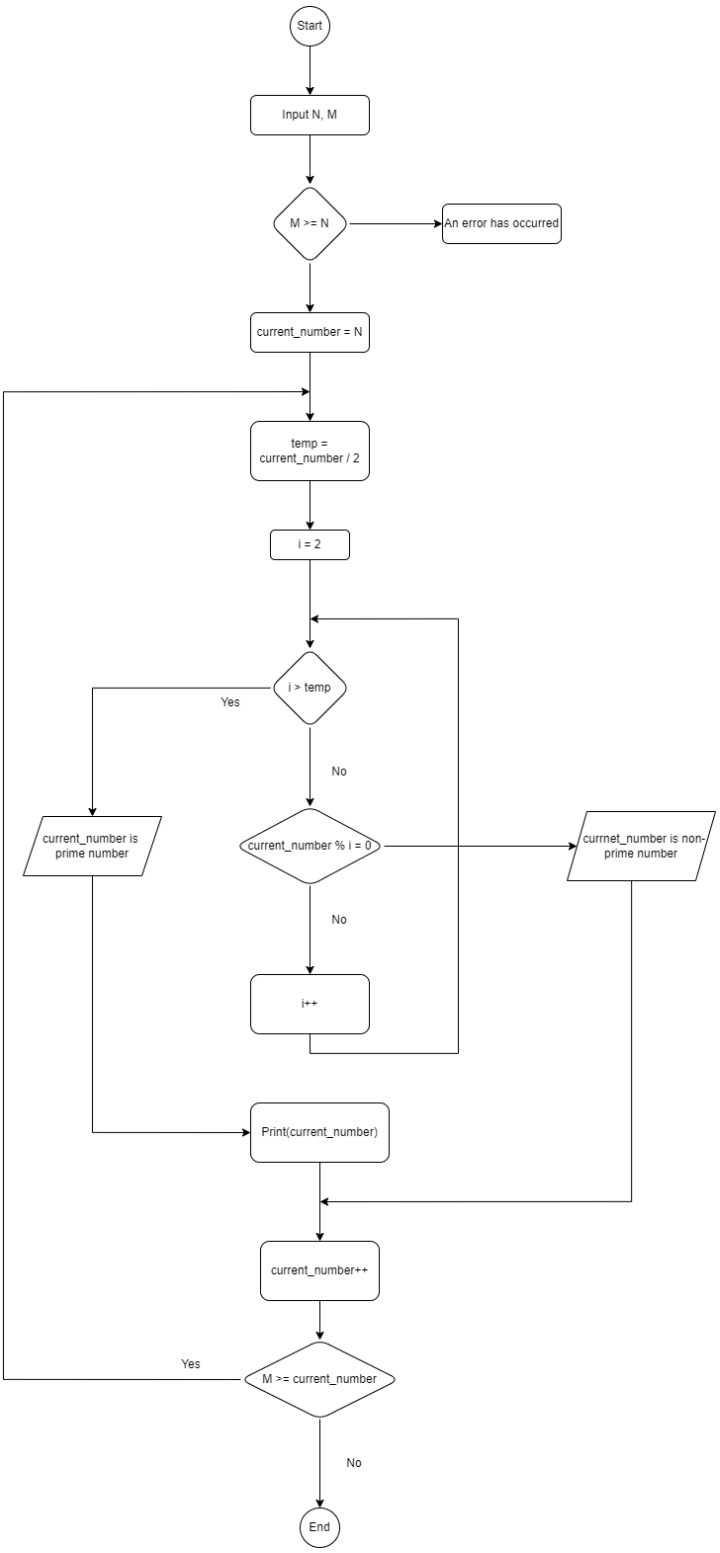
Hoàng Quốc Bảo - 20194484
Vũ Minh Hải - 20194550 **Group 2**

Exercise 2:

Đề bài:

Find all prime numbers (such as 2, 3, 5, 7, etc) in a range from the integer N to the integer M. N, M is entered from the keyboard.

Lưu đồ thuật toán:



Mã nguồn:

```
.data
Message1: .asciiz "Input the integer N:"
Message2: .asciiz "Input the integer M:"
Message3: .asciiz "An error has occurred (M < N)"

.text

check\_inputN:
     addi $v0, $zero, 51   # Read input N
     la $a0, Message1
     syscall
     bne $a1, 0, check\_inputN # if $a1 != 0: an error has occurred, branch to
         check\_inputN
     nop
     add $s0, $zero, $a0   # Store N in $s0

check\_inputM:
     addi $v0, $zero, 51   # Read input N
     la $a0, Message2
     syscall
     bne $a1, 0, check\_inputM # if $a1 != 0: an error has occurred, branch to
         check\_inputM
     nop
     add $s1, $zero, $a0   # Store M in $s1

     slt $t0, $s1, $s0  # if M < N: an error has occurred, end program
     bne $t0, 1, ok
     nop
     addi $v0, $zero, 55
     la $a0, Message3
     syscall
     j done
     nop

ok:
     add $s2, $zero, $s0   # initialize current\_number = N
     slti $t0, $s2, 2   # if current\_number < 2 then current\_number = 2
     bne $t0, 1, main\_loop
     nop
     addi $s2, $zero, 2    # current\_number = 2
main\_loop:
     slt $t0, $s1, $s2  # if current\_number > M then end program
     beq $t0, 1, done
     nop
     jal isprime
     nop
     bne $t1, 1, continue  # continue if current\_number is non-prime
     nop
     jal print\_number  # if current\_number is prime number, then print it
     nop
continue:
```

```
     addi $s2, $s2, 1   # current\_number = current\_number + 1
     j main\_loop



isprime:
push:
     addi $sp, $sp, -12    # adjust the stack pointer
     sw $s0, 8($sp)     # store $s0 (N)
     sw $s1, 4($sp)     # store $s1 (M)
     sw $s2, 0($sp)     # store $s2 (current\_number)

work:
     addi $t1, $zero, 1    # initialize return value = 1 (if $t1 = 1 then
        current\_number is prime number)
     srl $s1, $s2, 1       # temp = current\_number/2
     addi $s3, $zero, 2    # initialize i = 2
loop:
     slt $t0, $s1, $s3  # if i > temp then end procedure
     beq $t0, 1, pop
     nop

     div $s4, $s2, $s3  # $s4 = current\_number / i
     mul $s4, $s4, $s3  # $s4 = $s4 * i
     slt $t0, $s4, $s2  # if ($s4 = current\_number) then current\_number is
        divisible by i
     bne $t0, 1, noprime   # curent\_number is non-prime number
     nop
     addi $s3, $s3, 1   # i = i+1
     j loop
     nop
noprime:
     add $t1, $zero, $zero    # set $v0 = 0, end procedure
pop:
     lw $s2, 0($sp)     # restore $s2 (current\_number)
     lw $s1, 4($sp)     # restore $s1 (M)
     lw $s0, 8($sp)     # restore $s0 (N)
     addi $sp, $sp, 12  # adjust the stack pointer
     jr $ra          # end procedure


print\_number:
     addi $v0, $zero, 1
     add $a0, $zero, $s2
     syscall
     addi $v0, $zero, 11
     li $a0, ' '
     syscall
     jr $ra

done:
```

Giải thích phần chương trình chính:

- Khởi tạo giá trị:
  - Chúng ta khởi tạo 3 Message như hình dưới.

```
.data
Message1: .asciiz "Input the integer N:"
Message2: .asciiz "Input the integer M:"
Message3: .asciiz "An error has occurred (M < N)"
```

- Nhập giá trị input N, M

```
check_inputN:   addi $v0, $zero, 51             # Read input N
                la $a0, Message1
                syscall
                bne $a1, 0, check_inputN        # if $a1 != 0: an error has occurred
                nop
                add $s0, $zero, $a0             # Store N in $s0
```

- Nếu giá trị của N nhập vào không phải là số tự nhiên, thì thanh ghi $a1 $\neq 0$. Lệnh bne sẽ quay lại nhãn *check_ inputN* và bắt người dùng nhập lại N đến khi thoả mãn. *(Tương tự với M)*. Giá trị của N được lưu ở thanh ghi $s0, M ở $s1.
- *Thực hiện chương trình:*
Nhập vào giá trị N = 44, M = 84. Kết quả thực hiện:

$s0	16	0x0000002c
$s1	17	0x00000054

- Kiểm tra điều kiện của M và N

```
                slt $t0, $s1, $s0               # if M < N: an error has occurred, end program
                bne $t0, 1, ok
                nop
                addi $v0, $zero, 55
                la $a0, Message3
                syscall
                j done
                nop
ok:
```

- Lệnh *slt* và *bne* kiểm tra giá trị của M và N, nếu người dùng nhập vào M < N thì câu lệnh *syscall* sẽ gọi đến chức năng **MessageDialog** *($v0 = 55)* để thông báo lỗi (Message3: "An error has occurred (M < N)") và nhảy đến nhãn done, kết thúc chương trình.
- Nếu người dùng nhập vào M $\geq$ N thì lệnh *bne* sẽ nhảy đến nhãn *ok*, chương trình tiếp tục được thực hiện.
- *Thực hiện chương trình:*
Giá trị của thanh ghi $t0 = 0. Chứng tỏ M $\geq$ N, chương trình nhảy đến nhãn *ok*.

$t0	8	0x00000000

- Khởi tạo giá trị *current_ number*

```
add $s2, $zero, $s0          # initialize current_number = N
slti $t0, $s2, 2             # if current_number < 2 then current_number = 2
bne $t0, 1, main_loop
nop
addi $s2, $zero, 2           # current_number = 2
```

- Khối lệnh tiếp theo thực hiện khởi tạo giá trị cho biến *current_number* (Thanh ghi $s2). Đầu tiên gán giá trị N cho *current_number*. Sau đó kiểm tra, nếu giá trị *current_number* < 2 thì gán *current_number* $= 2$.
- Thực hiện chương trình:
Do giá trị N $= 44 > 2$ nên *current_number* $= 2$.

$s2	18	0x0000002c

- Khối lệnh chính của chương trình: *main_loop*

```
main_loop:
            slt $t0, $s1, $s2          # if current_number > M then end program
            beq $t0, 1, done
            nop
            jal isprime
            nop
            bne $t1, 1, continue       # continue if current_number is non-prime
            nop
            jal print_number           # if current_number is prime number, then print it
            nop
continue:   addi $s2, $s2, 1           # current_number = current_number + 1
            j main_loop
```

- Đầu tiên, chương trình sẽ kiểm tra nếu *current_number* $>$ M thì sẽ kết thúc chương trình.
- Nếu *current_number* $\leq$ M thì chương trình sẽ gọi đến hàm *isprime (Chi tiết về hàm isprime ở phần sau)*. Hàm *isprime* trả về giá trị nằm ở thanh ghi $t1. Nếu $t1 $= 1$ thì *current_number* là số nguyên tố, nếu $t1 $= 0$ thì *current_number* không phải là số nguyên tố.
- Nếu $t1 $= 1$ thì chương trình sẽ gọi đến thủ tục *print_number* để thực hiện in giá trị *current_number* ra màn hình.
- *Thực hiện chương trình:*

  * Giá trị hiện tại của *current_number* $= 44$. Thanh ghi $t0 có giá trị $= 0$, vậy *current_number* $\leq$ M.

$t0	8	0x00000000

  * Sau khi gọi đến hàm *isprime*, thanh ghi $pc nhảy đến địa chỉ của hàm *isprime*, thanh ghi $ra lưu giá trị của địa chỉ ngay sau lệnh *jal* trong chương trình chính.

$ra	31	0x00400094
pc		0x004000b4

* Sau khi thực hiện xong hàm *isprime*, do *current_number* = 44, không phải là số nguyên tố nên thanh ghi $t1 = 0 và thanh ghi $pc nhận giá trị mà thanh ghi $ra đã cất giữ trước đó.

$t1	9	0x00000000

$ra	31	0x00400094
pc		0x00400094

- Khối lệnh *continue*

```
continue:       addi $s2, $s2, 1              # current_number = current_number + 1
                j main_loop
```

- Khối lệnh này thực hiện tăng giá trị *current_number* lên 1 đơn vị và quay lại nhãn *main_loop*.

Kết quả thực hiện:
Với input N = 44, M = 84

```
47 53 59 61 67 71 73 79 83
-- program is finished running (dropped off bottom) --
```

Giải thích các thủ tục và hàm:

- Hàm *isprime:*
  Hàm *isprime* trả về giá trị nằm ở thanh ghi $t1. Nếu $t1 = 1 thì *current_number* là số nguyên tố, nếu $t1 = 0 thì *current_number* không phải là số nguyên tố.
  - Khối lệnh *push*:

```
isprime:
push:           addi $sp, $sp, -12          # adjust the stack pointer
                sw $s0, 8($sp)              # store $s0 (N)
                sw $s1, 4($sp)              # store $s1 (M)
                sw $s2, 0($sp)              # store $s2 (current_number)
```

    * Thực hiện lưu lại giá trị của 3 thanh ghi đầu vào ($s0 - N, $s1 - M, $s2 - *current_number*) vào ngăn xếp để cất giữ.
  - Khối lệnh *work*:

```
work:           addi $t1, $zero, 1          # initialize return value = 1
                srl $s1, $s2, 1             # temp = current_number/2
                addi $s3, $zero, 2          # initialize i = 2
```

    * Thực hiện khởi tạo 3 giá trị $t1 = 1 (return number), $s1 = *current_number*/2 (temp), $s3 = 2 (i).
  - Khối lệnh *loop*:

```
loop:
                slt $t0, $s1, $s3           # if i > temp then end procedure
                beq $t0, 1, pop
                nop
                div $s4, $s2, $s3           # $s4 = current_number / i
                mul $s4, $s4, $s3           # $s4 = $s4 * i
                slt $t0, $s4, $s2           # if ($s4 = current_number) then current_num
                bne $t0, 1, noprime         # curent_number is non-prime number
                nop
                addi $s3, $s3, 1            # i = i+1
                j loop
                nop
```

    * Đầu tiên, chương trình sẽ kiểm tra nếu i > temp thì sẽ nhảy đến nhãn *pop* để kết thúc hàm.
    * Thực hiện cho i chạy từ 2 -> *current_number*/2. Nếu *current_number* chia hết cho i thì chứng tỏ *current_number* không phải là số nguyên tố.
    * Tìm số dư của *current_number* / i bằng cách chia rồi lấy thương (giá trị nguyên) sau đó đem nhân lại với i. Nếu giá trị vừa nhân lên = *current_number* thì chứng tỏ *current_number* chia hết cho i, chương trình sẽ nhảy đến nhãn *noprime*. Nếu *current_number* không chia hết cho i thì tăng i lên 1 đơn vị rồi quay lại nhãn *loop*.
  - Khối lệnh *noprime* và *pop*:

```
noprime:
                add $t1, $zero, $zero       # set $t1 = 0, end procedure
pop:            lw $s2, 0($sp)              # restore $s2 (current_number)
                lw $s1, 4($sp)              # restore $s1 (M)
                lw $s0, 8($sp)              # restore $s0 (N)
                addi $sp, $sp, 12           # adjust the stack pointer
                jr $ra                      # end procedure
```

* Khối lệnh *noprime* sẽ thực hiện gán giá trị trả về $t1 = 0.
* Khối lệnh *pop* thực hiện trả lại giá trị ban đầu cho các biến đã lưu trong ngăn xếp và giải phóng ngăn xếp, gọi lệnh *jr* để quay về chương trình chính.

- Hàm *print_number*
  Hàm *print_number in ra số nguyên tố được lưu ở thanh ghi $s2*

```
print_number:
                addi $v0, $zero, 1
                add $a0, $zero, $s2
                syscall
                addi $v0, $zero, 11
                li $a0, ' '
                syscall
                jr $ra
done:
```

  – Đầu tiên, hàm sẽ gọi đến chức năng **print decimal integer** ($v0 = 1) để in ra số nguyên tố được lưu trong thanh ghi $s2.
  – Sau đó, hàm sẽ gọi đến chức năng **print character** ($v0 = 11) để in ra kí tự *space*, giúp ngăn cách các số nguyên tố khi liệt kê.