**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.

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

prompt: .asciiz "Enter an integer value: "

result\_msg: .asciiz "The result of multiplying by 10 using shifts and addition is: "

verify\_msg: .asciiz "\nVerification using 'mult' and 'mflo' operators, result is: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

# Multiply input by 10 using bit shifts and addition

sll $t1, $t0, 3

sll $t2, $t0, 1

add $t3, $t1, $t2

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t3

syscall

# Verify using mult and mflo operators

li $v0, 4

la $a0, verify\_msg

syscall

li $t4, 10

mult $t0, $t4

mflo $t5

li $v0, 1

move $a0, $t5

syscall

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.**

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

prompt\_x: .asciiz "Enter the value for 'x': "

prompt\_y: .asciiz "Enter the value for 'y': "

prompt\_z: .asciiz "Enter the value for 'z': "

result\_msg: .asciiz "The result of '5x + 3y + z' is: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_x

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, prompt\_y

syscall

li $v0, 5

syscall

move $t1, $v0

li $v0, 4

la $a0, prompt\_z

syscall

li $v0, 5

syscall

move $t2, $v0

# Calculation of "5x"

li $t3, 5

mult $t0, $t3

mflo $t4

# Calculation of "3y"

li $t3, 3

mult $t1, $t3

mflo $t5

# Calculation of "5x + 3y"

add $t6, $t4, $t5

# Calculation of "5x + 3y + z"

add $t7, $t6, $t2

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t7

syscall

li $v0, 10

syscall

==============================

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

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

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

prompt\_z: .asciiz "Enter the value of 'z': "

result\_msg: .asciiz "The result of '((5x + 3y + z) / 2) \* 3' is: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_x

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, prompt\_y

syscall

li $v0, 5

syscall

move $t1, $v0

li $v0, 4

la $a0, prompt\_z

syscall

li $v0, 5

syscall

move $t2, $v0

# Calculation part 01

li $t3, 5

mul $t3, $t3, $t0

li $t4, 3

mul $t4, $t4, $t1

add $t5, $t3, $t4

add $t5, $t5, $t2

# Calculation part 02

sra $t5, $t5, 1

# Calculation part 03

li $t6, 3

mul $t5, $t5, $t6

#result

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t5

syscall

li $v0, 10

syscall

=============================



**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

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

result\_msg: .asciiz "The result of '(x^3 + 2x^2 + 3x + 4)' is: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_x

syscall

li $v0, 5

syscall

move $t0, $v0

# Calculation for 'x^3'

mul $t1, $t0, $t0

mul $t1, $t1, $t0

# Calculation for '2x^2'

mul $t2, $t0, $t0

add $t2, $t2, $t2

# Calculation for '3x'

add $t3, $t0, $t0

add $t3, $t3, $t0

# Calculation for '(x^3 + 2x^2 + 3x + 4)'

add $t4, $t1, $t2

add $t4, $t4, $t3

addi $t4, $t4, 4

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 10

syscall

================================

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

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

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

result\_msg: .asciiz "The result of '((4x/3) \* y)' is: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_x

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, prompt\_y

syscall

li $v0, 5

syscall

move $t1, $v0

# Calculation for '4x'

sll $t2, $t0, 2

# Calculation for "(4x / 3)"

li $t3, 3

div $t2, $t3

mflo $t2

# Calculation for "((4x/3) \* y)"

mul $t4, $t2, $t1

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t4

syscall

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.

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

prompt\_num1: .asciiz "Enter the first number: "

prompt\_num2: .asciiz "Enter the second number: "

result\_msg: .asciiz "\n'After swapping, the numbers are'\n==============================="

num1\_msg: .asciiz "\nFirst number: "

num2\_msg: .asciiz "\nSecond number: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_num1

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, prompt\_num2

syscall

li $v0, 5

syscall

move $t1, $v0

# Swap the numbers using XOR operation

xor $t0, $t0, $t1

xor $t1, $t0, $t1

xor $t0, $t0, $t1

li $v0, 4

la $a0, result\_msg

syscall

#first number after swapping

li $v0, 4

la $a0, num1\_msg

syscall

li $v0, 1

move $a0, $t0

syscall

#second number after swapping

li $v0, 4

la $a0, num2\_msg

syscall

li $v0, 1

move $a0, $t1

syscall

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.

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

prompt: .asciiz "Enter the integer value: "

.text

.globl main

main:

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

# Check if the number is even or odd using sll and srl

srl $t1, $t0, 1

sll $t1, $t1, 1

sub $t2, $t0, $t1

# Check the result and return 0 or 1 directly

beq $t2, $zero, set\_even

li $v0, 1

li $a0, 1

syscall

j end\_program

set\_even:

li $v0, 1

li $a0, 0

syscall

end\_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*.

**Answer:**

*Source file also uploaded in “GitHub” and “canvas” server*

.data

prompt\_num1: .asciiz "Enter the first number: "

prompt\_num2: .asciiz "Enter the second number (prime number): "

.text

.globl main

main:

li $v0, 4

la $a0, prompt\_num1

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, prompt\_num2

syscall

li $v0, 5

syscall

move $t1, $v0

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

div $t0, $t1

mfhi $t2

# If remainder is 0, the second number is a prime factor of the first

bnez $t2, not\_prime\_factor

# Return 0 (second number is a prime factor)

li $v0, 1

li $a0, 0

syscall

li $v0, 10

syscall

not\_prime\_factor:

li $v0, 1

li $a0, 99

syscall

li $v0, 10

syscall

========================================