**Mini Project Report  
Week 8 - 9**

|  |  |
| --- | --- |
| Student ID | 20235890 |
| Student Name | Nguyễn Đức Anh |

**Problem A-3:**

**Code:**

.data

    Input\_M:    .asciz "Input M: "

    Input\_N:    .asciz "Input N: "

    prime\_msg:   .asciz "Prime Num: "

    newline:     .asciz "\n"

.text

start:

    # Print Input messenger for M

    la a0, Input\_M

    li a7, 4

    ecall

    # Input M

    li a7, 5

    ecall

    addi s0, a0, 0

    # Print Input messenger for N

    la a0, Input\_N

    li a7, 4

    ecall

    # Input N

    li a7, 5

    ecall

    addi s1, a0, 0

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Prime Number Checking Algorithms:

# Base Case: if t0 (M) = 2 -> print M

# Even numbers > 2 basically is not the prime number -> ignore

# 1 is the odd number but not the prime number

# To check other numbers: Start from the divisor 3 and only check odd divisors up to sqrt(t0),

# Because any composite number greater than 2 must have at least one factor less than or equal to sqrt(t0).

# Increse the divisor by 2 to skip even numbers.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    init\_index:

        addi t0, s0, 0  # t0 = M (starting from M)

    check\_prime:

        bgt t0, s1, end # Check if t0 > N -> jump to end

        # Case: Check if t0 == 2

        li t2, 2

        beq t0, t2, print\_prime

        # Check if t0 < 2 or t0 is even (t0 % 2 == 0)

        li t3, 2

        blt t0, t3, next\_num   # If t0 < 2, jump to next number

        rem t4, t0, t3         # t4 = t0 % t3 (M % 2)

        beq t4, zero, next\_num # If t0 is even, jump to next number

        # Set divisor to 3 and Bool value to check prime to 1 (true)

        li t1, 3 # Divisor starts at 3

        li t2, 1 # Prime Check value (1 (true) means prime, 0 (false) means not prime)

    is\_prime\_loop:

        # Break loop if divisor (t1) \* divisor (t1) > current number (t0)

        mul t3, t1, t1

        bgt t3, t0, print\_prime

        # Check if t0 % t1 == 0

        rem t4, t0, t1

        beq t4, zero, not\_prime

        # Increment divisor by 2 (only odd divisors)

        addi t1, t1, 2

        j is\_prime\_loop

    not\_prime:

        li t2, 0 # Set Prime check value to 0 (false) means not prime

    print\_prime:

        beq t2, zero, next\_num # If Prime check is still 1, print the number

        # Print "Prime Num: "

        la a0, prime\_msg

        li a7, 4

        ecall

        # Print the prime number (t0)

        mv a0, t0

        li a7, 1

        ecall

        # Print newline

        la a0, newline

        li a7, 4

        ecall

    next\_num:

        # Increse t0 and check the next number

        addi t0, t0, 1

        j check\_prime

end:

    # Exit the program

    li a7, 10

    ecall

**Explaination:** (Idea, how your code work, task reponse, etc…)

**Result:** (Input, output for each case; is the result same as the theory, etc…)

**Problem B-3:**

**Code:**

**Explaination:** (Idea, how your code work, task reponse, etc…)

**Result:** (Input, output for each case; is the result same as the theory, etc…)

**Problem C-13:**

**Code:**

.data

    A: .space 32    # String A

    B: .space 32    # String B

.text

start:

    # Get input string A from keyboard

    li a7, 8

    la a0, A

    li a1, 32

    ecall

    # Get input string B from keyboard

    li a7, 8

    la a0, B

    li a1, 32

    ecall

    # Set up pointers for iterating through A and B

    la t0, A              # t0 points to start of string A

    la t1, B              # t1 points to start of string B

    loop\_A:

        lb t2, 0(t0)            # Load a byte from string A into t2

        beq t2, zero, end       # If it's null (end of string A), we're done

        # Check if t2 is a lowercase letter (between 'a' and 'z')

        li t5, 'a'              # Load 'a' into t5

        slt t6, t2, t5          # if t2 < 'a', t6 = 1 or 0

        bne t6, zero, next\_A    # If t2 < 'a', skip to next\_A

        li t5, 'z'              # Load 'z' into t5

        slt t6, t5, t2          # if 'z' < t2, t6 = 1 or 0

        bne t6, zero, next\_A    # If 'z' < t2, skip to next\_A

        # Check if character in A appears in B

        la t1, B                # Reset t1 to the start of string B

        addi t3, zero, 0        # Clear flag (t3 = 0) to indicate not found in B

    loop\_B:

        lb t4, 0(t1)              # Load a byte from string B into t4

        beq t4, zero, check\_print # If end of B, check flag

        beq t2, t4, found\_B       # If char in A matches B, set found flag

        addi t1, t1, 1            # Move to next character in B

        j loop\_B                  # Repeat loop for B

    found\_B:

        li t3, 1              # Set flag indicating character was found in B

    check\_print:

        beq t3, zero, print\_char # If not found in B, print char in A

        j next\_A              # Otherwise, move to next char in A

    print\_char:

        mv a0, t2             # Character to print

        li a7, 11             # Syscall code for putchar

        ecall                 # System Call

    next\_A:

        addi t0, t0, 1        # Move to next character in A

        j loop\_A              # Repeat loop for A

end:

    li a7, 10             # Syscall code for exit

    ecall                 # Exit the program

**Explaination:** (Idea, how your code work, task reponse, etc…)

**Result:** (Input, output for each case; is the result same as the theory, etc…)