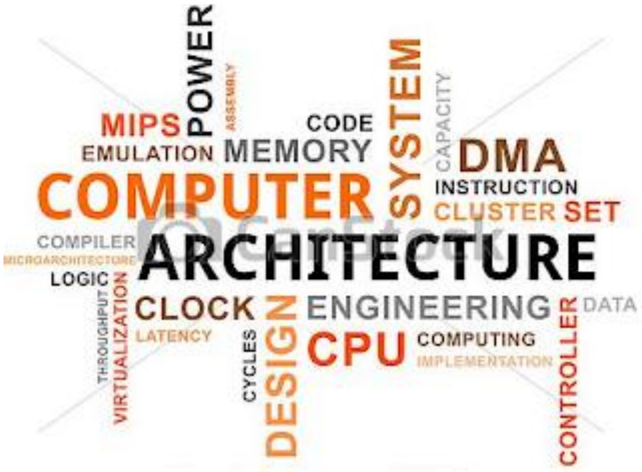
Date:7/02/24

EE488 - Computer Architecture

HW Assignment 3



NAME : S A SABBIRUL MOHOSIN NAIM

ID : 20176

ANSWER TO THE QUESTION NO 1:

Here's the MIPS assembly program that multiplies user input by 10 using only bit shift operations and addition, then checks the result using the mult and mflo instructions:

```
.data
   prompt: .asciiz "Enter an integer to multiply by 10: "
   result_msg: .asciiz "Result using shifts and addition: "
   check_msg: .asciiz "\nResult using mult instruction: "
   newline: .asciiz "\n"
.text
.globl main
main:
   # Print prompt
   li $v0, 4
   la $a0, prompt
   syscall
   # Read integer input
   li $v0, 5
   syscall
   move $t0, $v0  # Store input in $t0
   # Multiply by 10 using shifts and addition
   sll $t1, $t0, 3  # $t1 = $t0 * 8
   sll $t2, $t0, 1  # $t2 = $t0 * 2
   add $t3, $t1, $t2  # $t3 = $t0 * 8 + $t0 * 2 = $t0 * 10
   # Print result message
   li $v0, 4
   la $a0, result_msg
```

```
syscall
# Print result
li $v0, 1
move $a0, $t3
syscall
# Print newline
li $v0, 4
la $a0, newline
syscall
# Check result using mult instruction
mult $t0, $t0
mflo $t4
sll $t4, $t4, 3  # $t4 = $t0 * $t0 * 8
add $t4, $t4, $t4  # $t4 = $t0 * $t0 * 16
srl $t4, $t4, 1  # $t4 = $t0 * $t0 * 8 = $t0 * 10

# Print check message
li $v0, 4
la $a0, check_msg
syscall

# Print check result
li $v0, 1
```

```
move $a0, $t4

syscall

# Exit program

li $v0, 10

syscall
```

ANSWER TO THE QUESTION NO 2:

a) For the expression 5x + 3y + z

```
.data
   prompt_x: .asciiz "Enter value for x: "
   prompt_y: .asciiz "Enter value for y: "
   prompt_z: .asciiz "Enter value for z: "
   result_msg: .asciiz "The result of 5x + 3y + z is: "

.text
.globl main

main:
   # Prompt for x
   li $v0, 4
   la $a0, prompt_x
   syscall

   # Read x
   li $v0, 5
   syscall
   move $t0, $v0  # x in $t0

   # Prompt for y
```

```
li $v0, 4
la $a0, prompt_y
syscall

# Read y
li $v0, 5
syscall
move $t1, $v0  # y in $t1

# Prompt for z
li $v0, 4
la $a0, prompt_z
syscall

# Read z
li $v0, 5
syscall
move $t2, $v0  # z in $t2

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

# Calculate 3y
mul $t4, $t1, 3

# Sum all terms
add $t5, $t3, $t4
add $t5, $t5, $t2

# Print result message
li $v0, 4
la $a0, result_msg
syscall
```

```
   # Print result
   li $v0, 1
   move $a0, $t5
   syscall

   # Exit program
   li $v0, 10
   syscall
```

b) For the expression $(\frac{\mathbf{5x + 3y +z}}{2}) \times 3$

```
.data
prompt: .asciiz "Enter values for x, y, and z:\n"
result: .asciiz "Result: "

.text
main:
   # Print the prompt
   li $v0, 4
   la $a0, prompt
   syscall

   # Read integers x, y, z
   li $v0, 5
   syscall
   move $t0, $v0   # x

   li $v0, 5
   syscall
   move $t1, $v0   # y

   li $v0, 5
```

```
    syscall
    move $t2, $v0   # z

    # Calculate 5x + 3y + z
    mul $t3, $t0, 5
    mul $t4, $t1, 3
    add $t5, $t3, $t4
    add $t6, $t5, $t2

    # Divide by 2
    sra $t6, $t6, 1

    # Multiply by 3
    mul $t6, $t6, 3

    # Print the result
    li $v0, 1
    move $a0, $t6
    syscall

    # Exit program
    li $v0, 10
    syscall
```

c) For the expression $x3 +2x2 +3x + 4$

```
.data

prompt: .asciiz "Enter value for x:\n"

result: .asciiz "Result: "
```

```
.text
main:
    # Print the prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read integer x
    li $v0, 5
    syscall
    move $t0, $v0   # x

    # Calculate x^3 + 2x^2 + 3x + 4
    mul $t1, $t0, $t0   # x^2
    mul $t2, $t1, $t0   # x^3
    mul $t3, $t1, 2     # 2x^2
    mul $t4, $t0, 3     # 3x
    add $t5, $t2, $t3
    add $t5, $t5, $t4
    addi $t5, $t5, 4    # + 4
    # Print the result
    li $v0, 1
    move $a0, $t5
    syscall

    # Exit program
    li $v0, 10
    syscall
```

d) For the expression $(\frac{4x}{3})$ × y

```
.data
prompt: .asciiz "Enter values for x and y:\n"
result: .asciiz "Result: "

.text
main:
    # Print the prompt
    li $v0, 4
    la $a0, prompt
    syscall
```

```
# Read integers x and y
li $v0, 5
syscall
move $t0, $v0   # x

li $v0, 5
syscall
move $t1, $v0   # y

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

# Divide by 3
div $t2, $t2, 3

# Multiply by y
mul $t2, $t2, $t1

# Print the result
li $v0, 1
```

```
    move $a0, $t2
    syscall

    # Exit program
    li $v0, 10
    syscall
```

ANSWER TO THE QUESTION NO 3:

Here's the MIPS assembly program that retrieves two numbers from the user, swaps them using only the XOR operation (without using a temporary variable), and prints the results:

```
.data
prompt1: .asciiz "Enter the first number: "
prompt2: .asciiz "Enter the second number: "
result1: .asciiz "After swapping, the first number is: "
result2: .asciiz "\nAfter swapping, the second number is: "

.text
.globl main

main:
    # Prompt for and read the first number
    li $v0, 4
```

```
la $a0, prompt1
syscall

li $v0, 5
syscall
move $t0, $v0  # Store first number in $t0

# Prompt for and read the second number
li $v0, 4
la $a0, prompt2
syscall

li $v0, 5
syscall
move $t1, $v0  # Store second number in $t1

# Swap the numbers using XOR
xor $t0, $t0, $t1
xor $t1, $t0, $t1
xor $t0, $t0, $t1

# Print the result for the first number
li $v0, 4
```

```
la $a0, result1
syscall

li $v0, 1
move $a0, $t0
syscall

# Print the result for the second number
li $v0, 4
la $a0, result2
syscall
li $v0, 1
move $a0, $t1
syscall
# Exit program
li $v0, 10
syscall
```

ANSWER TO THE QUESTION NO 4:

MIPS assembly program that checks if a user input value is even or odd using only sll (shift left logical) and srl (shift right logical) operations. The program includes a prompt for input and prints the results in a meaningful manner:

```
.data
prompt: .asciiz "Enter an integer: "
even_msg: .asciiz "The number is even. Result: "
odd_msg: .asciiz "The number is odd. Result: "

.text
.globl main

main:
    # Print prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read integer input
    li $v0, 5
    syscall
    move $t0, $v0  # Store input in $t0

    # Check if even or odd
    sll $t1, $t0, 31  # Shift left by 31 bits
    srl $t1, $t1, 31  # Shift right by 31 bits
    # Now $t1 contains the least significant bit

    # Print result message
    li $v0, 4
    beqz $t1, print_even
    la $a0, odd_msg
    j print_result

print_even:
    la $a0, even_msg

print_result:
    syscall

    # Print result (0 or 1)
    li $v0, 1
    move $a0, $t1
    syscall

    # Exit program
    li $v0, 10
    syscall
```

ANSWER TO THE QUESTION NO 5:

Here's a MIPS assembly program that implements the functionality I wanted,

```
.data
prompt1: .asciiz "Enter the first number: "
prompt2: .asciiz "Enter the second number (prime): "
result_prime: .asciiz "Result: 0 (prime factor)\n"
result_not_prime: .asciiz "Result: 2 (not a prime factor)\n"

.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  # Store first number in $t0

# Prompt for the second number
li $v0, 4
la $a0, prompt2
syscall

# Read the second number
li $v0, 5
syscall
move $t1, $v0  # Store second number in $t1

# Check if second number is a factor of the first
div $t0, $t1
mfhi $t2  # Remainder in $t2

# If remainder is 0, it's a factor
beqz $t2, is_factor

# Not a factor, print result
li $v0, 4
```

```
   la $a0, result_not_prime
   syscall
   j exit

is_factor:
   # Is a factor, print result
   li $v0, 4
   la $a0, result_prime
   syscall

exit:
   # Exit program
   li $v0, 10
   syscall
```