**\*\*EE488 - Computer Architecture**

**Homework Assignment #3**

**Student ID: 20293**

**Name: Aye Myat Thiri Khin**

### ****Question 1****

**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****

To efficiently multiply a number by 10 using bit shifts, we take advantage of the fact that shifting left by one bit (sll) doubles a number. We can express multiplication by 10 as:

N×10=(N×8)+(N×2)N \times 10 = (N \times 8) + (N \times 2)N×10=(N×8)+(N×2)

Since multiplying by 8 is equivalent to shifting left by 3 bits (sll $t1, $t0, 3) and multiplying by 2 is shifting left by 1 bit (sll $t2, $t0, 1), we can compute:

sll $t1, $t0, 3 # N \* 8

sll $t2, $t0, 1 # N \* 2

add $t3, $t1, $t2 # (N \* 8) + (N \* 2) = N \* 10

To verify correctness, we also compute N \* 10 using mult and retrieve the result with mflo. The program prompts the user for input, processes the multiplication, and displays results meaningfully.

### ****Question 2****

**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.**

#### ****Answer****

The specific expressions were not included in the document, but assuming generic arithmetic expressions like:

Z=A×B+CZ = A \times B + CZ=A×B+C

The MIPS Assembly code will:

1. Prompt the user for values of A, B, and C.
2. Compute A \* B using mult and store the result using mflo.
3. Add C to the result.
4. Display the final value of Z with a user-friendly message.

### ****Question 3****

**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****

Swapping two numbers using XOR without using a temporary variable follows these steps:

xor $t0, $t0, $t1 # Step 1: A = A ^ B

xor $t1, $t0, $t1 # Step 2: B = A ^ B (now B holds original A)

xor $t0, $t0, $t1 # Step 3: A = A ^ B (now A holds original B)

The program prompts the user for two numbers, swaps them using the XOR trick, and displays the results with a clear message.

### ****Question 4****

**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****

A number is even if its least significant bit (LSB) is 0 and odd if it’s 1. We extract the LSB using bitwise shifting:

andi $t1, $t0, 1 # Extract LSB using AND with 1

If $t1 is 0, the number is even; if it’s 1, it’s odd. The program prompts for user input and prints "Even" or "Odd" accordingly.

### ****Question 5****

**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 return 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****

To check divisibility, we use modulo (rem). If A % B == 0, then B is a factor of A, and we return 0. Otherwise, we return the remainder:

div $t0, $t1 # A / B

mfhi $t2 # Get remainder

The program prompts the user, calculates the remainder, and prints the result in a clear, understandable way.