

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Kiểm tra giữa kỳ

Họ tên	MSSV
Phạm Minh Hiền	20235705

Assignment A - 11:

Nhập số nguyên dương N từ bàn phím, in ra màn hình tổng các chữ số là số lẻ và tổng các chữ số là số chẵn của N.

Source Code :

```
.data
```

```
msg: .asciz "Nhap so nguyen duong N: "  
odd: .asciz "Tong cac chu so la so le: "  
even: .asciz "Tong cac chu so la so chan: "  
newline: .asciz "\n"  
error_msg: .asciz "So phai la nguyen duong !!!"
```

```
.text
```

```
main:
```

```
la a0, msg # Thông báo nhập số  
li a7, 4  
ecall
```

```
li a7, 5 # Nhập số nguyên dương N  
ecall
```

```
add t0, a0, zero # Lưu N vào t0
```

```
# Kiểm tra số nhập vào có nguyên dương hay không  
beq t0, zero, error
```

blt t0, zero, error

li t1, 0 # Khoi tao tong chan

li t2, 0 # Khoi tao tong le

digit:

beq t0, zero, end # Neu N = 0 thi ket thuc

li t3, 10

rem t4, t0, t3 # N % 10 de lay chu so cuoi

div t0, t0, t3 # N / 10 de bo chu so cuoi

andi t5, t4, 1 # t5 = 1 neu le va = 0 neu chan

beq t5, zero, even_digit

odd_digit:

add t1, t1, t4

j digit

even_digit:

add t2, t2, t4

j digit

end:

In tong so le

la a0, odd

li a7, 4

ecall

add a0, t1, zero

li a7, 1

ecall

la a0, newline

li a7, 4

ecall

In tong chan

la a0, even

li a7, 4

ecall

add a0, t2, zero

li a7, 1

ecall

la a0, newline

li a7, 4

ecall

Ket thuc

li a7, 10

ecall

error:

la a0, error_msg # Thong bao loi

li a7, 4

ecall

- Phân tích cách thực hiện :
 - + Yêu cầu người dùng nhập vào một số nguyên dương N, sau đó tính tổng các chữ số lẻ và tổng các chữ số chẵn trong số đó, in kết quả ra màn hình. Nếu người dùng nhập sai (N không phải số nguyên dương), chương trình sẽ báo lỗi.

+ Trong *main*, sử dụng 2 lệnh *ecall* để thông báo và cho phép người dùng nhập số nguyên dương N, sau đó lưu N vào *t0*. Kế tiếp, ta lần lượt kiểm tra số N có phải nguyên dương hay không thông qua 2 câu lệnh so sánh **beq** để kiểm tra số 0 và **blt** để kiểm tra số âm, nếu không phải nguyên dương thì ngay lập tức chương trình nhảy đến nhãn *error* và thực hiện báo lỗi kết thúc chương trình. Nếu thỏa mãn N là nguyên dương thì khởi tạo *t1*, *t2* lần lượt là tổng của các chữ số chẵn và lẻ với giá trị hiện thời là 0.

+ *digit*: Vòng lặp để xử lý từng chữ số

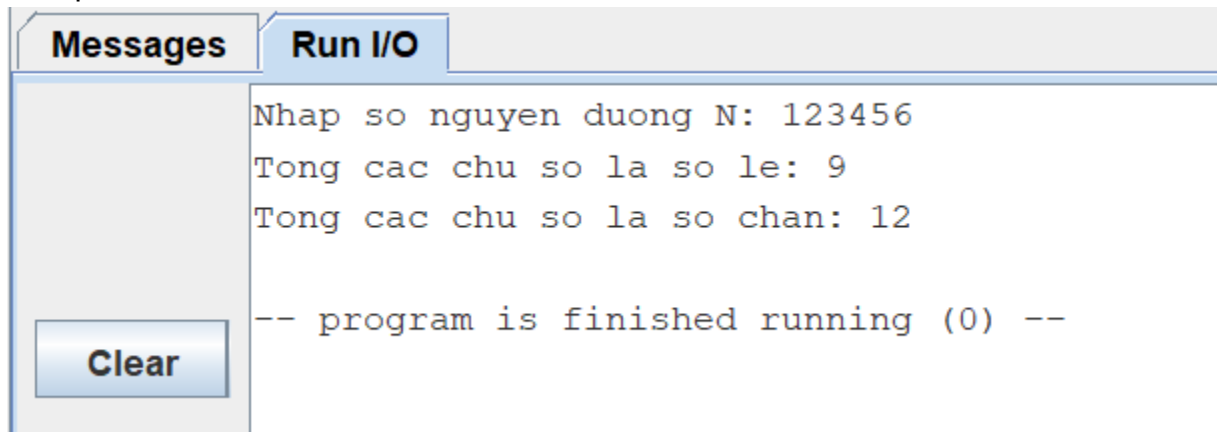
- Khi *t0* = 0, tức là không còn chữ số nào, kết thúc lặp.
- Tách từng chữ số từ trái qua phải thông qua chia lấy dư **rem** và chia **div**.
- Kiểm tra từng chữ số là chẵn hay lẻ: **andi** với 1, nếu *t5* = 1 thì lẻ, *t5* = 0 thì chẵn.
- Nếu chẵn thì nhảy đến nhãn *even_digit*, lẻ thì thực hiện tiếp *odd_digit*.

+ *even_digit* và *odd_digit*: Cộng dồn chữ số vào tổng tương ứng và quay lại vòng lặp.

+ *end*: In kết quả.

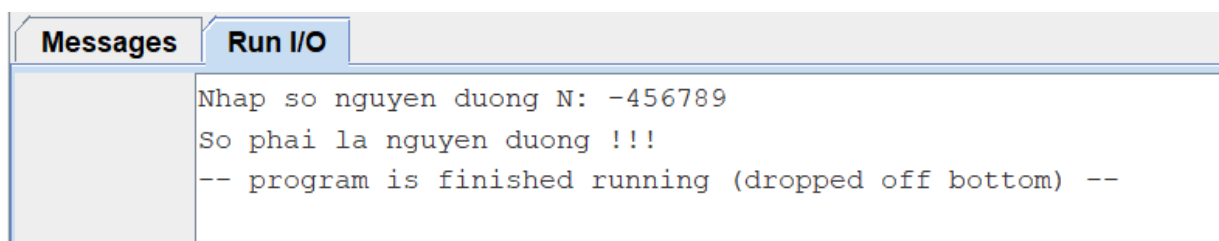
+ *error*: Thông báo lỗi.

- Kết quả:



```
Messages Run I/O
Nhap so nguyen duong N: 123456
Tong cac chu so la so le: 9
Tong cac chu so la so chan: 12

-- program is finished running (0) --
```



```
Messages Run I/O
Nhap so nguyen duong N: -456789
So phai la nguyen duong !!!
-- program is finished running (dropped off bottom) --
```

Messages	Run I/O
	Nhap so nguyen duong N: 0 So phai la nguyen duong !!! -- program is finished running (dropped off bottom) --

➔ Chương trình có vẻ thực hiện đúng.

Assignment B - 15:

Nhập mảng số nguyên từ bàn phím. In ra màn hình phần tử có số lần xuất hiện ít nhất trong mảng.

Source Code :

.data

```
array: .space 400 # Cho phép nhập 100 số nguyên
count: .space 400 # Lưu số lần xuất hiện
size: .asciz "Nhập kích thước mảng: "
number: .asciz "Nhập phần tử thứ "
colon: .asciz ": "
result: .asciz "Phần tử xuất hiện ít nhất: "
fre: .asciz "Số lần xuất hiện: "
error_msg: .asciz "Kích thước mảng phải > 0 hoặc < 100"
newline: .asciz "\n"
```

.text

main:

```
la a0, size # Thông báo nhập kích thước
```

```
li a7, 4
```

```
ecall
```

```
li a7, 5 # Nhập kích thước từ bàn phím
```

```
ecall
```

```
add t0, a0, zero # Lưu kích thước vào t0
```

```
ble t0, zero, error # Kiểm tra kích thước nếu ≤ 0 thì báo lỗi
```

li t1, 100

bgt t0, t1, error # Kich thuoc > 100 thi cung bao loi

la t1, array # Dia chi mang

li t2, 0 # Bien dem

input:

beq t2, t0, end_input # Neu nhap du phan tu roi thi bat dau xu ly

Nhap phan tu thu i

la a0, number

li a7, 4

ecall

add a0, zero, t2

li a7, 1

ecall

la a0, colon

li a7, 4

ecall

li a7, 5

ecall

sw a0, 0(t1)

addi t1, t1, 4 # Tang dia chi

addi t2, t2, 1 # i++

j input

end_input:

la t1, count # Khoi tao mang dem

li t2, 0

init:

Dam bao khoi tao 100 phan tu

li s9, 100

beq t2, s9, count_fre

sw zero, 0(t1)

addi t1, t1, 4

addi t2, t2, 1

j init

count_fre:

la t1, array # Dia chi mang

li t2, 0 # Bien dem

loop:

beq t2, t0, min # Lap den het kich thuoc mang thi tim min

lw t3, 0(t1) # t3 = array[i]

la t4, count

slli t5, t3, 2 # Tinh dia chi

add t4, t4, t5 # t4 = &counts[array[i]]

lw t6, 0(t4) # t6 = counts[array[i]]

addi t6, t6, 1 # counts[array[i]]++

sw t6, 0(t4) # Luu gia tri moi vao count[array[i]]

addi t1, t1, 4

addi t2, t2, 1

j loop

min:

li t2, 101 # t2 = min_freq

li t3, -1 # t3 = min_value (phan tu co min_freq)


```

    la t4, count
    li t5, 0
find:
    beq t5, s9, print
    lw t6, 0(t4)
    beq t6, zero, next # Bo qua neu counts[i] = 0
    bge t6, t2, next # Bo qua neu counts[i] >= min_freq
    add t2, t6, zero
    add t3, t5, zero
next:
    addi t4, t4, 4
    addi t5, t5, 1
    j find
print:
    la a0, result
    li a7, 4
    ecall
    add a0, t3, zero
    li a7, 1
    ecall
    la a0, newline
    li a7, 4
    ecall

    la a0, fre
    li a7, 4
    ecall

```

```
add a0, t2, zero
```

```
li a7, 1
```

```
ecall
```

```
j exit
```

error:

```
la a0, error_msg
```

```
li a7, 4
```

```
ecall
```

exit:

```
li a7, 10
```

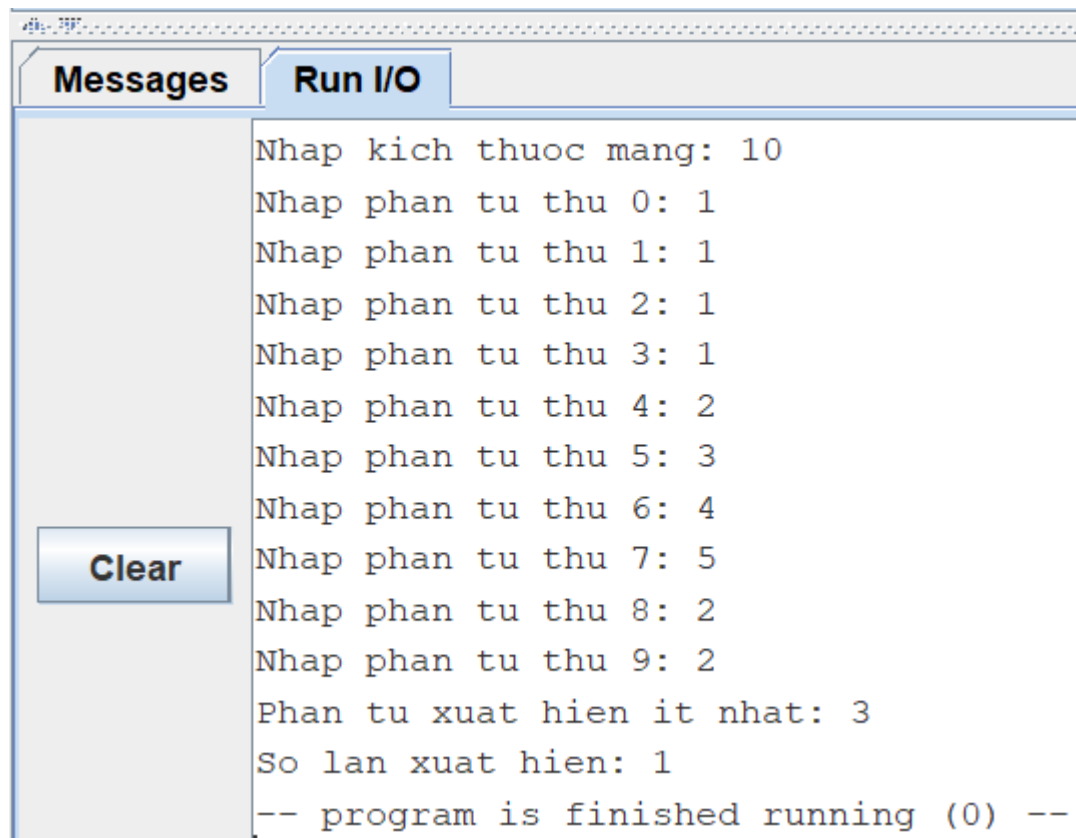
```
ecall
```

- Phân tích cách thực hiện:
 - + Chương trình cho phép người dùng nhập mảng với kích thước n quy định ở đây tối đa là 100, nếu kích thước không hợp lệ thì sẽ báo lỗi. Nếu kích thước hợp lệ người dùng có thể nhập n phần tử. Chương trình sẽ tạo mảng `count` để lưu số lần xuất hiện của từng phần tử (giá trị phần tử là chỉ số). Tìm phần tử có số lần xuất hiện ít nhất và in ra kết quả.
 - + *main*: Cho phép nhập kích thước mảng n và kiểm tra tính hợp lệ của n thông qua lệnh **ble** với trường hợp $n \leq 0$ và **bgt** với $n > 100$. Nếu hợp lệ thì lưu địa chỉ mảng vào `t1` và khởi tạo biến đếm `t2`. Nếu không hợp lệ thì nhảy đến nhãn *error* và báo lỗi kết thúc chương trình.
 - + *input*: Nhập từng phần tử của mảng
 - `t1` dùng làm con trỏ để ghi dữ liệu vào array.
 - `t2` dùng làm biến đếm số lượng phần tử đã nhập.
 - + *init*: Khởi tạo mảng `count` về 0.
 - + *loop*: Đếm số lần xuất hiện của từng phần tử
 - Dùng giá trị của phần tử làm chỉ số trong mảng `count`.
 - Tăng giá trị tương ứng để theo dõi số lần xuất hiện.

+ *min* và *find* : Tìm phần tử xuất hiện ít nhất.

+ *error*: Thông báo lỗi.

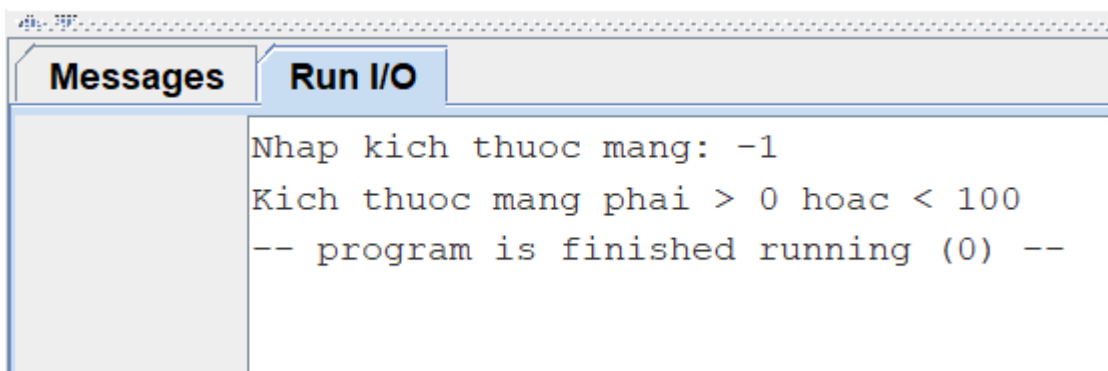
- Kết quả:



The screenshot shows a window with two tabs: "Messages" and "Run I/O". The "Run I/O" tab is active, displaying the following text:

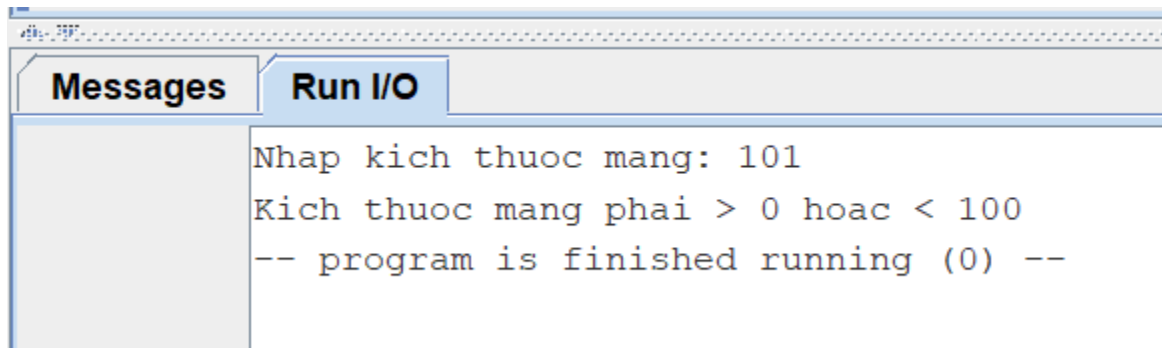
```
Nhap kích thước mảng: 10
Nhap phần tử thu 0: 1
Nhap phần tử thu 1: 1
Nhap phần tử thu 2: 1
Nhap phần tử thu 3: 1
Nhap phần tử thu 4: 2
Nhap phần tử thu 5: 3
Nhap phần tử thu 6: 4
Nhap phần tử thu 7: 5
Nhap phần tử thu 8: 2
Nhap phần tử thu 9: 2
Phần tử xuất hiện ít nhất: 3
Số lần xuất hiện: 1
-- program is finished running (0) --
```

On the left side of the "Run I/O" tab, there is a "Clear" button.



The screenshot shows a window with two tabs: "Messages" and "Run I/O". The "Run I/O" tab is active, displaying the following text:

```
Nhap kích thước mảng: -1
Kích thước mảng phải > 0 hoặc < 100
-- program is finished running (0) --
```



The screenshot shows a window with two tabs: "Messages" and "Run I/O". The "Run I/O" tab is active, displaying the following text:

```
Nhap kích thước mảng: 101
Kích thước mảng phải > 0 hoặc < 100
-- program is finished running (0) --
```

→ Chương trình có vẻ thực hiện chính xác yêu cầu.

Assignment C - 17:

Nhập vào 2 chuỗi ký tự A và B, kiểm tra xem A và B có phải là anagram của nhau hay không? (2 chuỗi ký tự được gọi là anagram nếu cùng chứa các ký tự giống nhau với số lần xuất hiện như nhau, nhưng thứ tự xuất hiện khác nhau, ví dụ “silent” và “listen”).

Source Code :

.data

```
strA: .space 100 # Chuoi A toi da 100 ky tu
strB: .space 100 # Chuoi B toi da 100 ky tu
freA: .space 128 # Bang dem tan suat ky tu
freB: .space 128

inputA: .asciz "Nhap chuoi A: "
inputB: .asciz "Nhap chuoi B: "
newline: .asciz "\n"
yes: .asciz "A va B la anagram."
no: .asciz "A va B khong la anagram."
```

.text

main:

```
la a0, inputA # Nhap chuoi A
li a7, 4
ecall

la a0, strA
li a1, 100
li a7, 8
ecall
```

la a0, inputB # Nhap chuoi B

li a7, 4

ecall

la a0, strB

li a1, 100

li a7, 8

ecall

Tinh freA

la a0, strA

la a1, freA

jal count

Tinh freB

la a0, strB

la a1, freB

jal count

So sanh tan suat

la a0, freA

la a1, freB

jal compare

li a0, 1

beq a0, a2, print_yes

print_no:

la a0, no

li a7, 4

ecall

j exit

print_yes:

la a0, yes

li a7, 4

ecall

exit:

li a7, 10

ecall

count:

lb t0, 0(a0) # Lay tung ky tu trong chuoi

beq t0, zero, end

add t3, a1, t0 # Tinh dia chi cua fre[t0]

lb t2, 0(t3) # Lay gia tri hien tai trong fre[t0]

addi t2, t2, 1 # Tang gia tri them 1

sb t2, 0(t3) # Cap nhat gia tri t3

addi a0, a0, 1 # Ky tu tiep theo

j count

end:

jr ra

compare:

li t0, 0 # i

loop:

```

li s9, 128 # 128 ki tu
bge t0, s9, fre_equal
add t3, a0, t0 # Dia chi freA[t0]
add t4, a1, t0 # Dia chi freB[t0]
lb t1, 0(t3) # Lay gia tri
lb t2, 0(t4)
bne t1, t2, fre_not_equal
addi t0, t0, 1
j loop
fre_equal:
    li a2, 1
    jr ra
fre_not_equal:
    li a2, 0
    jr ra

```

- Phân tích cách thực hiện:

+ Chương trình cho phép người dùng nhập hai chuỗi A và B, sau đó đếm tần suất xuất hiện của từng ký tự trong chuỗi A và B. So sánh tần suất của hai chuỗi, nếu giống nhau thì thông báo A và B là anagram, ngược lại thì không phải là anagram.

+ *main*: Cho phép nhập 2 chuỗi A và B lần lượt với tối đa 100 ký tự. Sau đó gọi chương trình con *count* để đếm tần suất ký tự cho A và B, tần suất lưu bằng byte, mỗi chỉ số ứng với mã ASCII của ký tự. So sánh hai bảng tần suất bằng *compare*, nếu bằng nhau trả về $a2 = 1$, khác nhau $a2 = 0$.

+ *count*: Đếm tần suất ký tự trong chuỗi

- Với mỗi ký tự trong chuỗi, tăng giá trị tương ứng tại `fre[char]`.
- Sử dụng giá trị mã ASCII làm chỉ số.
- Giới hạn 128 ký tự đầu tiên trong bảng ASCII.

+ *compare*: So sánh `freA[i]` với `freB[i]` cho `i` từ 0 đến 127.

+ Lưu ý: Không bỏ qua khoảng trắng, chữ hoa/thường (Ví dụ: Silent không là anagram của Listen).

- Kết quả:

```
Messages Run I/O
Nhap chuoi A: silent
Nhap chuoi B: listen
A va B la anagram.
-- program is finished running (0) --
```

```
Messages Run I/O
Nhap chuoi A: silence
Nhap chuoi B: listener
A va B khong la anagram.
-- program is finished running (0) --
```

```
Messages Run I/O
Nhap chuoi A: Silent
Nhap chuoi B: Listen
A va B khong la anagram.
-- program is finished running (0) --
```

➔ Có vẻ chương trình hoạt động như mong muốn.