prompt for x

li $v0, 4

la $a0, x\_prompt

syscall

# Read x

li $v0, 5

syscall

sw $v0, x

# Prompt for y

li $v0, 4

la $a0, y\_prompt

syscall

# Read y

li $v0, 5

syscall

sw $v0, y

# calculate the value

lw $t0, x

lw $t1, y

# (4x/3)\*y

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

div $t2, $t2, 3 # $t2 = 4x/3

mul $t2, $t2, $t1 # $t2 = (4x/3)\*y

# Print result

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 10

syscall

# Strings

.data

x\_prompt: .asciiz "Input the value for x: "

y\_prompt: .asciiz "Input the value for y: "

result: .asciiz "Result: "

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.

**Code:**

#Declations

.data

num1: .word 0

num2: .word 0

# User input prompt

.text

main:

# Prompt for num1

li $v0, 4

la $a0, num1\_prompt

syscall

# Read num1

li $v0, 5

syscall

sw $v0, num1

# Prompt for num2

li $v0, 4

la $a0, num2\_prompt

syscall

# Read num2

li $v0, 5

syscall

sw $v0, num2

# Swap values using XOR

lw $t0, num1 # $t0 = num1

lw $t1, num2 # $t1 = num2

xor $t0, $t0, $t1 # $t0 = num1 ^ num2

xor $t1, $t0, $t1 # $t1 = (num1 ^ num2) ^ num2 = num1

xor $t0, $t0, $t1 # $t0 = (num1 ^ num2) ^ num1 = num2

# Print swapped values

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 4

la $a0, space

syscall

li $v0, 1

move $a0, $t1

syscall

li $v0, 10

syscall

# Strings

.data

num1\_prompt: .asciiz "Input the first number: "

num2\_prompt: .asciiz "Input the second number: "

result: .asciiz "Swapped values: "

space: .asciiz " "

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.

**Code:**

#Declations

.data

prompt: .asciiz "Please enter a number: "

msg\_even: .asciiz "The number is even (0)\n"

msg\_odd: .asciiz "The number is odd (1)\n"

.text

main:

# Prompt for number

li $v0, 4

la $a0, prompt

syscall

# Read number

li $v0, 5

syscall

move $t0, $v0

# Even or Odd number check condition

srl $t1, $t0, 1 # shift right by 1 to get the MSB

sll $t1, $t1, 1 # shift left by 1 to clear the LSB

xor $t2, $t0, $t1 # get the difference between the original and even number

srl $t2, $t2, 31 # shift right to get the sign bit (0 for even, 1 for odd)

# Output printing

beq $t2, $zero, even

li $v0, 4

la $a0, msg\_odd

syscall

j end

even:

li $v0, 4

la $a0, msg\_even

syscall

end:

# Close the 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*.

**Code:**

#Declations

.data

prompt1: .asciiz "Input the first number: "

prompt2: .asciiz "Input the second number (prime): "

msg\_prime\_factor: .asciiz "The second number is a prime factor of the first number (0)\n"

msg\_not\_prime\_factor: .asciiz "The second number is not a prime factor of the first number\n"

.text

main:

# Prompt for first number

li $v0, 4

la $a0, prompt1

syscall

# Read first number

li $v0, 5

syscall

move $t0, $v0

# Prompt for second number

li $v0, 4

la $a0, prompt2

syscall

# Read second number

li $v0, 5

syscall

move $t1, $v0

# Check if second number is a prime factor of first number

div $t0, $t1 # divide first number by second number

mfhi $t2 # get remainder

# Print result

beq $t2, $zero, prime\_factor

li $v0, 1

li $a0, 2

syscall

j end

prime\_factor:

li $v0, 4

la $a0, msg\_prime\_factor

syscall

end:

# Close the program

li $v0, 10

syscall