**IMPLEMENTATION OF PRIME**

**NUMBERS IN**

**MIPS**

**LAB # 0****5**

**Fall 2023**

**CSE-304L**

**Computer Organization & Architecture Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Bilal Habib**

Thursday, November 2, 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**ASSESSMENT RUBRICS COA LABS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB REPORT ASSESSMENT** | | | | |
| **Criteria** | **Excellent** | **Average** | **Nill** | **Marks Obtained** |
| 1. **Objectives of Lab** | All objectives of lab are properly covered  [Marks 10] | Objectives of lab are partially covered  [Marks 5] | Objectives of lab are not shown  [Marks 0] |  |
| 1. **MIPS instructions with**   **Comments and proper indentations.** | All the instructions are well written with comments explaining the code and properly indented  [Marks 20] | Some instructions are missing are poorly commented code  [Marks 10] | The instructions are not properly written  [Marks 0] |  |
| 1. **Simulation run without error and warnings** | The code is running in the simulator without any error and warnings  [Marks 10] | The code is running but with some warnings or errors.  [Marks 5] | The code is written but not running due to errors  [Marks 0] |  |
| 1. **Procedure** | All the instructions are written with proper procedure  [Marks 20] | Some steps are missing  [Marks 10] | steps are totally missing  [Marks 0] |  |
| 1. **OUTPUT** | Proper output of the code written in assembly  [Marks 20] | Some of the outputs are missing  [Marks 10] | No or wrong output  [Marks 0] |  |
| 1. **Conclusion** | Conclusion about the lab is shown and written  [Marks 20] | Conclusion about the lab is partially shown  [Marks 10] | Conclusion about the lab is not shown [Marks0]  [Marks 0] |  |
| 1. **Cheating** |  |  | Any kind of cheating will lead to 0 Marks |  |
| Total Marks Obtained: \_\_\_\_\_\_\_\_\_\_  Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |

**Implementation of Prime Numbers in MIPS**

Objectives:

* To apply the learned MIPS commands and concepts to implement a logic for checking whether a number is prime or not.
* To extend the implementation to find and display the two largest prime numbers lower than a user-input number.
* To develop a program that takes two user-input limits and displays prime numbers within the specified range.

Tasks:

**Task 1**: Write a program to check whether a number input by user is prime or not.

**Code:**

.text

.globl main

main:

# show the prompt

li $v0, 4

la $a0, prompt

syscall

# take the number $t0=n from user

li $v0, 5

syscall

move $t0, $v0

# implementation of isPrime function

isPrime:

li $t1, 2

blt $t0, $t1, notPrimeNumberLabel

# calculate $t3 = n/2

div $t0, $t1

mflo $t3

# loop from 2 to n/2

loop:

div $t0, $t1

mfhi $t4

beq $t4, $zero, notPrimeNumberLabel

addi $t1, $t1, 1 # increment the index

ble $t1, $t3, loop # check to continue the loop

j isPrimeNumberLabel

notPrimeNumberLabel:

# show that number is not prime

li $v0, 4

la $a0, notPrimeNumberMsg

syscall

j Exit

isPrimeNumberLabel:

# show that number is prime

li $v0, 4

la $a0, isPrimeNumberMsg

syscall

j Exit

Exit:

li $v0, 10

syscall

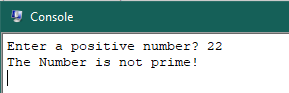
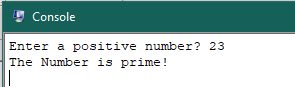
.data

prompt: .asciiz "Enter a positive number? "

notPrimeNumberMsg: .asciiz "The Number is not prime!\n"

isPrimeNumberMsg: .asciiz "The Number is prime!\n"

**Output:**

** **

**Task 2**: Repeat the above problem and display the largest two prime numbers lower than itself. Hint: If a user enters 20, then program displays 19 and 17.

**Code:**

.text

.globl main

main:

# Show the prompt

li $v0, 4

la $a0, prompt

syscall

# Take the number $t0=n from the user

li $v0, 5

syscall

move $t0, $v0

# Initialize $t5 to 2 for the isPrime function

li $t5, 2

# Find the largest two prime numbers lower than n

find\_primes:

addi $t0, $t0, -1

# Check if $t0 is less than 2 (end of search)

blez $t0, exit

# Call isPrime function with $t0 as the argument

move $a0, $t0

jal isPrime

# Check if $v0 (result from isPrime) is 1 (prime)

beq $v0, $zero, find\_primes

# If $v0 is 1, we found a prime, store it in $t1 and check for the second prime

move $t1, $t0 # Store the first prime in $t1

addi $t0, $t0, -1 # Decrement n by 1 to continue searching

# Call isPrime function with $t0 as the argument

move $a0, $t0

jal isPrime

# Check if $v0 (result from isPrime) is 1 (prime)

beq $v0, $zero, exit # If not prime, exit

# If $v0 is 1, we found the second prime, store it in $t2 and exit

move $t2, $t0

j exit

# isPrime function

isPrime:

li $t3, 2 # Initialize $t3 to 2

# Check if n < 2

blt $a0, $t3, notPrimeNumber

# Calculate n/2 and store it in $t4

div $a0, $t3

mflo $t4

# Loop from 2 to n/2

isPrimeLoop:

div $a0, $t3

mfhi $t6

beq $t6, $zero, notPrimeNumber

addi $t3, $t3, 1 # Increment the index

ble $t3, $t4, isPrimeLoop # Check to continue the loop

# Number is prime, set $v0 to 1

li $v0, 1

j isPrimeExit

notPrimeNumber:

# Number is not prime, set $v0 to 0

li $v0, 0

isPrimeExit:

jr $ra # Return to the caller

exit:

# Display the largest two prime numbers

li $v0, 4

la $a0, primeResult1

syscall

li $v0, 1

move $a0, $t1

syscall

li $v0, 4

la $a0, primeResult2

syscall

li $v0, 1

move $a0, $t2

syscall

# Exit the program

li $v0, 10

syscall

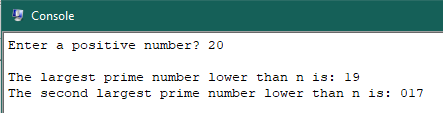
.data

prompt: .asciiz "Enter a positive number? "

primeResult1: .asciiz "\nThe largest prime number lower than n is: "

primeResult2: .asciiz "\nThe second largest prime number lower than n is: "

**Output:**

****

**Task 3**: Write a program which takes two limits from user and display prime numbers between the two limits (if user enter lower limit 10 and upper limit 30 then display prime numbers between 10 and 30)

**Code:**

.text

.globl main

main:

# show the prompt for lower number

li $v0, 4

la $a0, prompt1

syscall

# take the number $t0=n1 from user

li $v0, 5

syscall

move $t0, $v0

# show the prompt for upper number

li $v0, 4

la $a0, prompt2

syscall

# take the number $t1=n2 from user

li $v0, 5

syscall

move $t1, $v0

# taking loop from n1 to n2

loopFromLower2Upper:

bgt $t0, $t1, Exit

j isPrime

# implementation of isPrime function

isPrime:

li $t2, 2

blt $t0, $t2, notPrimeNumberLabel

# calculate $t3 = n/2

div $t0, $t2

mflo $t3

# loop from 2 to n/2

loop:

div $t0, $t2

mfhi $t4

beq $t4, $zero, notPrimeNumberLabel

addi $t2, $t2, 1 # increment the index

ble $t2, $t3, loop # check to continue the loop

j isPrimeNumberLabel

notPrimeNumberLabel:

# only increment the loop

addi $t0, $t0, 1

j loopFromLower2Upper

isPrimeNumberLabel:

# show that number is prime

li $v0, 1

move $a0, $t0

syscall

li $v0, 4

la $a0, comma

syscall

addi $t0, $t0, 1

j loopFromLower2Upper

Exit:

li $v0, 10

syscall

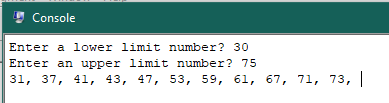
.data

prompt1: .asciiz "Enter a lower limit number? "

prompt2: .asciiz "Enter an upper limit number? "

comma: .asciiz ", "

**Output:**

****

Reference:

To view my codes, please refer to my [GitHub Account.](https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/computer_organization_and_architechure_lab/lab_reports/)

Conclusion:

This lab has provided me with a practical opportunity to apply my knowledge of MIPS assembly language to solve real-world mathematical problems, enhancing my understanding of computer organization and architecture (COA). It has also reinforced my skills in algorithmic thinking and problem-solving, which are essential in the field of computer science and engineering.

The End.