

Vũ Minh Hải 20194550

## Báo cáo bài tập tuần 7

### Assignment 1

Chương trình:

```
.data
array: .space 400      #100 elements
prompt: .asciiz "Enter an integer (0 to quit) :"
mes1: .asciiz "Before sorted:"
mes2: .asciiz "\nAfter sorted:"

.text
main:
    la $a2, array      # Load address of array

read_array:
    li $v0, 51         # Input int mode
    la $a0, prompt     # Message
    syscall

    bne $a1, 0, read_array    # If $a1 != 0: an error has occurred,
get input again
    nop

    beqz $a0, read_done      # If A[i] = 0, exit read_array
    nop

    sw $a0, 0($a2)          # Save to A[i]
    addiu $a2, $a2, 4       # Move to next element

    j read_array            # Loop to read array
    nop

read_done:
    addi $a1,$a2,-4         # $a1 = Address(array[n-1])
    li $v0, 4              # Print string mode
    la $a0, mes1
    syscall

    la $a0,array            # $a0 = Address(array[0])
    addi $a3,$a0,0          # Mark first element
    jal print_array
    nop

    la $a0,array            # $a0 = Address(array[0])
    addi $a3,$a0,0          # Remark first element
    addi $t8,$a1,0          # Mark last element
```

```

        j sort                # sort
        nop

end:
    li $v0, 4                # Print string mode
    la $a0, mes2
    syscall

    addi $a1,$t8,0           #Load last element
    jal print_array
    li $v0, 10               #exit
    syscall

sort:
    beq $a0,$a1,done #single element list is sorted
    j max                   #call the max procedure
    nop

after_max:
    addi $a1,$a1,-4          # Decrement pointer to last element
    j sort                   # Repeat sort for smaller list
    nop

done:
    j end
    nop

max:
    addi $v0,$a0,0           # Init pointer to first element

loop:
    beq $v0,$a1,ret # If now=last, return
    lw $t1,0($v0)            # Load current element into $t1
    slt $t3,$t1,$zero        # Check if (now) < 0
    bne $t3, $zero, skip    # If < 0, skip this element
    nop
    addi $v1,$v0,0           # $v1 = Address(current element)

find_next:
    addi $v1,$v1,4           # Point to next element
    slt $t3,$a1,$v1          # If next element out of range
    bne $t3,$zero,ret        # If out of range, return
    nop
    lw $t2,0($v1)            # Load next element into $t2
    slt $t3,$t2,$zero        # Check if (next) < 0
    bne $t3, $zero, find_next # If < 0, skip this element, move to
next
    nop

```

```

    slt $t0,$t2,$t1 # (next) < (now) ?
    addi $t3, $v0, 0    # Copy v0
    addi $v0, $v0, 4    # Point to next element
    beq $t0,$zero,loop  # If (next)>=(now), repeat
    sw $t2,0($t3)       # Copy A[next] to A[now]
    sw $t1,0($v1)       # Copy A[now] to A[next]
    j loop              # Change completed; now repeat
    nop

ret:
    j after_max

skip:
    addi $v0, $v0, 4    # Move to next element
    j loop              # Continue loop
    nop

print_array:
    addi $v0, $zero, 11
    li $a0, ' '
    syscall

    li $v0, 1          # Print int mode
    lw $a0,0($a3)
    syscall

    addi $a3, $a3, 4
    slt $t7, $a1, $a3
    beq $t7,$zero,print_array
    nop

    jr $ra

```

Kết quả:

```

Before sorted: -1 150 190 170 -1 160 180
After sorted:  -1 150 160 170 -1 180 190
-- program is finished running --

```

Giải thích:

Đầu tiên, load giá trị địa chỉ của mảng vào \$a2:

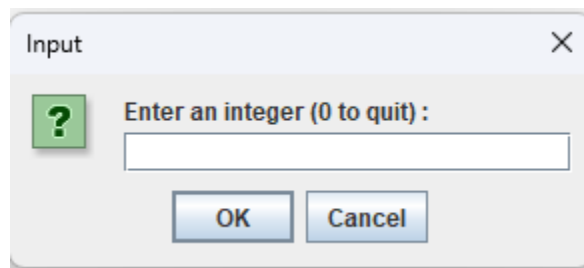
\$a2	6	0x10010000
------	---	------------

Tiếp theo, chuyển đến khối read\_array có nhiệm vụ nhận vào mảng được người dùng nhập từ bàn phím

Đưa thanh ghi \$v0 về giá trị 51 tương ứng với gọi dialog nhập số nguyên

Gán vào \$a0 địa chỉ của prompt để in chuỗi này vào hộp thoại dialog

Sau đó gọi lệnh syscall để hiển thị hộp thoại nhập số liệu:



Khi nhập sai dạng của số liệu (VD "hello", a1 sẽ nhận giá trị -1:

\$a1	5	0xffffffff
------	---	------------

Tương tự, giá trị này sẽ được gán như sau:

- 0: Giá trị nhập vào đúng dạng
- -1: Có giá trị được nhập nhưng sai kiểu dữ liệu
- -2: Người dùng chọn Cancel
- -3: Người dùng chọn OK nhưng giá trị để trống

Vì vậy, sau khi nhận giá trị, ta kiểm tra giá trị nhập vào đã thoả mãn hay chưa nếu chưa thì cho quay lại khối read\_array để nhập lại.

Nếu kiểu dữ liệu đã thoả mãn và thấy giá trị bằng 0, ta kết thúc nhập dữ liệu, rẽ nhánh sang khối read\_done. Còn ngược lại, lưu giá trị này vào mảng, sau đó tăng giá trị biến chạyj thêm 4 để trở đến phần tử tiếp theo và tiếp tục đọc từ bàn phím.

Sau khi đọc xong mảng, ta thu được kết quả:

-1	150	190	170	-1	160	180
----	-----	-----	-----	----	-----	-----

Tiếp theo, ta chuyển đến khối read\_done:

Đầu tiên, ta đưa giá trị \$v0 về 4 và gán \$a0 địa chỉ của chuỗi mes1 để in chuỗi này ra màn hình:

Before sorted:

Sau đó, ta gán \$a3 là địa chỉ của mảng, cũng là địa chỉ của phần tử đầu tiên, \$a1 lưu giá trị địa chỉ cuối cùng của mảng để đánh dấu vị trí cần in ra.

\$a1	5	0x10010018
\$a2	6	0x1001001c
\$a3	7	0x10010000

Sau khi đã đánh dấu, ta gọi hàm print\_array, dung lệnh jal để đánh dấu vị trí tiếp tục chương trình sau khi kết thúc hàm:

\$ra	31	0x00400060
pc		0x0040012c

Đầu tiên, ta chuyển \$v0 về giá trị 11 để in ra ký tự space ngăn cách giữa các phần tử của mảng. Sau đó, ta lại chuyển giá trị này về 1, load giá trị của \$a3 vào \$a0 để in nó ra màn hình.

Tiếp theo, ta tăng giá trị của \$a3 thêm 4 để trở đến phần tử tiếp theo, nếu phần tử này nằm ngoài phạm vi của mảng, có giá trị lớn hơn địa chỉ của phần tử cuối cùng trong mảng, ta biết mảng đã được in hết nên sẽ nhảy đến vị trí tiếp theo của chương trình đang được lưu trong thanh ghi \$ra.

Kết quả sau khi in mảng ra màn hình:

```
Before sorted: -1 150 190 170 -1 160 180
```

Khi trở lại khối lệnh chính, ta lấy lại các giá trị \$a0 lưu địa chỉ mảng và copy giá trị này vào \$a3, tương tự, ta copy thêm \$a1 vào \$t8 để lưu địa chỉ phần tử cuối cùng của mảng.

Sau khi đã lưu lại các giá trị cần thiết, ta thực hiện việc sắp xếp lại mảng:

\$a0 lưu địa chỉ của A[0], \$a1 lưu địa chỉ của A[n-1].

Sau đó kiểm tra 2 giá trị này, nếu \$a0 = \$a1 => mảng chỉ có 1 phần tử hoặc đã xét đến phần tử cuối cùng => Kết thúc chương trình.

Ở đây, \$a0 != \$a1 => chương trình tiếp tục, nhảy đến khối max.

Tiếp theo, lưu giá trị \$a0 vào \$v0 để làm biến chạy:

\$v0	2	0x10010000
\$v1	3	0x00000000
\$a0	4	0x10010000

Khi bắt đầu vòng lặp, ta kiểm tra biến chạy đã chạy đến phần tử cuối cùng của mảng chưa, nếu nó có giá trị trùng với \$a1, tức là ta đã xét đến phần tử cuối cùng và nhảy đến khối ret. Ở đây, ta bắt đầu xét từ phần tử đầu tiên nên điều kiện này sai, ta sẽ tiếp tục thực hiện chương trình.

Ta thực hiện load giá trị của phần tử đang xét vào \$t1:

\$t1	9	0xffffffff
------	---	------------

Sau đó kiểm tra giá trị này, nếu <0, tương ứng với vị trí của một cái cây, ta nhảy đến khối skip để tăng giá trị biến chạy thêm 4 và quay lại vòng lặp như trên.

Khi xét đến A[1] = 0x96 (150) > 0, ta tiếp tục chương trình, copy giá trị của \$v0 vào \$v1.

Khối find\_next thực hiện tìm phần tử >0 ngay sau phần tử đang xét. Khi bắt đầu vòng lặp, thực hiện tăng \$v1 lên 4 để trở đến phần tử tiếp theo, so sánh với \$a1 để đảm bảo không vượt phạm vi của mảng.

Nếu \$v1 > \$a1, tức phần tử \$v0 đã là phần tử cuối cùng của mảng, ta chuyển đến khối ret

Ở đây, ta tìm được giá trị tiếp theo là 190, lưu giá trị này vào \$t2:

\$t1	9	0x00000096
\$t2	10	0x000000be

Nếu phần tử này < 0, tương ứng với vị trí của cây, ta quay lại khối find\_next để xét các phần tử tiếp theo trong mảng.

Sau khi tìm được 2 phần tử liền kề, dương trong mảng, ta tiến hành so sánh 2 giá trị. Nếu  $A[j+1] \geq A[j] \Rightarrow$  tăng giá trị \$v0 thêm 4 để đưa biến chạy đến phần tử tiếp theo và quay lại vòng lặp mà không làm gì. Ngược lại, thực hiện swap 2 giá trị. Ở đây,  $190 > 150$ , ta chỉ quay lại vòng lặp.

Chương trình tiếp tục đến khi \$v0 lưu địa chỉ  $A[2] = 190$ , \$v1 lưu địa chỉ  $A[3] = 170$ . Hai lệnh sw lưu chéo giá trị của 2 biến này cho nhau:

Value (+8)	Value (+c)
0x000000aa	0x000000be

Sau đó tiếp tục quay lại vòng lặp.

Kết thúc lần lặp đầu tiên, khi \$v0 = \$a1:

\$v0	2	0x10010018
\$v1	3	0x10010018
\$a0	4	0x10010000
\$a1	5	0x10010018

Ta đã xét đến phần tử cuối cùng. Khi này, phần tử lớn nhất 0xbe đã được đẩy đến cuối cùng:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	0xffffffff	0x00000096	0x000000aa	0x000000a0	0xffffffff	0x000000b4	0x000000be
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Ta trừ \$a1 đi 4 để trở đến phần tử  $A[n-2]$  và tiếp tục quay lại khối loop để sắp xếp tiếp mảng còn lại.

Vòng lặp kết thúc khi \$a1 = \$a0, tất cả phần tử đều đã được sắp xếp.

Khi đã sắp xếp xong, ta tiếp tục thực hiện syscall tương tự ở trên để in chuỗi ra màn hình:

After sorted:

Sau đó ta load lại địa chỉ phần tử cuối mảng đã được copy vào \$t8 từ trước để trả lại cho \$a1, cùng với \$a3 vẫn lưu địa chỉ của mảng để gọi hàm print\_array, in lại mảng sau khi đã được sắp xếp:

After sorted: -1 150 160 170 -1 180 190

Cuối cùng, gán \$v0 = 10 và gọi syscall để kết thúc chương trình.