**CCCN 221 – Computer Architecture**

**LAB#6 Task5**

**Task Date: As per BB Submission Date: As per BB**

**Student Name: Amin Yahya Selhabi Student ID: 2140632**

**Note: Student must attach the code and the** **screenshot of the Final output using MIPS or Qtsmpm. *Copy and cheating will be mark “0”***

**Marks:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exercises | 1 | 2 | 3 | Total |
| Allocated | 1 | 1 | 1 | 3 |
| Obtained |  |  |  |  |
| **CLO, PLO, SO** | 3.1, V3, S05 | 3.1, V3, S05 | 3.1, V3, S05 |  |

1. **Task**

Write a program that prompts the user to enter an integer and determines whether it is divisible by 5 and 6 but not both. For example, if your input is 10, the output should be**.** *Read Lab6 sheet.*

(**Hint:** Use the **AND , OR ,X\_OR** operators)

***screenshot of the Final output using MIPS or Qtsmpm (without screenshot marks will be deducted).***

**Output Sample.**

|  |
| --- |
| Is 10 divisible by 5 and 6? false  Is 10 divisible by 5 or 6? true  Is 10 divisible by 5 or 6, but not both? True |

1. **Task**

Implement the following assembly program in Mars MIPS that performs the following tasks:

**Write a program that prints the numbers from 1 to given number N.** *Read Lab sheets.*

***screenshot of the Final output using MIPS or Qtsmpm (without screenshot marks will be deducted).***

1. Implement the following assembly program in Mars MIPS that performs the following tasks:

**Write a program that asks user to enter Numbers and displays the Average.** *Read Lab sheets.*

***screenshot of the Final output using MIPS or Qtsmpm (without screenshot marks will be deducted)***

Contents

[Task 1: 3](#_Toc124759951)

[Task 2: 5](#_Toc124759952)

[Task 3: 6](#_Toc124759953)

[Task 1 code: 7](#_Toc124759954)

[Task 2 code: 12](#_Toc124759955)

[Task 3 code: 13](#_Toc124759956)

[References 15](#_Toc124759957)

**Code included in the end**

# Task 1:

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

# Task 2:

Graphical user interface, application

Description automatically generated

# Task 3:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# Task 1 code:

.data

Num: .asciiz "Please Enter a Number: "

is: .asciiz "\nIs "

andd: .asciiz " divisible by 5 and 6? "

orr: .asciiz " divisible by 5 or 6? "

xorr: .asciiz " divisible by 5 or 6, but not both?? "

true: .asciiz "True "

false: .asciiz "False "

.text

# Asking for input

li $v0, 4

la $a0, Num

syscall

li $v0, 5

syscall

move $s0, $v0

# Dividing by 5

div $t0, $s0, 5

mfhi $s1

# Checking if reminder is bigger than zero

bnez $s1, True1

move $t1, $zero # zero for false

j Skip1

True1:

addi $s1, $s1, 1 # 1 for true

Skip1:

# Dividing by 6

li $v0, 1

div $t0, $s0, 6

mfhi $s2

# Checking if reminder is bigger than zero

bnez $s2, True2

move $t2, $zero # zero for false

j End

True2:

addi $s2, $s2, 1 # 1 for true

j End # Habbit

# LOGIC Check if both $s1, $s2 are 1 print true for AND

End:

or $t0, $s1, $s2

# Prints out "Is Num divisible by 5 and 6?"

li $v0, 4

la $t3, is

move $a0, $t3

syscall

li $v0, 1

move $a0, $s0

syscall

li $v0, 4

la $a0, andd

syscall

# Checking if 5 and 6 are divisible

beqz $t0, andZero

j andOne

andZero: # Prints "True"

li $v0, 4

la $a0, true

syscall

j orCheck

andOne: # Prints "False"

li $v0, 4

la $a0, false

syscall

j orCheck # Habbit

orCheck:

# Prints out "Is Num divisible by 5 and 6?"

li $v0, 4

la $t3, is

move $a0, $t3

syscall

li $v0, 1

move $a0, $s0

syscall

li $v0, 4

la $a0, orr

syscall

# LOGIC Check if both $s1, $s2 are 1 print true for OR

beqz $s1, orZero

beqz $s2, orZero

j orOne

orZero:

li $v0, 4

la $a0, true

syscall

j xorCheck

orOne:

li $v0, 4

la $a0, false

syscall

j xorCheck # Habbit

xorCheck:

# Prints out "Is Num divisible by 5 and 6 but not both??"

li $v0, 4

la $t3, is

move $a0, $t3

syscall

li $v0, 1

move $a0, $s0

syscall

li $v0, 4

la $a0, xorr

syscall

# LOGIC Check if both $s1, $s2 are 1 print true for XOR

beqz $s1, xorZero

beqz $s2, xorZero

j xorFalse

xorZero:

beqz $t0, xorFalse

li $v0, 4

la $a0, true

syscall

li $v0, 10

syscall

xorFalse:

li $v0, 4

la $a0, false

syscall

# Awesome homework!

# Task 2 code:

# This programme will count from 0 to the giving positive integer

.data

num: .asciiz "Please enter a number: "

count: .word -1

space: .asciiz " "

# Print "Please enter a number: "

.text

li $v0, 4

la $a0, num

syscall

# Input

li $v0, 5

syscall

move $s0, $v0

# Printing numbers from 0 to giving positive integer

lw $t0, count

la $t1, space

loop:

addi $t0, $t0, 1

li $v0, 1

move $a0, $t0

syscall

# Printing Blank " "

li $v0, 4

move $a0, $t1

syscall

blt $t0, $s0, loop #looping

li $v0, 10

syscall

# Task 3 code:

# This programme will calculate the average of giving numbers

.data

num: .asciiz "Please enter a number to continue or a character to quit: "

average: .asciiz "\nThe average is: "

remainder: .asciiz "\nThe remainder is: "

error: .word 0

# Prints "Please enter a number: "

.text

lw $t3, error # To stop the loop when an error is caught

Loop:

li $v0, 4

la $a0, num

syscall

li $v0, 5

syscall

move $s0, $v0

add $t0, $t0, $s0 # Sum of inputs

add $t1, $t1, 1 # Counting the inputs

j Loop

# Special thanks to https://courses.missouristate.edu/kenvollmar/mars/help/MarsExceptions.html for helping me figuring out how to catch an error

.ktext 0x80000180 # Default Location of Exception Handler

# Prints "\nThe average is: "

li $v0, 4

la $a0, average

syscall

# Calculates and prints the quotient

li $v0, 1

div $t0, $t1

mflo $a0

syscall

# Prints "\nThe remainder is: "

li $v0, 4

la $a0, remainder

syscall

li $v0, 1

mfhi $a0

syscall

li $v0, 10

syscall

# Coprocessor 0 register $14 has address of trapping instruction

mfc0 $k0,$14

# Add 4 to point to next instruction

addi $k0,$k0, 4

# Store new address back into $14

mtc0 $k0,$14

# Error return; set PC to value in $14

eret

.kdata

msg: .asciiz "Trap generated\n"

# References

University, M. S. (n.d.). Retrieved from https://courses.missouristate.edu/kenvollmar/mars/help/MarsExceptions.html