|  |  |
| --- | --- |
|  | **Computer Organization & Assembly Language**  **BSCS 3rd**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Quiz: 3**

Date: Week 12, 24th May 2023

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO1: CLO statement**  *Simulate the internal representation of data, and show how data is stored and accessed in, I/O modules, and the interconnecting components of the computer systems* | 1 | 2.5 |  |
|  |  |  |
| **Total Marks** | | **2.5** |  |

**Question 1.**

1. Can multiplication of two variables (not constants) be implemented using the bit shift operations. Would you consider using the bit shift operations implementation of multiplication and divide for two variables, or would you always use the mul or div operators in MIPS assembly? Defend your choice.

Multiplication of two variables can be implemented using bit shift operations, specifically left shifts and additions. However, using the mul instruction in MIPS assembly is generally preferred due to its efficiency and simplicity. The mul instruction utilizes dedicated hardware for multiplication, resulting in faster execution compared to bit shift-based implementations. Additionally, using mul improves code readability and maintainability. Here's a short MIPS assembly code snippet showcasing the use of mul for variable multiplication:

.data

var1: .word 5

var2: .word 7

result: .word 0

.text

lw $t0, var1 # Load var1 into $t0

lw $t1, var2 # Load var2 into $t1

mul $t2, $t0, $t1 # Multiply var1 and var2, result in $t2

sw $t2, result # Store the result in memory

In this code, the mul instruction is used to multiply the contents of var1 and var2, storing the result in result.

1. Implement a program 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, or any number if it is not. For example, if the user enter 60 and 5, the program returns 0. If the user enters 62 and 5, the program returns 2.

**Sol**

.data

prompt1: .asciiz "Enter the first number: "

prompt2: .asciiz "Enter the second prime number: "

output\_msg: .asciiz "Result: "

not\_factor\_msg: .asciiz "2"

factor\_msg: .asciiz "0"

.text

.globl main

main:

# Prompt for the first number

li $v0, 4

la $a0, prompt1

syscall

# Read the first number

li $v0, 5

syscall

move $t0, $v0

# Prompt for the second prime number

li $v0, 4

la $a0, prompt2

syscall

# Read the second prime number

li $v0, 5

syscall

move $t1, $v0

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

div $t0, $t1

mfhi $t2

# Output the result

li $v0, 4

la $a0, output\_msg

syscall

beqz $t2, is\_factor

li $v0, 4

la $a0, not\_factor\_msg

syscall

j end

is\_factor:

li $v0, 4

la $a0, factor\_msg

syscall

end:

li $v0, 10

syscall