Lab 6

Array and Pointer

Đinh Ngọc Khánh Huyền - 20225726

Assigment 1

b,

Code

```
.data
        arr: .word 9,1,3,-4,34,12
        space: .asciiz " "
 .text
 .globl main
main:
        lui $s0, 0x1001
        li $t0, 0 #i = 0
        li $t1, 0 #j = 0
        li $s1, 6 #n = 5
        li $s2, 6 #n-i for inner loop
        add $t2, $zero, $s0 #for iterating addr by i
        add $t3, $zero, $s0 #for iterating addr by j
        addi $s1, $s1, -1
outer_loop:
        li $t1, 0 #j = 0
        addi $s2, $s2, -1 #decreasing size for inner_loop
        add $t3, $zero, $s0 #resetting addr itr j
 inner_loop:
```

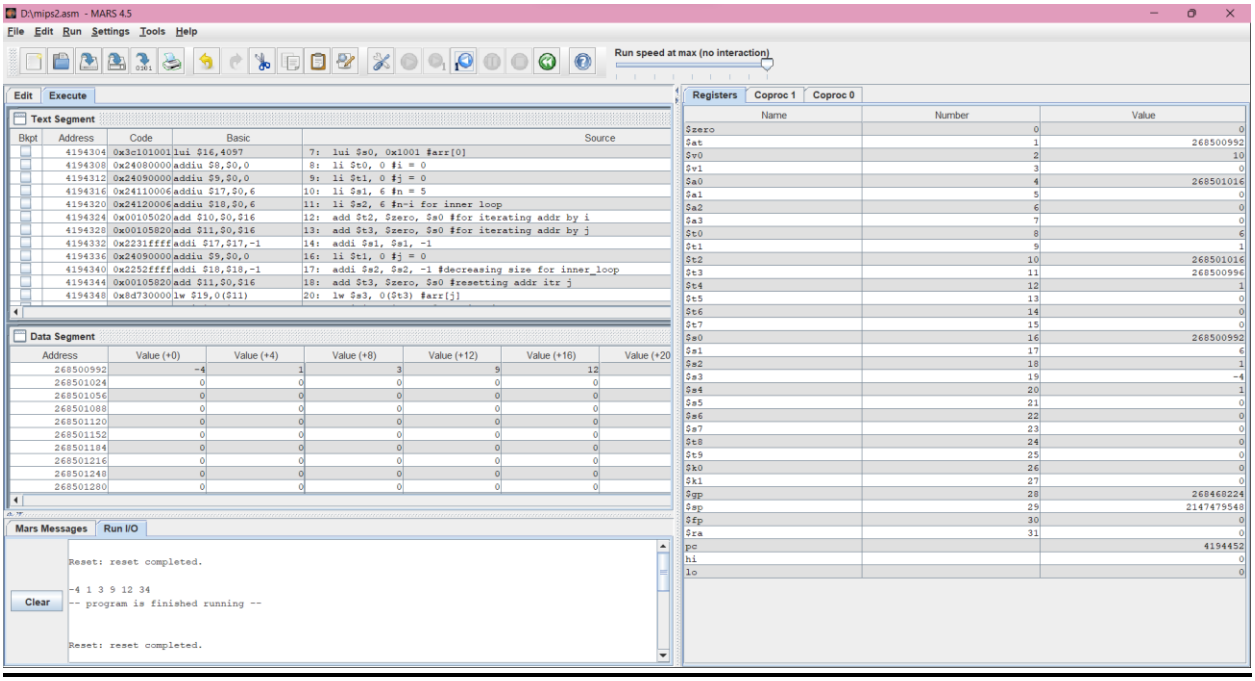
```
        lw $s3, 0($t3) #arr[j]
        addi $t3, $t3, 4 #addr itr j += 4
        lw $s4, 0($t3) #arr[j+1]
        addi $t1, $t1, 1 #j++
        slt $t4, $s3, $s4 #set $t4 = 1 if $s3 < $s4
        bne $t4, $zero, cond
 swap:
        sw $s3, 0($t3)
        sw $s4, -4($t3)
        lw $s4, 0($t3)
 cond:
        bne $t1, $s2, inner_loop #j != n-i
        addi $t0, $t0, 1 #i++
        bne $t0, $s1, outer_loop #i != n
        li $t0, 0
        addi $s1, $s1, 1
print_loop:
        li $v0, 1
        lw $a0, 0($t2)
        syscall
        li $v0, 4
        la $a0, space
        syscall
        addi $t2, $t2, 4
        addi $t0, $t0, 1
        bne $t0, $s1, print_loop
```

exit:

li $v0, 10

syscall

Result



Explain

Nạp địa chỉ của mảng vào thanh ghi $s0, khởi tạo 2 biến i và j bằng 0.

Thanh ghi $s1 đại diện cho tổng số phần tử trong mảng và $s2 được thiết lập cho số vòng lặp trong được thiết lập bằng 6.

Vòng lặp ngoài: duyệt các phần tử.

Vòng lặp trong: so sánh 2 phần tử liền kề.

Nạp 2 giá trị j và j+1 vào 2 thanh ghi $s3, $s4. Dùng thanh ghi $t4 để kiểm tra nếu j<i+1 thì thanh ghi là giá trị 1 và ngược lại. Nếu $t4 =1 thì sẽ swap 2 giá trị cạnh nhau. Kết thúc vòng lặp khi j>=n-1.

Sau khi vòng lặp trong kết thúc, thanh ghi $t0 sẽ được tăng lên thành 1, đại diện cho việc tăng bộ đếm I và vòng lặp ngoài sẽ dừng lại khi i>=n.

C,

```
.data
        output: .asciiz "\nSorted array: "
        array: .word -5,6,7,3,4,1,2,3,45,-2
.text
.globl main
main:
         jal Sort
         jal PrintArray
         j Exit


## Sorting
Sort:
         li $t0, 0 # $t0 = i = 0
         la $a1, array # a[0]
         addi $a1, $a1, 4 # skip the first element
while1:
         beq $t0, 10, end # exit if the end of array is reached
         lw $t1, ($a1) # $t1 = a[i]
         addi $t2, $t0, 0 # $t2 = j = i
         la $a2, ($a1)
while2:
        blez $t2, cont
        la $a3, ($a2)
        addi $a3, $a3, -4
        lw $t4, ($a3) # $t4 = Value of a[j-1]
        bge $t4, $t1, cont # if a[j-1] >= a[i], continue
```

```
        la $a3, ($a2)
        addi $a3, $a3, -4
        lw $t5, ($a3) # $t5 = Value of a[j-1]
        sw $t5, ($a2) # assign a[j] = a[j-1]
        addi $t2, $t2, -1 # decrease j by 1
        addi $a2, $a2, -4 # move pointer backwards
        j while2
cont:
        sw $t1, ($a2) # assign a[j] = a[i]
        addi $t0, $t0, 1 # increment i by 1
        addi $a1, $a1, 4 # move pointer to the next element
        j while1
end:
        jr $ra

# Print
PrintArray:
        li $v0, 4
        la $a0, output
        syscall
        li $t0, 0 # reset counter to loop through array elements
        la $a1, array # address of the array
        li $t1, 11 # length of the array
PrintElement:
        beq $t0, $t1, Exit # if all elements are printed, exit the loop
        lw $a0, ($a1) # read the $t0th element of the array
```

```
    li $v0, 1
    syscall # print that element
    li $v0, 11
    la $a0, 32
    syscall # print space
    addi $t0, $t0, 1 # increment counter to move to the next element
    addi $a1, $a1, 4 # move pointer to the next element
    j PrintElement # return to loop to print next
Exit:
    li $v0, 10 # Exit syscall
    syscall
```

Assigment 2

Indexing method: dùng 1 thanh ghi để giữ chỉ mục của phần tử mảng đang được kiểm tra.

Ưu điểm:

- Dễ hiểu và dễ sử dụng
- Truy cập dễ dàng: Trong một mảng, việc truy cập một phần tử rất dễ dàng bằng cách sử dụng số chỉ mục.
- Tối ưu hóa hiệu suất: Khi sử dụng chỉ mục, việc truy cập các phần tử của mảng có thể được thực hiện một cách hiệu quả hơn, đặc biệt là trong các tình huống có nhiều phép tính logic và toán học phức tạp.

Nhược điểm:

- Yêu cầu bộ nhớ phụ: Việc lưu trữ chỉ mục trong bộ nhớ yêu cầu không gian bộ nhớ phụ, làm tăng bộ nhớ cần thiết cho chương trình.
- Tăng chi phí của các thao tác cập nhật: Khi dữ liệu trong mảng được thay đổi, chỉ mục phải được cập nhật để thể hiện sự thay đổi này. Việc này có thể tăng thời gian và chi phí của các thao tác cập nhật.

Pointer-updating method: dung 1 thanh ghi để giữ địa chỉ của yếu tố đang được kiểm tra của mảng.

Ưu điểm:

- Tiết kiệm bộ nhớ: Phương pháp này không yêu cầu lưu trữ thêm thông tin chỉ mục, giúp giảm bộ nhớ cần thiết cho chương trình.
- Giảm độ trễ khi truy cập: do truy cập trực tiếp vào địa chỉ của phần tử nên có thể giảm độ trễ so với việc sử dụng chỉ mục.

Nhược điểm:

- Phức tạp hơn trong việc quản lí con trỏ: Con trỏ không cung cấp mức độ an toàn như các loại dữ liệu khác, vì chúng cho phép thao tác trực tiếp các địa chỉ bộ nhớ. Điều này có thể khiến việc đưa lỗi hoặc lỗi vào mã trở nên dễ dàng hơn và có thể khiến việc xác định và khắc phục sự cố trở nên khó khăn hơn.
- Khó khăn trong việc tối ưu hóa hiệu suất: Việc tối ưu hóa hiệu suất của các thao tác truy cập dữ liệu thông qua con trỏ có thể phức tạp hơn so với việc sử dụng chỉ mục.