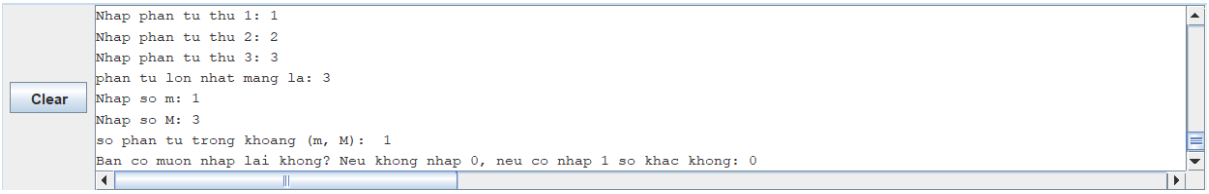
BÁO CÁO BÀI GIỮA KÌ

Họ và tên: Nguyễn Khánh Toàn

MSSV: 20225936

Bài 1: (Bài 4 trong File đề)

Với đầu vào là 1 mảng 3 phần tử 1, 2, 3; m = 1 và M = 3 và không muốn nhập tiếp thì ta sẽ có kết quả:



Code:

```
.data
array: .space 400
input_prompt: .asciiz "Nhap so phan tu cua mang (1 so be hon 100): "
prompt: .asciiz "Nhap phan tu thu "
prompt2: .asciiz ": "
enter_prompt: .asciiz "\n"
comclu1_prompt: .asciiz "phan tu lon nhat mang la: "
comclu2_prompt: .asciiz "so phan tu trong khoang (m, M):  "
strm: .asciiz "Nhap so m: "
strM: .asciiz "Nhap so M: "
prompt_reset: .asciiz "Ban co muon nhap lai khong? Neu khong nhap 0, neu co nhap 1 so khac khong: "
.text
main:la $s0, array # lưu địa chỉ phần tử đầu tiên của mảng
    # In prompt số lượng phần tử của mảng
    li $v0, 4
    la $a0, input_prompt
    syscall
```

```
    # Nhập số lượng phần tử của mảng từ bn phím
    li $v0, 5
    syscall

    addi $t0, $zero, 101
    slt $t1, $v0, $t0  #check < 100
    beq $t1, $zero, exit  #check xem số bé hơn 100

    addi $t2, $zero, 1
    slt $t1, $v0, $t2
    bne $t1, $zero, exit # check xem số phần tử bé hơn 1 ko
    add $t0, $zero, $v0 # Lưu số lượng phần tử vào $t0

    addi $t2, $zero, 1 # khởi tạo thứ tự đầu tiên của mảng

    jal read_int # lưu phần tử đầu tiên
    add $s2, $zero, $s1 # đặt giá trị max đầu tiên = phần tử đầu tiên
    sw $s1, 0($s0) # lưu vào mảng
    addi $s0, $s0, 4

    beq $t0, $t2, exit_add #check xem số phần tử lớn hơn 1 không, nếu có thì thoát nhập

    addi $t2, $t2, 1 # cập nhật thứ tự phần tử nhập vào
    j add_element

read_int: li $v0, 4          # System call for print_str
    la $a0, prompt     # Load address of prompt into $a0
    syscall            # Call operating system to perform operation

    li $v0, 1              # Lệnh in số nguyên
    add $a0, $t2, $zero          # Di chuyển số nguyên từ $t2 đến $a0 để in
```

```
    syscall

    li $v0, 4          # System call for print_str
    la $a0, prompt2     # Load address of prompt into $a0
    syscall

    li $v0, 5              # Lệnh để đọc một số nguyên
    syscall
    add $s1, $zero, $v0

    jr $ra

add_element:
    jal read_int

    slt $t3, $s2, $s1 # check xem giá trị mới nhập có phải giá trị lớn nhất không
    beq $t3, $zero, skip_save_max
    add $s2, $s1, $zero # lưu giá trị lớn nhất hiện thời

    skip_save_max:
    sw $s1, 0($s0) # lưu vào mảng
    addi $s0, $s0, 4

    addi $t2, $t2, 1 # cập nhật thứ tự phần tử nhập vào
    slt $t3, $t0, $t2
    beq $t3, $zero, add_element

exit_add: li $v0, 4
    la $a0, comclu1_prompt
    syscall

    li $v0, 1
```

```
    add $a0, $s2, $zero        # in ra giá trị lớn nhất trong mảng
    syscall

    li $v0, 4
    la $a0, enter_prompt
    syscall

#Nhập các giá trị m,M vào t1, t2
    # In prompt số m
    li $v0, 4
    la $a0, strm
    syscall

    # nhập m
    li $v0, 5
    syscall
    add $t1, $zero, $v0

    # In prompt số M
    li $v0, 4
    la $a0, strM
    syscall

    # Nhập M
    li $v0, 5
    syscall
    add $t2, $zero, $v0

    # bắt đầu xét
    add $t3, $zero, $zero # count = 0
    addi $t6, $zero, 0 # temp = 0 -> số phần tử của mảng đã được xét
    la $s0, array # lấy phần tử đầu tiên của mảng
```

```
loop_count: lw $s1, 0($s0)
    slt $t4, $t1, $s1
    beq $t4, $zero, next_loop #so sánh với m
    slt $t4, $s1, $t2
    beq $t4, $zero, next_loop #so sánh với M

    addi $t3, $t3, 1 # tăng count
next_loop: addi $t6, $t6, 1
    addi $s0, $s0, 4
    bne $t6, $t0, loop_count

print_result:li $v0, 4
    la $a0, comclu2_prompt
    syscall

    li $v0, 1
    add $a0, $t3, $zero  # in ra số phần tử trong khoảng (m, M)
    syscall

    li $v0, 4
    la $a0, enter_prompt
    syscall

reset: li $v0, 4
    la $a0, prompt_reset # in ra prompt hỏi người dùng có muốn lặp lại chương trình
    syscall

    # nhập lựa chọn
    li $v0, 5
    syscall

    bne $v0, $zero, main
```

exit:

Giải thích:

- Ta khai báo một khoảng 400 byte cho mảng tối đa 100 phần tử, cùng với các câu lệnh prompt

```
.data
array: .space 400
input_prompt: .asciiz "Nhap so phan tu cua mang (1 so be hon 100): "
prompt: .asciiz "Nhap phan tu thu "
prompt2: .asciiz ": "
enter_prompt: .asciiz "\n"
comclul_prompt: .asciiz "phan tu lon nhat mang la: "
comclu2_prompt: .asciiz "so phan tu trong khoang (m, M):  "
strm: .asciiz "Nhap so m: "
strM: .asciiz "Nhap so M: "
prompt_reset: .asciiz "Ban co muon nhap lai khong? Neu khong nhap 0, neu co nhap 1 so khac khong: "
```

- Nhập số lượng phần tử của mảng

```
.text
main:la $s0, array # lưu địa chỉ phần tử đầu tiên của mảng
    # In prompt số lượng phần tử của mảng
    li $v0, 4
    la $a0, input_prompt
    syscall

    # Nhập số lượng phần tử của mảng từ bàn phím
    li $v0, 5
    syscall
```

- Kiểm tra xem số lượng phần tử của mảng có lớn hơn 100 hay bé hơn 1 không, nếu có tự động exit chương trình, nếu có thì lưu vào thanh ghi $t1

```
    #check < 100
    addi $t0, $zero, 101
    slt $t1, $v0, $t0
    beq $t1, $zero, exit  #check xem số bé hơn 100

    addi $t2, $zero, 1
    slt $t1, $v0, $t2
    bne $t1, $zero, exit # check xem số phần tử bé hơn 1 ko
    add $t0, $zero, $v0 # Lưu số lượng phần tử vào $t0
```

- Viết hàm để đọc giá trị các phần tử nhập vào mảng, sau đó lưu các giá trị đó vào thanh ghi $s1

```
read_int: li $v0, 4           # System call for print_str
    la $a0, prompt      # Load address of prompt into $a0
    syscall             # Call operating system to perform operation

    li $v0, 1               # Lệnh in số nguyên
    add $a0, $t2, $zero            # Di chuyển số nguyên từ $t2 đến $a0 để in
    syscall

    li $v0, 4           # System call for print_str
    la $a0, prompt2      # Load address of prompt into $a0
    syscall

    li $v0, 5               # Lệnh để đọc một số nguyên
    syscall
    add $s1, $zero, $v0

    jr $ra
```

- Thực hiện thao tác nhập phần tử đầu tiên của mảng, lưu địa chỉ phần tử đầu tiên của mảng vào ô nhớ $s0 qua lệnh " la $s0, array ", lấy giá trị max kaf giá trị thanh ghi $s2 và khởi tạo nó bằng giá trị phần tử đầu tiên của mảng, sau đó check xem đã nhập hết mảng chưa, nếu chưa ta nhảy xuống nhãn add_element để nhập thêm

```
    jal read_int # lưu phần tử đầu tiên
    add $s2, $zero, $s1 # đặt giá trị max đầu tiên = phần tử đầu tiên
    sw $s1, 0($s0) # lưu vào mảng
    addi $s0, $s0, 4

    beq $t0, $t2, exit_add #check xem số phần tử lớn hơn 1 không, nếu có thì thoát nhập

    addi $t2, $t2, 1 # cập nhật thứ tự phần tử nhập vào
    j add_element
```

- Gọi hàm đọc giá trị phần tử nhập vào và lưu vào thanh ghi $s1, sau đó so sánh với giá trị max hiện tại xem có lớn hơn không (giá trị thanh ghi $s2) nếu lớn hơn thì cập nhật giá trị thanh ghi $s2 bằng giá trị thanh ghi $s1, sau đó lưu vào ô nhớ có địa chỉ là giá trị thanh ghi $s0, cập nhật địa chỉ ô nhớ tiếp theo bằng cách cộng thang ghi $s0 cho 4vaf tăng biến đếm thứ tự $t2 thêm 1. So sánh biến đếm phần tử $t2 xem đã nhập đủ chưa, nếu chưa thì nhảy lại nhãn "add_element"

```
add_element:
    jal read_int

    slt $t3, $s2, $s1 # check xem giá trị mới nhập có phải giá trị lớn nhất không
    beq $t3, $zero, skip_save_max
    add $s2, $s1, $zero # lưu giá trị lớn nhất hiện thời

    skip_save_max:
    sw $s1, 0($s0) # lưu vào mảng
    addi $s0, $s0, 4

    addi $t2, $t2, 1 # cập nhật thứ tự phần tử nhập vào
    slt $t3, $t0, $t2
    beq $t3, $zero, add_element
```

- In ra giá trị thanh ghi $s2 là giá trị lớn nhất sau khi đã nhập hết mảng

```
exit_add: li $v0, 4
    la $a0, comclu1_prompt
    syscall

    li $v0, 1
    add $a0, $s2, $zero         # in ra giá trị lớn nhất trong mảng
    syscall

    li $v0, 4
    la $a0, enter_prompt
    syscall
```

- Nhập các giá trị m, M

```
89  #Nhập các giá trị m,M vào t1, t2
90      # In prompt số m
91      li $v0, 4
92      la $a0, strm
93      syscall
94
95      # nhập m
96      li $v0, 5
97      syscall
98      add $t1, $zero, $v0
99
100     # In prompt số M
101     li $v0, 4
102     la $a0, strM
103     syscall
104
105     # Nhập M
106     li $v0, 5
107     syscall
108     add $t2, $zero, $v0
```

- Đặt giá trị thanh ghi $t3 = 0 trượng trưng cho giá trị biến đếm count, giá trị thanh ghi $t6 = 0 để đếm số phần tử trong mảng đã được xét là lấy địa chỉ ô nhớ đầu tiên của mảng

```
110     # bắt đầu xét
111     add $t3, $zero, $zero # count = 0
112     addi $t6, $zero, 0 # temp = 0 -> số phần tử của mảng đã được xé
113     la $s0, array # lấy phần tử đầu tiên của mảng
```

- Lặp qua tất cả các phần tử trong mảng, nếu phần tử đó thỏa mãn lớn hơn m và bé hơn M thì ta tăng giá trị thanh ghi $t3 lên 1, cuối mỗi vòng lặp ta cập nhật lại giá trị địa chỉ phần tử tiếp theo và tăng số lượng phần tử đã xét lên, nếu số lượng phần tử đã xét bằng số phần tử trong mảng thì dừng

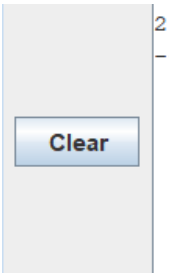
```
114 loop_count: lw $s1, 0($s0)
115     slt $t4, $t1, $s1
116     beq $t4, $zero, next_loop #so sánh với m
117     slt $t4, $s1, $t2
118     beq $t4, $zero, next_loop #so sánh với M
119
120     addi $t3, $t3, 1 # tăng count
121 next_loop: addi $t6, $t6, 1
122     addi $s0, $s0, 4
123     bne $t6, $t0, loop_count
```

- In ra số phần tử trong mảng nằm trong khoảng (m, M) và hỏi người dùng có muốn chạy lại chương trình, nếu không bấm 0 và nếu có bấm 1 số khác còn lại

```
print_result:li $v0, 4
    la $a0, comclu2_prompt
    syscall

    li $v0, 1
    add $a0, $t3, $zero  # in ra số phần tử trong khoảng (m, M)
    syscall

    li $v0, 4
    la $a0, enter_prompt
    syscall

reset: li $v0, 4
    la $a0, prompt_reset # in ra prompt hỏi người dùng có muốn lặp
    syscall

    # nhập lựa chọn
    li $v0, 5
    syscall

    bne $v0, $zero, main

exit:
```

Bài 2:

Lấy 2 mảng ngẫu nhiên làm đầu vào (1,2,3,4,5) và (5,4,3,2,1) , chương trình ra kết quả:



Code:

```
.data
A: .word 1,2,3,4,5
Aend: .word
B: .word  5,4,3,2,1
Bend: .word
equal_prompt: .asciiz "2 mang bang nhau"
not_equal_prompt: .asciiz "2 mang khong bang nhau"
.text
main:
check_arr:la $t0,A #$t0 = Address(A[0])
    la $t1,Aend
    la $s0,B #$s0 = Address(B[0])
    la $s1,Bend

 #check độ dài 2 mảng
    subu $t2, $t1, $t0  # Tính độ dài của mảng A
    subu $s2, $s1, $s0  # Tính độ dài của mảng 8
    bne $t2, $s2, not_equal
# sắp xếp 2 mảng
    la $a0,A #$a0 = Address(A[0])
    la $a1,Aend
    addi $a1,$a1,-4 #$a1 = Address(A[n-1])
    jal sort #sort
    la $a0,B #$s0 = Address(B[0])
    la $a1,Bend
    addi $a1,$a1,-4 #$a1 = Address(B[n-1])
    jal sort #sort
    nop

 check_array: la $t0,A #$t0 = Address(A[0])
    la $t1,Aend
```

```
    la $s0,B #$s0 = Address(B[0])
    la $s1,Bend
 loop_check_arr:
 #lấy 2 phần tử tương đương index 2 mảng cần check hiện tại
     lw $t3,0($t0)
     lw $s3, 0($s0)
 #check 2 phần tử bằng nhau không, nếu ko thoát vòng lặp
     bne $s3, $t3, not_equal
 #tiến hành cập nhật phần tử cần check hiện tại
     addi $t0, $t0, 4
     addi $s0, $s0, 4
 # check xem đã là phần tử cuối cùng
     bne $s0, $s1,  loop_check_arr

equal:li $v0, 4          # thông báo 2 mảng bằng nhau
    la $a0, equal_prompt     # Load address of prompt into $a0
    syscall            # Call operating system to perform operation
    j exit

not_equal:li $v0, 4          # Thông báo 2 mảng không bằng nhau
    la $a0, not_equal_prompt     # Load address of prompt into $a0
    syscall            # Call operating system to perform operation
    j exit

sort: beq $a0,$a1,done #check xem hết phần tử chưa, nếu hết thì thoát vòng lặp
    j max #call the max procedure
after_max: lw $t0,0($a1) #lấy giá trị phần tử cuối cùng chưa xét đến của mảng
#swap 2 giá trị cuối và gia trị lớn nhất
    sw $t0,0($v0) #lưu giá trị $t0 vào ô có địa chỉ v0
    sw $v1,0($a1) #lưu giá trị $v0 vào ô có địa chỉ a1
    addi $a1,$a1,-4 #cập nhật lại giá trị ô nhớ cuối cùng của mảng chưa xét đến
    j sort # tiếp tục vòng sort
```

```
done: jr $ra

max: addi $v0,$a0,0 #khởi tạo biến chứa địa chỉ ô nhớ có phần tử max
    lw $v1,0($v0) #lưu giá trị phần tử đầu tiên của vào v1 ( max = v1)
    addi $t0,$a0,0 #địa chỉ phần tử đầu tiên của mảng vào t0
loop:beq $t0,$a1,ret #if next=last, return
    addi $t0,$t0,4 #lưu địa chỉ phần tử tiếp theo
    lw $t1,0($t0) #load giá trị tiếp theo vào t1
    slt $t2,$t1,$v1 #(next)<(max) ?
    bne $t2,$zero,loop #if (next)<(max), repeat
    addi $v0,$t0,0 #địa chỉ ô nhớ có phần tử max
    addi $v1,$t1,0 #lưu giá trị max hiện tại
    j loop #tiếp tục loop
ret: j after_max

exit:
```

Giải thích:

- Khai báo 2 mảng và các prompt cần thiết

```
1  .data
2  A: .word 1,2,3,4,5
3  Aend: .word
4  B: .word  5,4,3,2,1
5  Bend: .word
6  equal_prompt: .asciiz "2 mang bang nhau"
7  not_equal_prompt: .asciiz "2 mang khong bang nhau"
```

- Kiểm tra xem 2 mảng có số phần tử bằng nhau không bằng cách lấy địa chỉ phần tử cuối trừ phần tử đầu tiên của cả 2 mảng, nếu không bằng thì nhảy tới nhãn not_equal và in ra prompt không bằng

```
10  check_arr:la $t0,A #$t0 = Address(A[0])
11      la $t1,Aend
12      la $s0,B #$s0 = Address(B[0])
13      la $s1,Bend
14
15   #check độ dài 2 mảng
16      subu $t2, $t1, $t0  # Tính độ dài của mảng A
17      subu $s2, $s1, $s0  # Tính độ dài của mảng B
18      bne $t2, $s2, not_equal
```

- Sử dụng selection sort cho việc sắp xếp mảng theo thứ tự tăng dần. Đầu tiên ta lưu địa chỉ phần tử đầu tiên của mảng vào thanh ghi $a0, địa chỉ phần tử cuối cùng của mảng vào thanh ghi $a1, sau đó kiểm tra 2 địa chỉ có trùng nhau không, nếu có thì mảng chỉ có 1 phần tử và ta không cần sắp xếp

```
50  sort: beq $a0,$a1,done #check xem hết phần tử chưa, nếu hết thì thoát vòng lặp
51      j max #call the max procedure
```

- Đặt giá trị $v0 là giá trị địa chỉ phần tử lớn nhất mảng và khởi tạo nó là địa chỉ phần tử đầu tiên của mảng, $v1 chứa giá trị lớn nhất mảng, $t0 là địa chỉ phần tử duyệt đến. Sau đó ta duyệt hết tất cả các phần tử trong mảng để tìm ra phần tử lớn nhất và lưu địa chỉ phần tử đó vào thanh ghi $v0, giá trị vào thanh ghi $v1

```
66  max: addi $v0,$a0,0 #khởi tạo biến chứa địa chỉ ô nhớ có phần tử max
67      lw $v1,0($v0) #lưu giá trị phần tử đầu tiên của vào v1 ( max = v1)
68      addi $t0,$a0,0 #địa chỉ phần tử đầu tiên của mảng vào t0
69  loop:beq $t0,$a1,ret #if next=last, return
70      addi $t0,$t0,4 #lưu địa chỉ phần tử tiếp theo
71      lw $t1,0($t0) #load giá trị tiếp theo vào t1
72      slt $t2,$t1,$v1 #(next)<(max) ?
73      bne $t2,$zero,loop #if (next)<(max), repeat
74      addi $v0,$t0,0 #địa chỉ ô nhớ có phần tử max
75      addi $v1,$t1,0 #lưu giá trị max hiện tại
76      j loop #tiếp tục loop
77  ret: j after_max
```

- Đổi chỗ phần tử cuối cùng chưa được xét và phần tử lớn nhất của mảng cho nhau, sau đó cập nhật lại phần tử cuối cùng chưa được xét đó và tiếp tục quá trình tìm phần tử lớn nhất của mảng đến khi nào toàn bộ phần tử trong mảng được xét

```
56  sort: beq $a0,$a1,done #check xem hết phần tử chưa, nếu hết thì thoát vòng lặp
57      j max #call the max procedure
58  after_max: lw $t0,0($a1) #lấy giá trị phần tử cuối cùng chưa xét đến của mảng
59  #swap 2 giá trị cuối và gia trị lớn nhất
60      sw $t0,0($v0) #lưu giá trị $t0 vào ô có địa chỉ v0
61      sw $v1,0($a1) #lưu giá trị $v0 vào ô có địa chỉ a1
62      addi $a1,$a1,-4 #cập nhật lại giá trị ô nhớ cuối cùng của mảng chưa xét đế
63      j sort # tiếp tục vòng sort
64  done: jr $ra
```

- Sắp xếp 2 mảng đã cho theo thứ tự lớn dần

```
# sắp xếp 2 mảng
    la $a0,A #$a0 = Address(A[0])
    la $a1,Aend
    addi $a1,$a1,-4 #$a1 = Address(A[n-1])
    jal sort #sort
    la $a0,B #$s0 = Address(B[0])
    la $a1,Bend
    addi $a1,$a1,-4 #$a1 = Address(B[n-1])
    jal sort #sort
    nop
```

- Duyệt 2 mảng song song, kiểm tra từng phần tử của mảng xem có trùng nhau không, nếu xuất hiện 2 phần tử 2 mảng cùng thứ tự nhưng khác nhau thì nhảy xuống nhãn not_equal và in ra prompt 2 mảng không bằng nhau, nếu tất cả phần tử trong 2 mảng đều giống nhau và duyệt đến phần tử cuối cùng (“bne $s0, $s1, loop_check_arr”) thì nhảy xuống nhãn equal và in ra prompt 2 mảng bằng nhau

```
check_array: la $t0,A #$t0 = Address(A[0])
   la $t1,Aend
   la $s0,B #$s0 = Address(B[0])
   la $s1,Bend
loop_check_arr:
#lấy 2 phần tử tương đương index 2 mảng cần check hiện tại
    lw $t3,0($t0)
    lw $s3, 0($s0)
#check 2 phần tử bằng nhau không, nếu ko thoát vòng lặp
    bne $s3, $t3, not_equal
#tiến hành cập nhật phần tử cần check hiện tại
    addi $t0, $t0, 4
    addi $s0, $s0, 4
# check xem đã là phần tử cuối cùng
    bne $s0, $s1,  loop_check_arr

equal:li $v0, 4           # thông báo 2 mảng bằng nhau
    la $a0, equal_prompt     # Load address of prompt into $a0
    syscall            # Call operating system to perform operation
    j exit

not_equal:li $v0, 4           # Thông báo 2 mảng không bằng nhau
    la $a0, not_equal_prompt     # Load address of prompt into $a0
    syscall            # Call operating system to perform operation
    j exit
```