ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TỔ CHỨC VÀ CẤU TRÚC MÁY TÍNH II

LỚP: IT012.N21.2

BÁO CÁO THỰC HÀNH SỐ 5 (LAB 05)

Giảng viên hướng dẫn: Nguyễn Thành Nhân

Sinh viên: Hồ Trọng Hiển

MSSV: 22520414

MỤC LỤC

Câu 1, 2: Thao tác với mảng

```
.data
array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
size1: .word 10
array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
size2: .word 16
array3: .space 8
size3: .word 8
s1: .asciiz " "
s2: .asciiz "\nARRAY1: "
s3: .asciiz "\nARRAY2: "
s4: .asciiz "\nARRAY3: "
s5: .asciiz "\nMang thu: "
s6: .asciiz "\nPhan tu thu: "
s7: .asciiz "INVALID."
```

- Khai báo data.

```
la $s0, array1
lw $a1, size1

la $s1, array2
lw $a2, size2

la $s2, array3
lw $a3, size3
```

- Load địa chỉ của array1, array2, array3 lần lượt vào $s0, $s1, $s2.
- Lưu kích thước của array1, array2, array3 lần lượt vào $a1, $a2, $a3.

- In array 1:

```
# In array1
add $t0, $0, $0
li $v0, 4
la $a0, s2
syscall
loop:
slt $t1, $t0, $a1
beq $t1, $0, exit
sll $t2, $t0, 2
add $t3, $s0, $t2
lw $t4, ($t3)
li $v0, 1
la $a0, ($t4)
syscall
li $v0, 4
la $a0, s1
syscall
addi $t0, $t0, 1
j loop
```

- Gán $t0 mang giá trị 0.
- In ra s2 là “ARRAY1: ”.
- Kiểm tra $t0 có bé hơn $a1 là kích thước mảng không?
+ Nếu có gán $t2 là độ dời.
+ Cộng độ dời với địa chỉ của array1 ra địa chỉ của phần tử muốn in.
+ Load word phần tử đó lên gán vào $t4
+ In ra giá trị $t4
+ Tăng $t0 lên 1, quay lại vòng lặp.
- Nếu $t0 >= $a1 thì thoát.

- In array 2:

```
#In array2
add $t0, $0, $0
li $v0, 4
la $a0, s3
syscall
loop2:
slt $t1, $t0, $a2
beq $t1, $0, exit2
add $t3, $s1, $t0
lb $t4, ($t3)
li $v0, 1
la $a0, ($t4)
syscall
li $v0, 4
la $a0, s1
syscall
addi $t0, $t0, 1
j loop2
exit2:
```

- Tương tự khi in array1.

- Tính toán: array3[i] = array2[i] + array2[size2-i-1]

```
#    array3[i] = array2[i] + array2[size2 - 1 - i]
add $t0, $0, $0
loop3:
slt $t1, $t0, $a3
beq $t1, $0, exit3
#Tinh size2-1-i
addi $t2, $a2, -1
sub $t2, $t2, $t0
# array2[size2-1-i]
add $t3, $s1, $t2
lb $t4, ($t3)
#array2[i]
add $t5, $s1, $t0
lb $t6, ($t5)
#array2[i] + array2[size2-1-i]
add $t7, $t6, $t4
#array3[i]
add $t8, $s2, $t0
sb $t7, ($t8)
addi $t0, $t0, 1
j loop3
exit3:
```

- Gán $t0 mang giá trị 0.

- Kiểm tra nếu $t0 >= $a3 thì thoát.

- Gán $t2 = size2-1

- Tiếp tục lấy $t2 - $t0 là size2-1-i.

- Cộng $t2 với $s1 ra địa chỉ của phần tử size2-1-i

- Load byte phần tử đó lên gán vào $t6 = array2[size2-i-1]

- Cộng $s1 và $t0 ra địa chỉ của phần tử i.

- Load byte phần tử đó vào $t4 ta có array2[i].

- Cộng $t4 và $t6 lưu vào $t7.

- Cộng $s2 với $t0 gán vào $t8 là địa chỉ của array3[i].

- Store byte $t7 vào $t8 là array3[i].

- Tăng $t0 lên 1.

- Lặp lại từ bước kiểm tra đến khi $t0 >= $a3 thì thoát.

- Nhập mảng thứ mấy và phần tử nào.

```
#Nhap mang thu may va phan
li $v0, 4
la $a0, s5
syscall
li $v0, 5
syscall
move $t0, $v0

sge $t1, $t0, 1
sle $t2, $t0, 3
bne $t1, $t2, EXIT

li $v0, 4
la $a0, s6
syscall

li $t1, 1
li $t2, 2
li $t3, 3

beq $t0, $t1, A_1
beq $t0, $t2, A_2
beq $t0, $t3, A_3
```

- Nhập số vào và lưu vào $t0

- So sánh $t0 >= 1 và $t0 <= 3:

+ Nếu sai thì in ra câu : “INVALID”

+ Nếu đúng thì tiếp tục so sánh coi nó là số nào: nếu là 1 thì nhảy đến A_1, 2 là A_2 và 3 là A_3.

```
A_1:
li $v0, 5
syscall
move $t0, $v0

sge $t1, $t0, 0
slt $t2, $t0, $a1
bne $t1, $t2, EXIT

sll $t1, $t0, 2
add $t2, $s0, $t1
lw $t3, ($t2)
li $v0, 1
la $a0, ($t3)
syscall
j exit_pro
```

- Ví dụ cho mảng A_1, hai mảng kia tương tự.

- Nhập vị trí index muốn lấy, lưu vào $t0

- Kiểm tra $t0 có nằm trong phạm vi index của mảng không, nếu không thì in câu INVALID.

- Nếu có thì dịch trái $t0 hai bit, cộng nó với địa chỉ nền của array1. Load word phần tử đó lên gán vào $t3. In ra $t3 là phần tử cần tìm.

- Sau khi xong thì nhảy đến exit_pro kết thúc chương trình.

Câu 3: Dùng con trỏ.

3a)

```
.data
array: .space 200
a: .word
s1: .asciiz "\nNhap so luong phan tu: "
s2: .asciiz "\nNhap cac phan tu: "
s3: .asciiz " "
s4: .asciiz "\nMAX: "
s5: .asciiz "\nMIN: "
s6: .asciiz "\nSUM = "
s7: .asciiz "\nPhan tu thu: "
s8: .asciiz "INVALID."
```

- Khai báo data.

- a là biến con trỏ.

```
la $s7, a
li $v0, 4
la $a0, s2
syscall

li $a1, 0
add $t0, $s7, $0

loop_test:
li $v0, 5
syscall
move $t8, $v0
beq $t8, -1, MIN_MAX
sw $t8, 0($t0)
addi $t0, $t0, 4
addi $a1, $a1, 1
j loop_test
```

- Lấy địa chỉ của con trỏ a lưu vào $s7

- Gán $a1 là 0 mang ý nghĩa số phần tử của mảng.

- $t0 đang lưu địa chỉ của phần tử đầu tiên trong mảng.

- Vào vòng lặp:

+ Nhập một số và lưu vào $t8

+ Kiểm tra nếu số vừa nhập khác -1 thì tiếp tục, không thì kết thúc vòng lặp và nhảy vào MIN_MAX

+ Store word giá trị vừa nhập vào địa chỉ mà $t0 đang lưu.

+ Tăng $t0 lên 4 là nhảy sang word mới, tăng $a1 lên 1 là mảng có thêm một phần tử.

+ Quay về lại vòng lặp, lặp đến khi nào nhập -1 thì ngừng lặp.

```
MIN_MAX:

lw $s2, ($s7) #MAX
lw $s3, ($s7) #MIN
add $t0, $0, $s7
addi $t0, $t0, 4
sll $t4, $a1, 2
add $t4, $t4, $s7

loop_2:
slt $t1, $t0, $t4
beq $t1, 0, print
lw $t3, ($t0)
slt $t2, $t3, $s3
bne $t2, $0, MIN
sgt $t2, $t3, $s2
bne $t2, $0, MAX
addi $t0, $t0, 4
j loop_2
```

- Gán giá trị đầu của mảng là hai giá trị min và max.

- Lặp qua các phần tử của mảng, nếu có phần tử nào lớn hơn $s2 thì nhảy vào MAX cập nhật lại giá trị $s2, nếu nhỏ hơn $s3 thì nhảy vào MIN và cập nhật lại giá trị $s3.

- Nếu nằm giữa khoảng min và max thì tăng $t0 lên 4 để qua word tiếp theo.

```
print:
li $v0, 4
la $a0, s4
syscall
li $v0, 1
la $a0, ($s2)
syscall
li $v0, 4
la $a0, s5
syscall
li $v0, 1
la $a0, ($s3)
syscall
```

- In ra hai giá trị min và max lần lượt nằm trong thanh ghi $s3, $s2.

```
#Tong
add $t0, $0, $s7
sll $t4, $a1, 2
add $t4, $t4, $s7
add $s0, $0, $0
loop_tong:
slt $t2, $t0, $t4
beq $t2, $0, print_sum
lw $t3, ($t0)
add $s0, $s0, $t3
add $t0, $t0, 4
j loop_tong

print_sum:
li $v0, 4
la $a0, s6
syscall
li $v0, 1
la $a0, ($s0)
syscall
```

- Tính tổng:

+ $t0 lưu địa chỉ nền của mảng.

+ sll $t4, $t1, 2: $t4 là độ dịch tối đa của địa chỉ nền của mảng.

+ Cộng $t4 với $s7 là giới hạn của mảng.

+ Gán $s0 = 0 là tổng cần tính.

+ Lặp qua các phần tử của mảng. Mỗi lần như thế kiểm tra xem $t0 còn bé hơn $t4 hay không? Nếu không thì nhảy tới print_sum in ra tổng. Nếu còn bé hơn thì load word giá trị tại địa chỉ hiện tại của $t0 lên và cộng dồn vào $s0. Sau đó tăng $t0 lên 4 là nhảy sang word tiếp theo.

- In ra tổng: Gọi syscall để in ra giá trị trong $s0 là tổng cần tìm.

- Nhập chỉ số và in ra màn hình giá trị tại chỉ số đó.

```
#Nhap vao chi so va in ra
li $v0, 4
la $a0, s7
syscall
li $v0, 5
syscall
move $s0, $v0

sge $t2, $s0, 0
slt $t3, $s0, $a1
bne $t2, $t3, invalid

sll $t2, $s0, 2
add $t3, $s7, $t2
lw $t2, ($t3)
li $v0, 1
la $a0, ($t2)
syscall
j exit

invalid:
li $v0, 4
la $a0, s8
syscall
exit:
```

- Các thao tác kiểm tra index có hợp lệ hay không và thao tác in ra tương tự câu 1.

3b)

- Khai báo data

```
.data
a: .word
s1: .asciiz "\nNhap so luong phan tu: "
s2: .asciiz "\nNhap cac phan tu: "
s3: .asciiz " "
s4: .asciiz "\nNhap i: "
s5: .asciiz "\nNhap j: "
s6: .asciiz "\nINVALID"
```

- Nhập mảng tương tự câu 3a

```
.text

la $s2, a
li $v0, 4
la $a0, s2
syscall

li $al, 0
add $t0, $s2, $0

loop_test:
li $v0, 5
syscall
move $t8, $v0
beq $t8, -1, EXIT
sw $t8, 0($t0)
addi $t0, $t0, 4
addi $al, $al, 1
j loop_test

EXIT:
```

```
exit:
li $v0, 4
la $a0, s4
syscall
li $v0, 5
syscall
move $s0, $v0

sge $t1, $s0, 0
slt $t2, $s0, $al
bne $t1, $t2, invalid


li $v0, 4
la $a0, s5
syscall
li $v0, 5
syscall
move $s1, $v0

slt $t0, $s0, $s1
sll $t3, $s0, 2
add $t4, $s2, $t3
beq $t0, $0, else
sw $s0, ($t4)
j exit_p

else:
sw $s1, ($t4)
j exit_p

invalid:
li $v0, 4
la $a0, s6
syscall
j exit
```

- Nhập i: Nhập giá trị i lưu vào $s0, kiểm tra coi giá trị vừa nhập có nằm trong khoảng hợp lệ không?
+ Nếu không thì nhảy vào invalid và in ra “INVALID” và nhảy về lại exit để nhập lại giá trị i.

+ Nếu có thì tiến hành nhập j, lưu giá trị đó vào $s1.

+ sll $t3, $s0, 2: Lấy địa chỉ của A[i]

+ So sánh hai giá trị trong hai thanh ghi $s0 là i và $s1 là j. Nếu i < j thì thì store word giá trị i vào $t3 là A[i]. Nếu ngược lại thì store word giá trị j vào A[i]