

THỰC THÀNH KIẾN TRÚC MÁY TÍNH

BÙI QUANG HƯNG – 20225849

KIỂM TRA GIỮA KÌ

Bài 1(A3): Nhập số nguyên dương N từ bàn phím, in ra các số nguyên tố nhỏ hơn N.

Bài làm

- Mã nguồn:

.data

mess1: .ascii "Nhap so nguyen duong N: "

mess2: .ascii "Cac so nguyen to nho hon "

newline: .ascii "\n"

space: .ascii " "

mess3: .ascii "Khong co so nt nao"

.text

.globl main

main:

In ra man hinh mess1

li \$v0, 4

la \$a0, mess1

syscall

Doc N tu ban phim

li \$v0, 5

syscall

move \$t0, \$v0 # Luu N vao \$t0

Kiem tra xem N < 2

ble \$t0, 2, exit_program

```
# In ra man hinh mess2
```

```
li $v0, 4
```

```
la $a0, mess2
```

```
syscall
```

```
move $a0, $t0 # Truyen N nhu mot doi so
```

```
li $v0, 1
```

```
syscall
```

```
# In ki tu xuong dong \n
```

```
li $v0, 4
```

```
la $a0, newline
```

```
syscall
```

```
li $v0, 1
```

```
li $a0, 2
```

```
syscall
```

```
li $v0, 4
```

```
la $a0, space
```

```
syscall
```

```
# Khoi tao bien dem i bat dau tu 2
```

```
li $t1, 2
```

```
loop:
```

```
# Kiem tra xem i co phai so nguyen to khong
```

```
move $a0, $t1 # Truyen i nhu mot doi so
```

```
jal isPrime
```

```
# Neu $s3 == 1 (la so nguyen to), in ra i
```

```
beq $s3, 1, print_prime
```

```
# Tang i len 1: i=i+1
```

```
addi $t1, $t1, 1
```

```
# Kiem tra xem i < N
```

```
blt $t1, $t0, loop
```

```
# Thoat khoi vong lap neu i >= N
```

```
j exit
```

```
print_prime:
```

```
# In ra i
```

```
li $v0, 1
```

```
move $a0, $t1
```

```
syscall
```

```
# In ra ki tu khoang trang
```

```
li $v0, 4
```

```
la $a0, space
```

```
syscall
```

```
# Tang i len 1 : i=i+1
```

```
addi $t1, $t1, 1
```

```
# Kiểm tra xem i < N
```

```
blt $t1, $t0, loop
```

exit:

li \$v0, 10 #Ket thuc chuong trinh

syscall

exit_program:

#In ra man hinh mess3

li \$v0, 4

la \$a0, mess3

syscall

#Ket thuc chuong trinh

li \$v0, 10

syscall

Hàm kiểm tra xem một số có phải là số nguyên tố hay không

isPrime:

sw \$ra, 0(\$sp) # Luu dia chi tra ve

li \$t2, 2 #Khoi tao so chia la 2

ble \$a0, \$t2, not_prime # Neu so nho hon 2, tra ve 0

Kiem tra tinh chia het

chiahet:

div \$a0, \$t2

mfhi \$t3

beq \$t3, \$zero, not_prime

Tang so chia len 1

addi \$t2, \$t2, 1

```
#Kiem tra xem ($t2)^2 > i không
```

```
mul $t4, $t2, $t2
```

```
bgt $t4, $a0, prime
```

```
# Neu ($t2)^2 <= i, tiep tục kiểm tra tính chia hết
```

```
j chiahet
```

```
prime:
```

```
# là số nguyên tố, return 1
```

```
li $s3, 1
```

```
j end
```

```
not_prime:
```

```
# không phải số nt, return 0
```

```
li $s3, 0
```

```
end:
```

```
# Khởi phục địa chỉ và trả về
```

```
lw $ra, 0($sp)
```

```
jr $ra
```

- Kết quả:

```

Nhập số nguyên dương N: 100
Các số nguyên tố nhỏ hơn 100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
-- program is finished running --

Nhập số nguyên dương N: 2
Không có số nt nào
-- program is finished running --
```

- Phân tích cách thực hiện:

+ Sau khi nhập N vào cần kiểm tra xem N có lớn hơn 2 hay không, nếu có thì tiếp tục còn không thì in ra màn hình “Không có số nguyên tố nào thỏa mãn”

+ Chạy vòng lặp với biến đếm i bắt đầu từ 2 chạy tới N-1

+ Kiểm tra xem i có phải là một số nguyên tố hay không, nếu có thì in ra màn hình, nếu không thì tiếp tục duyệt.

+ Tăng biến đếm i lên 1 để tiếp tục chạy vòng lặp cho đến khi nào $i = N$ thì vòng lặp kết thúc.

- Ý nghĩa các hàm con:

+ Hàm `print_prime` để in ra các số nguyên tố $< N$

+ Hàm `is_Prime`:

/Hàm này nhận một số trong thanh ghi `%a0` và kiểm tra xem nó có phải là số nguyên tố không.

/Nếu số đó nhỏ hơn 2, hàm sẽ trả về 0.

/Hàm sẽ kiểm tra tính chia hết của số đó với các số từ 2 đến căn bậc hai của số đó để xác định xem nó có phải là số nguyên tố hay không.

/Nếu là số nguyên tố, hàm sẽ trả về 1, ngược lại trả về 0.

+ Hàm `chiahet`: kiểm tra tính chia hết của một số để từ đó xem số đó có phải số nguyên tố hay không.

Bài 2(B8): Nhập mảng số nguyên từ bàn phím. In ra vị trí và giá trị của phần tử âm lớn nhất trong mảng.

Bài làm

- Mã nguồn:

```
.data
```

```
array: .space 100    # Khai báo mảng có thể chứa tối đa 100 phần tử
```

```
prompt: .asciiz "Nhap so phan tu : "
```

```
msg_input: .asciiz "Nhap phan tu thu "
```

```
msg_neg_max: .asciiz "Phan tu am lon nhat la "
```

```
msg_pos: .asciiz " tai vi tri "
```

```
msg_not_found: .asciiz "K co phan tu am "
```

```
.text
```

```
main:
```

```
# In ra thông báo yêu cầu nhập số phần tử của mảng
```

```
li $v0, 4
```

```
la $a0, prompt
```

```
syscall
```

```
# Nhập số phần tử của mảng từ bàn phím
```

```
li $v0, 5
```

```
syscall
```

```
move $t0, $v0 # Lưu số phần tử vào $t0
```

```
# Nhập các phần tử của mảng từ bàn phím
```

```
la $t1, array # Địa chỉ bắt đầu của mảng
```

```
li $t2, 0 # Biến đếm cho vòng lặp nhập
```

```
input_loop:
```

```
    bge $t2, $t0, find_max_neg # Nếu đã nhập đủ số phần tử thì chuyển sang tìm phần tử âm  
    lớn nhất
```

```
    addi $t3, $t2, 1 # Số thứ tự của phần tử đang nhập (bắt đầu từ 1)
```

```
    li $v0, 4
```

```
    la $a0, msg_input
```

```
    syscall
```

```
    move $a0, $t3 # In ra số thứ tự của phần tử đang nhập
```

```
    li $v0, 1
```

```
    syscall
```

```
    li $v0, 5
```

```
    syscall
```

```
    sw $v0, ($t1) # Lưu phần tử vào mảng
```

```
    addi $t1, $t1, 4 # Địa chỉ của phần tử tiếp theo trong mảng
```

```
    addi $t2, $t2, 1 # Tăng biến đếm
```

j input_loop

find_max_neg:

la \$t1, array # Lấy địa chỉ bắt đầu của mảng

li \$t2, 0 # Biến đếm cho vòng lặp kiểm tra

li \$t4, -999999999 # Khởi tạo giá trị lớn nhất là giá trị âm nhỏ nhất có thể

li \$t5, -1 # Biến lưu vị trí của phần tử âm lớn nhất, ban đầu đặt là -1

find_max_neg_loop:

bge \$t2, \$t0, print_result # Nếu đã kiểm tra hết các phần tử trong mảng thì in kết quả

lw \$t3, (\$t1) # Load phần tử từ mảng

bltz \$t3, check_max_neg # Nếu phần tử là số âm thì kiểm tra

addi \$t1, \$t1, 4 # Chuyển sang kiểm tra phần tử tiếp theo

addi \$t2, \$t2, 1 # Tăng biến đếm

j find_max_neg_loop

check_max_neg:

bgt \$t3, \$t4, update_max_neg # Nếu phần tử hiện tại lớn hơn phần tử âm lớn nhất hiện tại thì cập nhật

addi \$t1, \$t1, 4 # Chuyển sang kiểm tra phần tử tiếp theo

addi \$t2, \$t2, 1 # Tăng biến đếm

j find_max_neg_loop

update_max_neg:

move \$t4, \$t3 # Cập nhật giá trị lớn nhất

move \$t5, \$t2 # Lưu vị trí của phần tử âm lớn nhất

addi \$t1, \$t1, 4 # Chuyển sang kiểm tra phần tử tiếp theo

addi \$t2, \$t2, 1 # Tăng biến đếm

j find_max_neg_loop

print_result:

li \$v0, 4

la \$a0, msg_neg_max

syscall

li \$v0, 1

move \$a0, \$t4

syscall

li \$v0, 4

la \$a0, msg_pos

syscall

li \$v0, 1

addi \$t5,\$t5,1

move \$a0, \$t5

syscall

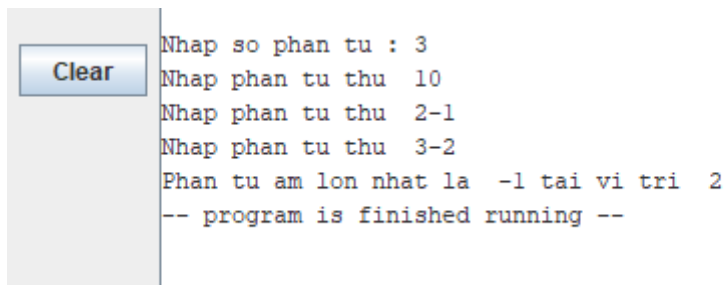
j exit

exit:

li \$v0, 10

syscall

- Kết quả:



```
Nhap so phan tu : 3
Nhap phan tu thu 10
Nhap phan tu thu 2-1
Nhap phan tu thu 3-2
Phan tu am lon nhat la -1 tai vi tri 2
-- program is finished running --
```

Bài 3(C7)

Bài làm

- Mã nguồn:

.data

str: .space 100

mess1: .ascii "Nhap vao xau ky tu: "

mess2: .ascii "Xau sau khi chuyen doi: "

.text

main:

Hiển thị thông báo nhập chuỗi

li \$v0, 4

la \$a0, mess1

syscall

Nhập chuỗi từ bàn phím

li \$v0, 8

la \$a0, str

li \$a1, 100

syscall

Gọi hàm convert

la \$a0, str # Load địa chỉ của mảng str vào \$a0

jal convert # Gọi hàm convertCase

Hiển thị kết quả

li \$v0, 4

la \$a0, mess2

syscall

Hiển thị chuỗi sau khi chuyển đổi

li \$v0, 4

la \$a0, str

syscall

Kết thúc chương trình

li \$v0, 10 # Sử dụng syscall 10 để thoát chương trình

syscall

Hàm convert

Đầu vào: \$a0 - địa chỉ của chuỗi cần chuyển đổi

Đầu ra: Chuỗi đã được chuyển đổi

convert:

move \$t0, \$a0 # Sao chép địa chỉ của chuỗi vào \$t0

loop:

lb \$t1, 0(\$t0) # Load ký tự hiện tại từ chuỗi

beqz \$t1, done # Nếu gặp ký tự kết thúc chuỗi thì thoát khỏi vòng lặp

Kiểm tra nếu là chữ hoa

li \$t2, 'A' # Gán giá trị 'A' vào \$t2

li \$t3, 'Z' # Gán giá trị 'Z' vào \$t3

blt \$t1, \$t2, lowercase

bgt \$t1, \$t3, lowercase

addi \$t1, \$t1, 32 # Chuyển chữ hoa thành chữ thường bằng cách tăng giá trị ký tự lên

32

sb \$t1, 0(\$t0) # Lưu ký tự đã được chuyển đổi vào chuỗi

j continue # Tiếp tục vòng lặp

lowercase:

li \$t2, 'a' # Gán giá trị 'a' vào \$t2

li \$t3, 'z' # Gán giá trị 'z' vào \$t3

blt \$t1, \$t2, continue # Nếu ký tự không phải chữ thường thì tiếp tục vòng lặp

bgt \$t1, \$t3, continue # Nếu ký tự không phải chữ thường thì tiếp tục vòng lặp

addi \$t1, \$t1, -32 # Chuyển chữ thường thành chữ hoa bằng cách giảm giá trị ký tự

xuống 32

sb \$t1, 0(\$t0) # Lưu ký tự đã được chuyển đổi vào chuỗi

continue:

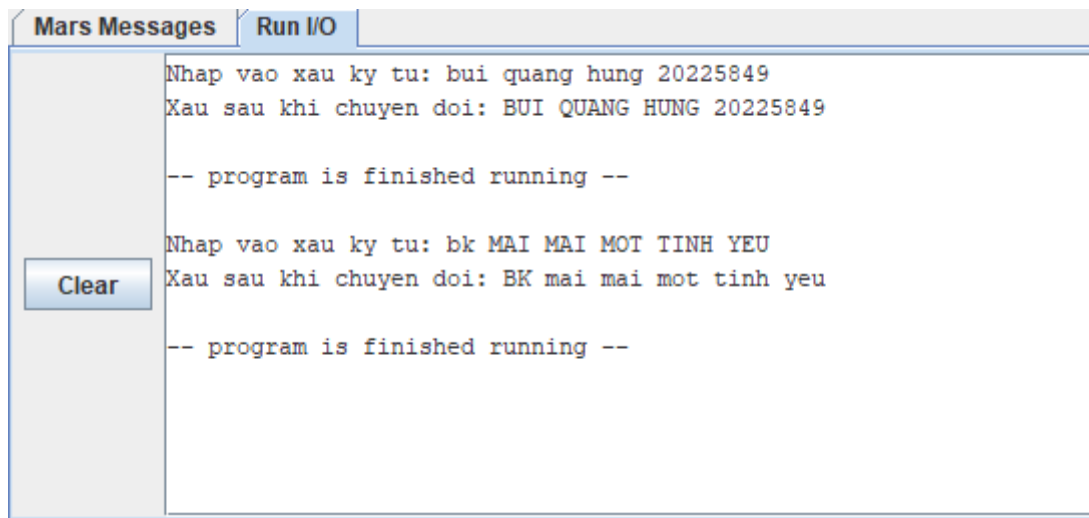
```
addi $t0, $t0, 1    # Tiến tới ký tự tiếp theo trong chuỗi
```

```
j loop    # Lặp lại vòng lặp
```

done:

```
jr $ra    # Trả về địa chỉ trở về
```

- Kết quả:



- Phân tích cách thực hiện:

+ Nhập một chuỗi từ bàn phím.

+ Dùng vòng lặp duyệt qua từng ký tự trong chuỗi.

+ Trong mỗi vòng lặp, chương trình kiểm tra xem ký tự hiện tại có phải là chữ hoa hay chữ thường hay không. Nếu là chữ hoa, chương trình chuyển đổi nó thành chữ thường và ngược lại bằng cách thay đổi giá trị ASCII tương ứng. Điều này thường được thực hiện bằng cách thay đổi giá trị của ký tự trong bộ nhớ.

- Ý nghĩa các hàm con:

+ **convert:** Đây là hàm thực hiện chuyển đổi các ký tự chữ hoa thành chữ thường và ngược lại trong chuỗi.

+ **lowcase:** Chuyển chữ thường thành chữ hoa

+ **loop:** Thực hiện vòng lặp duyệt qua từng ký tự và chuyển chữ hoa thành chữ thường

+ **continue:** Chuyển tới vòng lặp tiếp theo