**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**Kiểm tra giữa kỳ**

|  |  |
| --- | --- |
| **Họ tên** | **MSSV** |
| Phạm Minh Hiển | 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 # Thong bao nhap so

li a7, 4

ecall

li a7, 5 # Nhap so nguyen duong N

ecall

add t0, a0, zero # Luu N vao t0

# Kiem tra so nhap vao co nguyen duong hay khong

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 thoả 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:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* 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 phep nhap 100 so nguyen

count: .space 400 # Luu so lan xuat hien

size: .asciz "Nhap kich thuoc mang: "

number: .asciz "Nhap phan tu thu "

colon: .asciz ": "

result: .asciz "Phan tu xuat hien it nhat: "

fre: .asciz "So lan xuat hien: "

error\_msg: .asciz "Kich thuoc mang phai > 0 hoac < 100"

newline: .asciz "\n"

.text

main:

la a0, size # Thong bao nhap kich thuoc

li a7, 4

ecall

li a7, 5 # Nhap kich thuoc tu ban phim

ecall

add t0, a0, zero # Luu kich thuoc vao t0

ble t0, zero, error # Kiem tra kich thuoc neu <= 0 thi bao loi

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 <= 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:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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

**Assignment C - 17:**

Nhập vào 2 xâu ký tự A và B, kiểm tra xem A và B có phải là anagram của nhau hay không? (2 xâu 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:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Có vẻ chương trình hoạt động như mong muốn.