

# BÁO CÁO MINI – PROJECT

## Nhóm 2:

1. Phạm Vân Anh – 20214988
2. Kha Minh Bảo – 20210098

Ex2: Find all prime numbers (such as 2, 3, 5, 7..) in a range from the integer N to the integer M. N and M are inputted from keyboard.

### 1. Code

```
.data
    InN:      .asciiz "Input N: "
    InM:      .asciiz "Input M: "
    Output:   .asciiz "Prime number in [N,M]: "
    No_number: .asciiz "No number"
    Space:    .asciiz " "
    Endline:  .asciiz "\n"

.text
main:
input_n:
    li    $v0, 4
    la    $a0, InN
    syscall

    li    $v0, 5
    syscall
    add   $t0, $v0, $zero

input_m:
    li    $v0, 4
    la    $a0, InM
    syscall
```

```
li    $v0, 5
syscall
add   $t1, $v0, $zero
```

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

```
slt   $s6, $t1, $t0
bne   $s6, $zero, exit
```

loop: #For i in range[n,m]

```
jal   is_prime
addi  $t0, $t0, 1
sgt   $t8, $t0, $t1
beq   $t8, $zero, loop
```

exit:

```
bne   $t3, $zero, halt
li    $v0, 4
la    $a0, No_number
syscall
```

halt:

```
li    $v0, 10
syscall
```

is\_prime:

```
li    $t2, 2
slt   $t7, $t0, $t2
bne   $t7, $zero, return
```

```

prime_test:
    beq    $t0, $t2, print          #if ($t2 >= n) -> print
    div    $t0, $t2
    mfhi   $s0                     #s0 = $t0 % $t2
    beq    $s0, $zero, return
    addi   $t2, $t2, 1              #$t2+=1
    j      prime_test

print:
    addi   $t3, $zero, 1
    li     $v0, 1
    add    $a0, $t0, $zero
    syscall

    li     $v0, 4
    la     $a0, Space
    syscall

return:
    jr     $ra

```

## 2. Mã giả

```

1. n = input()
2. m = input()
3.
4. if (n>m): return("No number")
5.
6. while (n<=m):
7.     for i in range(2,n): #i<n
8.         if (n%i == 0):
9.             -> not a prime number
10.        -> prime number, print n
11. n=n+1 #increase n

```

### 3. Giải thích

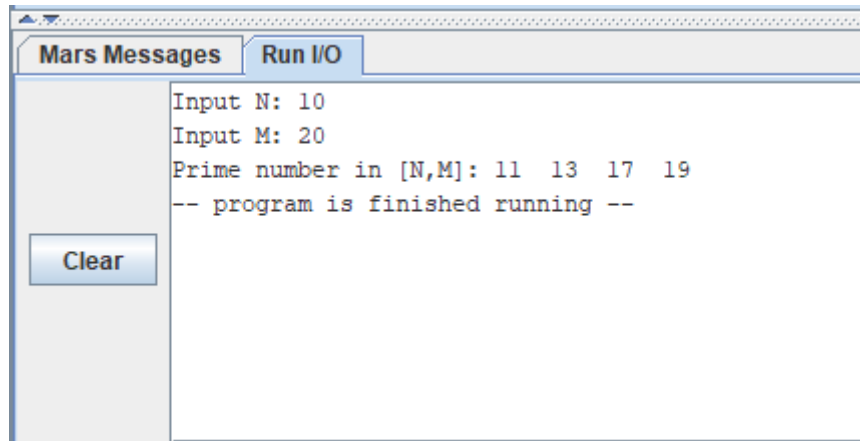
- Bài toán dùng thuật toán tìm số nguyên tố bằng cách: Thực hiện vòng lặp xét lần lượt từng phần tử từ  $N \rightarrow M$  và chia lần lượt từng phần tử cho  $2 \rightarrow M-1$ . Nếu có số nào chia hết cho số nào khác ngoài chính nó  $\rightarrow$  là số nguyên tố và ngược lại không phải là số nguyên tố.
- Hàm *input\_n*: yêu cầu người dùng nhập  $N$ , lưu giá trị của  $N$  vào thanh ghi  $\$t0$
- Hàm *input\_m*:
  - Yêu cầu người dùng nhập  $M$ , lưu giá trị của  $M$  vào thanh ghi  $\$t1$
  - Output hiển thị thông điệp “Prime number in  $[N,M]$ :”
  - Nếu  $\$t0 < \$t1$  ( $M < N$ )  $\rightarrow$  exit
- Hàm *loop*:
  - Kiểm tra lần lượt từng số từ  $N$  đến  $M$  có là số nguyên tố
  - Lệnh jal: Tương tác với hàm *is\_prime*: kiểm tra số hiện tại (lưu trong thanh ghi  $\$t0$ ) có phải là số nguyên tố không
  - Sau mỗi lần lặp, giá trị của  $\$t0$  được tăng thêm 1
  - Sgt: so sánh giá trị hiện tại của  $\$t0$  với  $M$  và lưu kết quả vào  $\$t8$ . Nếu  $\$t0 < M \rightarrow$  loop
- Hàm *exit*:
  - Được thực hiện khi vòng lặp kết thúc
  - Nếu  $\$t3 \neq 0$  (đã tìm thấy ít nhất 1 số nguyên tố)  $\rightarrow$  halt. Ngược lại, hiển thị “No number”
- Hàm *end*: kết thúc chương trình
- Hàm *is\_prime*:
  - Khởi tạo  $\$t2=2$  cho hàm *prime\_test* sử dụng
  - Lọc kết quả, nếu  $\$t0 < 2 \rightarrow$  Không là số nguyên tố
- Hàm *prime\_test*:
  - Vòng lặp kiểm tra tính chia hết của  $\$t0$  (số hiện tại) cho các số từ 2 đến  $\$t2$  ( $M$ ).
  - Khi  $\$t0$  chia cho  $\$t2$ , gán phần dư ở thanh **hi** vào thanh ghi  $\$s0$ . Ví dụ:  $\$t0 = 5, \$t2 = 2$ :

hi		1
lo		2

- So sánh \$s0 và 0. Nếu \$s0 = 0 → chia hết → là số nguyên tố → print
- Nếu \$t0 != 0 → tăng \$t2 lên 1 đơn vị và tiếp tục vòng lặp
- Hàm *print*: hiển thị số nguyên tố, theo sau là kí tự dấu cách
- Hàm *return*: nhảy về lệnh gọi chương trình con

#### 4. Kết quả

- Thử N = 10, M = 20

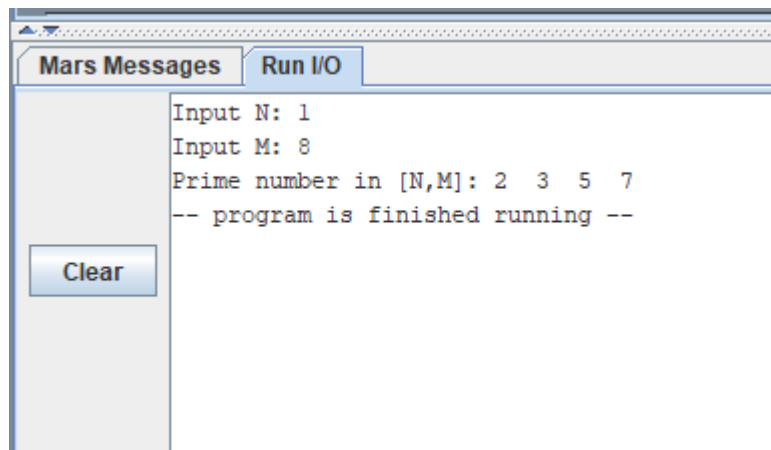


The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The output text is as follows:

```
Input N: 10
Input M: 20
Prime number in [N,M]: 11 13 17 19
-- program is finished running --
```

There is a "Clear" button on the left side of the window.

- Thử N = 1, M = 8

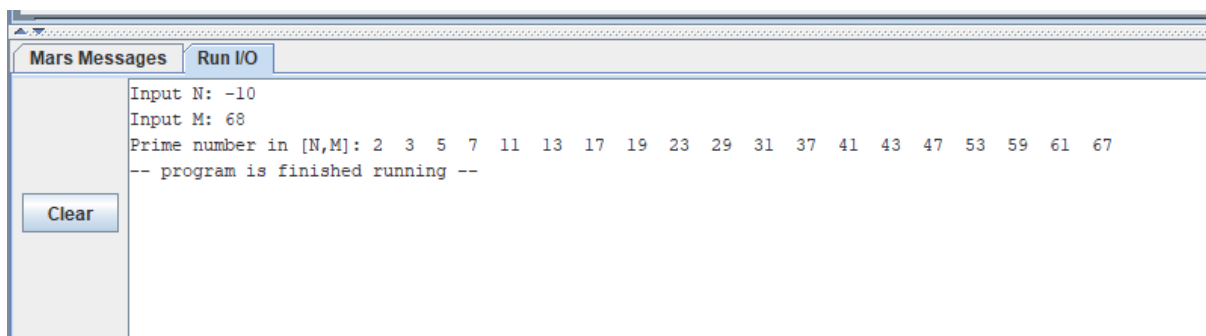


The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The output text is as follows:

```
Input N: 1
Input M: 8
Prime number in [N,M]: 2 3 5 7
-- program is finished running --
```

There is a "Clear" button on the left side of the window.

- Thử N = -10, M = 68

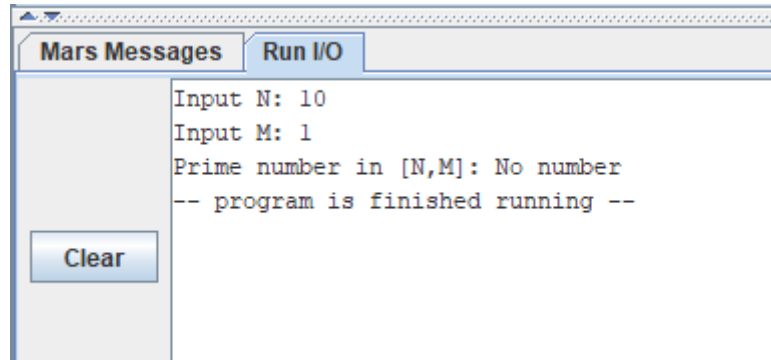


The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The output text is as follows:

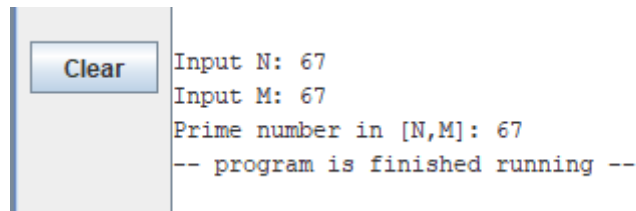
```
Input N: -10
Input M: 68
Prime number in [N,M]: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67
-- program is finished running --
```

There is a "Clear" button on the left side of the window.

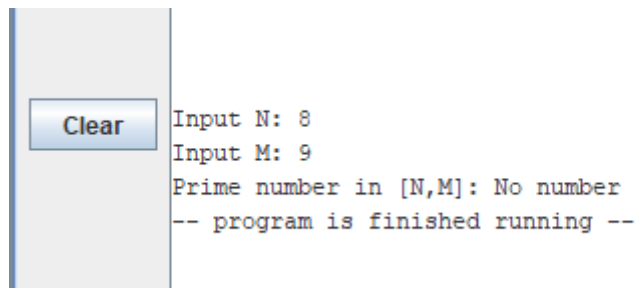
- Thử  $N = 10, M = 1$



- Thử  $N = 67, M = 67$



- Thử  $N = 8, M = 9$



Ex12: Write a function that converts a string of ASCII digits into a 32-bit integer. The function will receive as an argument the starting address of the string and must return a 32-bit integer containing the integer value of the string. Assume that the string is an ASCIIZ string, i.e., ends with the null character (ASCII code 0). You don't need to check for errors in the string, i.e., you may assume the string contains only characters '0' through '9' (i.e., their corresponding ASCII codes), and will not represent a negative number or a non-decimal value or too large a number. For example, `a_to_i` called with the argument "12345" will return the integer 12345.

## 1. Code

```
.data
    Input:      .space 16
    Message:    .ascii "Please input a string number: "
    Exit_code:  .ascii "Input problem"
    Output:     .ascii "32-bit integer: "
    Min_number: .word 48    #0
    Max_number: .word 57    #9
    Enter:      .word 10

.text
main:

    li    $s0, 0 #number
    la    $t7, Enter
    lw    $t7, 0($t7)
    la    $t8, Min_number
    lw    $t8, 0($t8)
    la    $t9, Max_number
    lw    $t9, 0($t9)

input_string:
    li    $v0, 54
    la    $a0, Message
    la    $a1, Input
```

```
la    $a2, 10
```

```
syscall
```

```
la    $t0, Input
```

```
read_char:
```

```
lbu   $t1, 0($t0)
```

```
sle   $s3, $t8, $t1
```

```
beq   $s3, $zero, exit
```

```
sge   $s3, $t9, $t1
```

```
beq   $s3, $zero, exit
```

```
mul   $s0, $s0, 10 #s0*=10
```

```
sub   $t1, $t1, $t8 #ord($t1)
```

```
add   $s0, $s0, $t1 #s0+ = t1
```

```
addi  $t0, $t0, 1      #t0 = $t0 + 1
```

```
j     read_char
```

```
exit:
```

```
beq   $t1, $zero, print
```

```
bne   $t1, $t7, error
```

```
print:
```

```
li     $v0, 4
```

```
la     $a0, Output
```

```
syscall
```

```
li     $v0, 1
```

```
add    $a0, $s0, $zero
```

```
syscall
```

```
j     done
```

```
error:
```

```
li     $v0, 4
```

```
la     $a0, Exit_code
```



```
syscall
```

```
done:
```

```
li    $v0, 10
```

```
syscall
```

## 2. Giải thích

- *.data*
  - Dành 16 byte cho chuỗi nhập (Input)
  - Đặt giá trị tối thiểu và tối đa cho các giá trị ASCII cho các chữ số (*Min\_number* và *Max\_number*)
  - *Enter* có giá trị là mã ASCII của phím Enter
- Hàm *main*:
  - Khởi tạo giá trị của thanh ghi \$s0 thành 0. Thanh ghi này được sử dụng để lưu trữ số nguyên cuối cùng.
  - Gán giá trị biến Enter vào thanh ghi \$t7
  - Gán giá trị của *Min\_number* và *Max\_number* vào các thanh ghi \$t8 và \$t9 tương ứng. Các biến này đại diện cho giá trị ASCII tối thiểu và tối đa của một chữ số.
- Hàm *input\_string*: Input chuỗi đầu vào từ người dùng.
- Hàm *read\_char*:
  - Vòng lặp *read\_char* thực hiện chuyển đổi chuỗi thành số nguyên
  - *lbu \$t1, 0(\$t0)*: tải một byte từ địa chỉ mà thanh ghi \$t0 đang trỏ tới thanh ghi \$t1. Điều này đại diện cho một ký tự trong chuỗi nhập
  - Chương trình kiểm tra đầu vào có là số nguyên hay không (0 → 9)
  - *mul \$s0, \$s0, 10*: Số nguyên trong thanh ghi \$s0 được nhân với 10 bằng lệnh *mul* để tạo ra một chỗ trống cho chữ số tiếp theo. Số chữ số trong \$s0 được tăng lên bằng cách lấy giá trị cuối cùng của \$t1 sau khi trừ đi \$t8.

- Địa chỉ trong \$t0 được tăng thêm 1 để trở đến ký tự tiếp theo trong chuỗi. Sau đó quay lại *read\_char*.
- Kết thúc vòng lặp → *exit*
- Hàm *exit*:
  - Kiểm tra \$t1 có bằng 0 hay không? Nếu có nghĩa là chuỗi đã được đọc đến cuối và không có lỗi xảy ra → *print* để hiển thị số nguyên
  - Kiểm tra \$t1 có khác \$t7 (giá trị của Enter) hay không? Nếu có, chương trình sẽ nhảy tới *Error* để xử lý lỗi.
- Hàm *print*: in số nguyên → *done*
- Hàm *error*: → *Exit\_code*
- Hàm *done*: kết thúc chương trình
- Nếu giả sử số là 12345 thì hàm sẽ thực hiện thuật toán:
 
$$12 = 1 \times 10 + 2$$

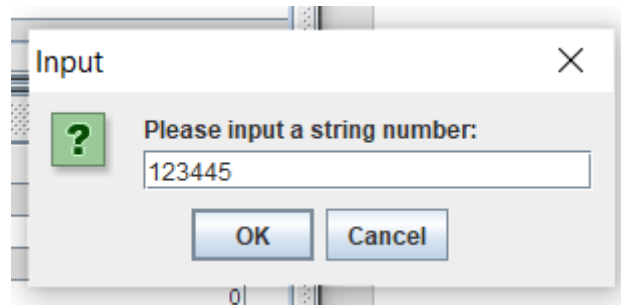
$$123 = 12 \times 10 + 3$$

$$1234 = 123 \times 10 + 4$$

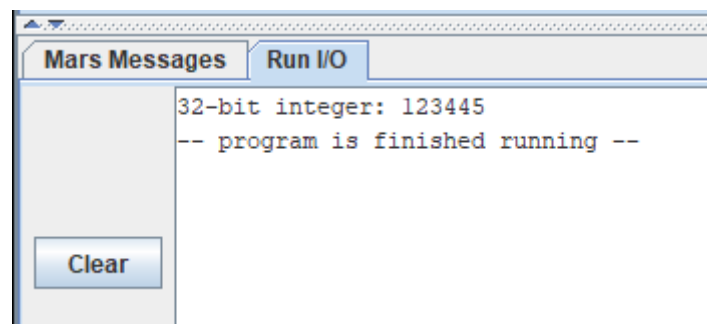
$$12345 = 1234 \times 10 + 5$$

### 3. Kết quả

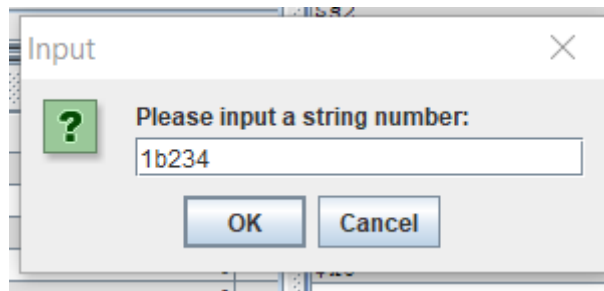
- Thử chuỗi



Kết quả:



- Thử chuỗi



Kết quả:

