

Báo cáo Thực hành Kiến trúc máy tính
MidTerm

Họ và tên: Ôn Quang Tùng

MSSV:20226096

Assignment 1 (A15)

```
# 20226096_OnQuangTung_A15
```

```
.data
```

```
    string_inputM: .ascii "Nhap vao so M: "  
    string_inputN: .ascii "Nhap vao so N: "  
    string_inputQ: .ascii "Nhap vao so Q: "  
    space: .ascii " "
```

```
.text
```

```
inputM:
```

```
    li $v0, 4  
    la $a0, string_inputM  
    syscall  
    li $v0, 5  
    syscall  
    bltz $v0, inputM # if M < 0 -> re-input M  
    move $s0, $v0 # $s0 = M
```

```
inputN:
```

```
    li $v0, 4  
    la $a0, string_inputN  
    syscall  
    li $v0, 5  
    syscall  
    bltz $v0, inputN # if N < 0 -> re-input N  
    move $s1, $v0 # $s1 = N  
    blt $s1, $s0, inputN # if N < M -> re-input N  
    mul $s3, $s0, $s1 # $s3 = M*N
```

```
inputQ:
```

```
    li $v0, 4  
    la $a0, string_inputQ  
    syscall  
    li $v0, 5  
    syscall  
    move $s2, $v0 # $s2 = Q
```

```

        blt $s2, $s3, inputQ # if Q < M*N -> re-inputQ
        li $t5, 0 # initialize the number of divisor = 0

move $t0, $s0 # initialize i = M
loop:
    div $s2, $t0
    mfhi $t1 # $t1 = Q%i
    jal check
    addi $t0, $t0, 1
    ble $t0, $s1, loop # loop while i <= N

end:
    beqz $t5, no_ans # if there's no divisor -> print -1
    li $v0, 10
    syscall

check:
    bne $t1, $0, return
    li $v0, 1
    move $a0, $t0 # print i
    syscall
    li $v0, 4
    la $a0, space # print " "
    syscall
    addi $t5, $t5, 1 # update the number of divisor

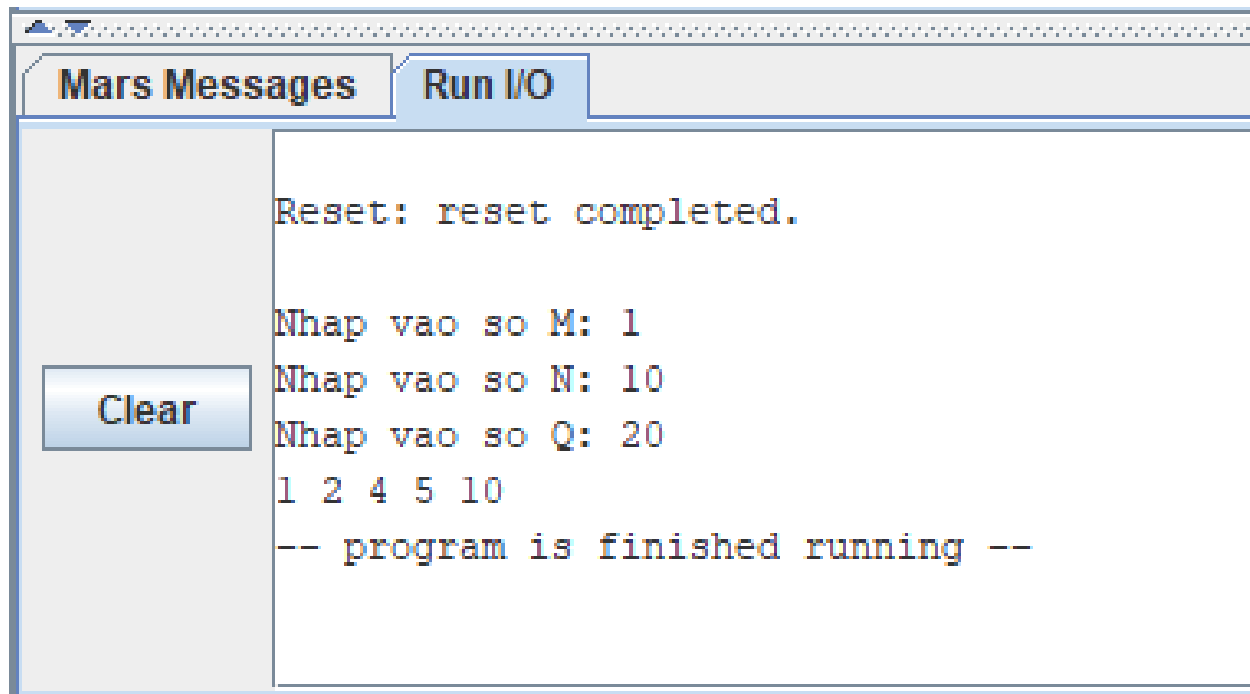
return: jr $ra

no_ans:
    li $v0, 1
    li $a0, -1
    syscall

```

- Hàm loop:
 - Thực hiện phép Q/i và lưu số dư vào thanh ghi \$t1
 - Sau đó thực hiện nhảy sang nhãn check
 - Sau khi thực hiện xong nhãn check, index sẽ tăng 1 và lặp lại chừng nào $index \leq N$
- Hàm check:

- Thực hiện return để thực hiện tiếp hàm loop nếu số dư khác 0
- Nếu số dư = 0 thì thực hiện in i ra màn hình
- Sau đó update rằng đã có ước(nếu có) để không chạy nữa no_ans
- Hàm end:
 - Kiểm tra xem đã có ước chưa để thực hiện nữa no_ans
 - Kết thúc chương trình
- Hàm no_ans:
 - In ra -1 nếu không có đáp án
- Kết quả sau khi thực hiện chương trình:



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

```
Reset: reset completed.  
  
Nhap vao so M: 1  
Nhap vao so N: 10  
Nhap vao so Q: 20  
1 2 4 5 10  
-- program is finished running --
```

On the left side of the window, there is a "Clear" button.

Assignment 2 (B2)

```
# 20226096_OnQuangTung_B2
.data
A: .space 1000
n_input: .asciiz "Nhap n: "
space: .asciiz " "
res: .asciiz "Cap phan tu lien ke co tich lon nhat la: "

.text
li $v0, 4
la $a0, n_input
syscall

li $v0, 5 # get n
syscall
move $s0, $v0 # $s0 = n

la $s1, A # A[] = $s1
li $t0, 0 # i = 0
loop_cin: #for(int i =0; i<n; i++) cin>>a[i]
    li $v0, 5
    syscall
    sw $v0, 0($s1)
    addi $t0, $t0, 1
    addi $s1, $s1, 4
    blt $t0, $s0, loop_cin

la $s1, A # A[] = $s1
lw $t2, 0($s1) # $t2 = A[i-1]
addi $s1, $s1, 4
lw $t3, 0($s1) # $t3 = A[i+1]
li $t0, 1 # i = 1
li $s7, -999999 # $s7 = max_mul
```

```

loop:
    mul $t1, $t2, $t3 # $t1 = A[i-1] * A[i] (no overflow)
    jal check
    addi $t0, $t0, 1
    lw $t2, 0($s1) # $t2 = A[i-1]
    addi $s1, $s1, 4
    lw $t3, 0($s1) # $t3 = A[i+1]
    blt $t0, $s0, loop

```

```

print:
    li $v0, 4
    la $a0, res
    syscall
    li $v0, 1
    move $a0, $s2 # print A[i-1]
    syscall
    li $v0, 4
    la $a0, space
    syscall
    li $v0, 1
    move $a0, $s3 # print A[i-1]
    syscall

```

```

exit:
    li $v0, 10
    syscall

```

```

check:
    blt $t1, $s7, return # if A[i-1] * A[i] < max_mul -> return
    move $s7, $t1 # update max_mul
    move $s2, $t2 # $s2 = A[i-1] with current largest mul
    move $s3, $t3 # $s3 = A[i] with current largest mul

```

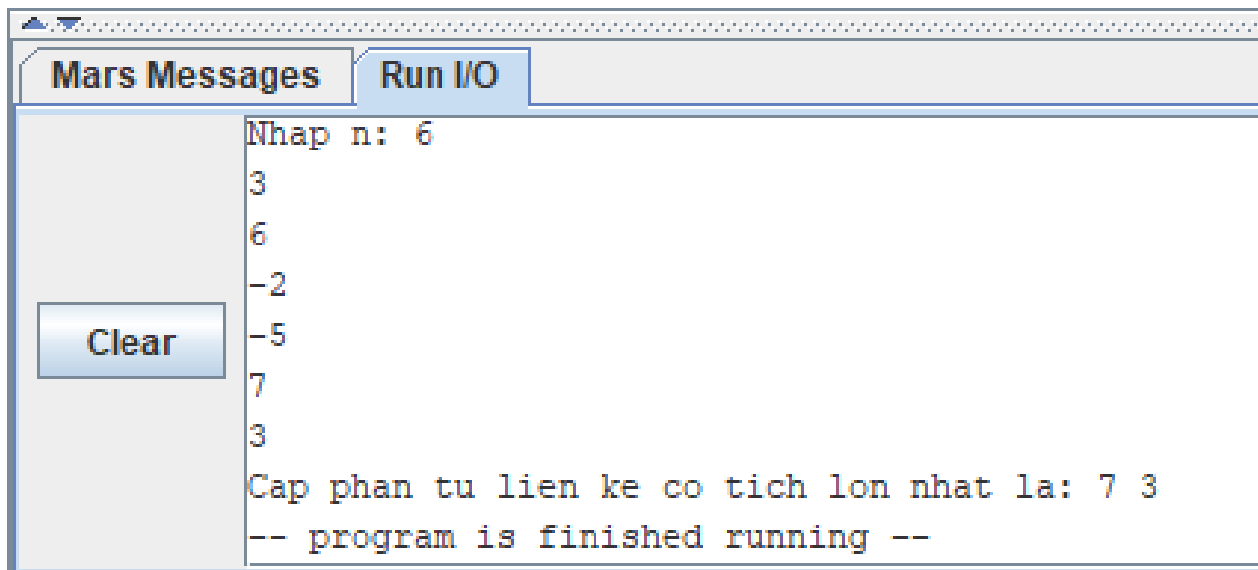
```

return: jr $ra

```

- Hàm loop:
 - Thực hiện phép nhân(không overflow) và lưu nó vào thanh ghi \$t1

- Thực hiện nhảy sang nhãn check
- Sau khi thực hiện xong hàm check ta tăng index, cập nhật giá trị của các phần tử liên kế cần được xét mới
- Hàm sẽ được lặp lại chừng nào còn $\text{index} < n$
- Hàm check:
 - Return để thực hiện tiếp hàm loop nếu tích 2 phần tử liên nhau $< \text{max_mul}$ hiện tại
 - Nếu không, cập nhật max_mul và giá trị 2 phần tử liên kế để thực hiện in kết quả sau khi xét hết các trường hợp
- Kết quả sau khi thực hiện chương trình:



```
Nhap n: 6
3
6
-2
-5
7
3
Cap phan tu lien ke co tich lon nhat la: 7 3
-- program is finished running --
```

Assignment 3 (C6)

```
# 20226096_OnQuangTung_C6
.data
s: .space 100
message1: .asciiz "Nhap string: "
message2: .asciiz "Nhap char: "
message3: .asciiz "\nSo lan suat hien cua no la: "
newline: .ascii "\n"

.text

#----- input -----
li $v0, 4
la $a0, message1 # Nhap string
syscall
li $v0, 8
la $a0, s
li $a1, 1000
syscall
# Change uppercase to lowercase
la $s0, s
la $t7, newline
lb $t7, 0($t7)
UtoL:
lb $t2, 0($s0) # $t2 = s[i]
beq $t2, $t7, end_UtoL # end loop if s[i] == "\n"
andi $t2, $t2, 0xFFDF # Change uppercase to lowercase
sb $t2, 0($s0)
addi $s0, $s0, 1 # next char
j UtoL
end_UtoL:

li $v0, 4
la $a0, message2 # Nhap char
syscall
```



```

li $v0, 12
syscall
andi $s2, $v0, 0xFFDF # Change lowercase to uppercase and store at $s2
                        # 0xFFDF = 11111111101111
                        # <-> +32 Ascii

#----- traversal -----
li $s7, 0 # result
la $s0, s
la $t7, endlene
lb $t7, 0($t7)
loop:
lb $t2, 0($s0) # $t2 = s[i]
beq $t2, $t7, endloop # end loop if s[i] == "\n"
jal check
addi $s0, $s0, 1 # next char
j loop
endloop:

print:
li $v0, 4
la $a0, message3
syscall
li $v0, 1
move $a0, $s7 # print result
syscall

exit:
li $v0, 10
syscall

check:
beq $t2, $s2, update # update if current char = input char
j return

update:

```

```
addi $s7, $s7, 1
```

```
return:
```

```
jr $ra
```

- Trong phần Input:
 - Ta nhập string, ngay sau đó, thực hiện duyệt từng phần tử để chuyển tất cả các chữ cái trong string thành chữ hoa bằng phép toán andi với 0xFFDF (11111111101111). Phép toán này tương đương với việc -32 theo mã ascii nhưng lại giảm việc phải so sánh -> hiệu quả cao
 - Ta nhập kí tự cần đếm tần suất, và tương tự trên, chuyển kí tự này thành chữ hoa(nếu là chữ thường)
- Nhãn loop:
 - Duyệt từng phần tử để thực hiện check xem đó có phải kí tự đã nhập vào không
 - Hàm sẽ được lặp lại đến khi gặp kí tự “\n”
- Nhãn check:
 - Nhảy xuống nhãn update nếu kí tự đang xét hiện tại bằng với kí tự nhập vào
 - Nếu không thì return để thực hiện tiếp hàm loop
- Nhãn update:
 - Cập nhật biến đếm thêm 1 đơn vị. Biến đếm này sau sẽ được gọi lại để in ra
- Kết quả sau khi thực hiện chương trình:

