

# BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

## Lab 5 : Nhập xuất dữ liệu với hàm ECALL, xử lý chuỗi ký tự

Họ tên	MSSV
Phạm Minh Hiền	20235705

### Assignment 1:

Tạo project thực hiện chương trình trong Home Assignment 1. Khởi tạo bộ giá trị mới cho mảng, dịch và nạp lên mô phỏng. Chạy chương trình từng bước một và quan sát sự thay đổi các thanh ghi để kiểm nghiệm chương trình hoạt động đúng với thuật toán.

### SOURCE CODE :

```
.data
```

```
A: .word -5, 6, -8, 9, -2, 8
```

```
msg: .asciz "Do dai mang con co tong lon nhat: "
```

```
msg1: .asciz "Tong lon nhat: "
```

```
newline: .asciz "\n"
```

```
.text
```

```
main:
```

```
    la a0, A
```

```
    li a1, 6
```

```
    j mspfx
```

```
continue:
```

```
    li a7, 4 # ln ra msg
```

```
    la a0, msg
```

```
    ecall
```

```
    li a7, 1 # ln ra do dai cua mang con co tong lon nhat (s0)
```

```
    addi a0, s0, 0
```

```
    ecall
```

```
    li a7, 4 # ln ra dau xuong dong
```

```

    la a0, newline
    ecall

    li a7, 4 # ln ra msg1
    la a0, msg1
    ecall

    li a7, 1 # ln ra tong lon nhat (s1)
    addi a0, s1, 0
    ecall

exit:
    li a7, 10
    ecall

end_of_main:

mspfx:
    li s0, 0
    li s1, 0x80000000
    li t0, 0
    li t1, 0

loop:
    add t2, t0, t0
    add t2, t2, t2
    add t3, t2, a0
    lw t4, 0(t3)
    add t1, t1, t4
    blt s1, t1, mdfy
    j next

mdfy:
    addi s0, t0, 1

```

```

    addi s1, t1, 0

next:
    addi t0, t0, 1
    blt t0, a1, loop

done:
    j continue

mspfx_end:

```

- Đoạn code có thêm phần in kết quả để dễ quan sát.
- Ta khởi tạo với bộ giá trị mới :  
A: .word -5, 6, -8, 9, -2, 8  
+ Bộ dữ liệu gồm 6 phần tử với chuỗi tiền tố dài nhất là 6 có tổng lớn nhất là 8.
- Quan sát sự thay đổi của thanh ghi :

	t2	t3	t4	s0	s1	t0
Lần lặp 1	0x00000000	0x10010000	0xffffffffb	0x00000001	0xffffffffb	0x00000001
Lần lặp 2	0x00000004	0x10010004	0x00000006	0x00000002	0x00000001	0x00000002
Lần lặp 3	0x00000008	0x10010008	0xffffffff8	0x00000002	0x00000001	0x00000003
Lần lặp 4	0x0000000c	0x1001000c	0x00000009	0x00000004	0x00000002	0x00000004
Lần lặp 5	0x00000010	0x10010010	0xffffffffe	0x00000004	0x00000002	0x00000005
Lần lặp 6	0x00000014	0x10010014	0x00000008	0x00000006	0x00000008	0x00000006

- + Lần lặp 1:  $s1 = 0x80000000 < t1 = -5$  nên cập nhật  $s1 = -5$ ,  $s0 = 1$  và  $t0 = 1$ .
- + Lần lặp 2:  $s1 = -5 < t1 = (-5) + 6$  nên cập nhật  $s1 = -5 + 6 = 1$ ,  $s0 = 2$  và  $t0 = 2$ .
- + Lần lặp 3:  $s1 = 1 > t1 = 1 + (-8)$  nên không cập nhật  $s0$  và  $s1$ , xét điều kiện *next*  $t0 = 2 + 1 = 3 < 6$  nên tiếp tục vòng lặp.
- + Lần lặp 4:  $s1 = 1 < t1 = 1 + (-8) + 9 = 2$  nên cập nhật  $s1 = 2$ ,  $s0 = 4$  và  $t0 = 4$ .
- + Lần lặp 5:  $s1 = 2 > t1 = 2 + (-2)$  nên không cập nhật  $s0$  và  $s1$ , xét điều kiện *next*  $t0 = 5 < 6$  nên lặp tiếp.
- + Lần lặp 6:  $s1 = 2 < t1 = 2 + (-2) + 8 = 8$  nên cập nhật  $s0 = 6$  và  $s1 = 0$ ,  $t0 = 6$  thoả mãn điều kiện kết thúc vòng lặp.
- Kết quả in ra màn hình :

Messages	Run I/O
	Do dai mang con co tong lon nhat: 6 Tong lon nhat: 8

→ Vậy chương trình hoạt động như yêu cầu.

## Assignment 2:

Tạo mới một project thực hiện chương trình trong Home Assignment 2. Khởi tạo bộ giá trị mới cho mảng, dịch và nạp lên mô phỏng. Chạy chương trình từng bước một và quan sát sự thay đổi các thanh ghi để kiểm nghiệm chương trình hoạt động đúng với thuật toán. Viết thêm chương trình con để in ra mảng sau mỗi lượt sắp xếp.

### SOURCE CODE :

```
.data
```

```
A: .word -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55
```

```
Aend: .word
```

```
.text
```

```
main:
```

```
    la a0, A
```

```
    la a1, Aend
```

```
    addi a1, a1, -4
```

```
    j sort
```

```
after_sort:
```

```
    li a7, 10
```

```
    ecall
```

```
end_main:
```

```
sort:
```

```
    beq a0, a1, done
```

```
    j max
```

```
after_max:
```

```
    lw t0, 0(a1)
```

```
    sw t0, 0(s0)
```

```
    sw s1, 0(a1)
```

```
    addi a1, a1, -4
```

```

        j print # In mang
        j sort
done:
        j after_sort
max:
        addi s0, a0, 0
        lw s1, 0(s0)
        addi t0, a0, 0
loop:
        beq t0, a1, ret
        addi t0, t0, 4
        lw t1, 0(t0)
        blt t1, s1, loop
        addi s0, t0, 0
        addi s1, t1, 0
        j loop
ret:
        j after_max
print:
        addi t2, a0, 0 # Sao chep du lieu thanh ghi a0
        addi t3, a1, 0 # Sao chep du lieu thanh ghi a1
        la t0, A
        la t1, Aend # Lay vi tri cuoi cua mang
        addi t1, t1, -4
print_loop:
        bgt t0, t1, endloop
        lw a0, 0(t0) # In tung phan tu

```

```

li a7, 1
ecall

li a0, 32 # In dau cach
li a7, 11
ecall

addi t0, t0, 4 # Chuyen sang phan tu tiep theo
j print_loop

endloop:

li a0, 10 # In newline
li a7, 11
ecall

addi a0, t2, 0 # Khoi phuc du lieu
addi a1, t3, 0 # Khoi phuc du lieu
j sort

```

- Đã bổ sung thêm phần in mảng sau mỗi lượt sắp xếp.
- Khởi tạo bộ dữ liệu :  
A: .word -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55  
Nếu chương trình chạy đúng sẽ ra dãy được sắp xếp là : -7 -2 1 3 5 6 8 9 17 55 58 59 64
- Quan sát sự thay đổi của các thanh ghi :
  - + Lần lặp 1: s0 trở đến A[0], chương trình duyệt qua mảng thông qua các vòng lặp nhỏ hơn và tìm thấy 64 lớn nhất tại A[5]. Lúc này s0 = address(A[5]), và s1 = 64. Chương trình sẽ thực hiện hoán đổi A[5] với A[12] và cập nhật con trỏ a1 = a1 – 4.
  - + Lần lặp 2: s0 = address(A[11]), s1 = 59 giữ nguyên vị trí.
  - + Tương tự với các vòng lặp tiếp theo, chương trình sẽ thực hiện duyệt mảng tìm phần tử lớn nhất và đưa về cuối.
- Kết quả in ra màn hình :

Messages	Run I/O	
	-7 -2 5 1 58 55 17 3 6 9 8 59 64	
	-7 -2 5 1 58 55 17 3 6 9 8 59 64	
	-7 -2 5 1 8 55 17 3 6 9 58 59 64	
	-7 -2 5 1 8 9 17 3 6 55 58 59 64	
	-7 -2 5 1 8 9 6 3 17 55 58 59 64	
	-7 -2 5 1 8 3 6 9 17 55 58 59 64	
Clear	-7 -2 5 1 6 3 8 9 17 55 58 59 64	
	-7 -2 5 1 3 6 8 9 17 55 58 59 64	
	-7 -2 3 1 5 6 8 9 17 55 58 59 64	
	-7 -2 1 3 5 6 8 9 17 55 58 59 64	
	-7 -2 1 3 5 6 8 9 17 55 58 59 64	
	-7 -2 1 3 5 6 8 9 17 55 58 59 64	

→ Có vẻ chương trình chạy như kỳ vọng.

### Assignment 3:

Viết chương trình thực hiện thuật toán sắp xếp nổi bọt (bubble sort).

#### SOURCE CODE :

.data

A: .word -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55

Aend: .word

.text

main:

la a0, A

la a1, Aend

addi a1, a1, -4

j sort

after\_sort:

li a7, 10

ecall

end\_main:

sort:

addi t2, a0, 0 # Sao chep du lieu

addi t3, a1, 0

outer\_loop:

beq t2, t3, done # Kiem tra dieu kien ket thuc vong lap

addi t0, t2, 0

li t6, 0 # Danh dau su thay doi

inner\_loop:

beq t0, t3, end\_inner\_loop # Kiem tra vong lap trong

lw t4, 0(t0)



```
lw t5, 4(t0)
ble t4, t5, no_swap # Neu t4 <= t5 khong doi vi tri
sw t5, 0(t0) # Doi vi tri
sw t4, 4(t0)
li t6, 1
```

no\_swap:

```
addi t0, t0, 4 # Tro den phan tu ke tiep
j inner_loop
```

end\_inner\_loop:

```
beq t6, zero, skip_print # Neu khong co su thay doi nao thi ko in
la t6, A
la t1, Aend
addi t1, t1, -4
```

print\_loop:

```
bgt t6, t1, end_print_loop
lw a0, 0(t6)
li a7, 1
ecall
li a0, 32
li a7, 11
ecall
addi t6, t6, 4
j print_loop
```

end\_print\_loop:

```
li a0, 10
li a7, 11
ecall
```

skip\_print:

```
addi t3, t3, -4 # Lui xuống 1 phần tử
```

```
j outer_loop
```

done:

```
j after_sort
```

- Chạy thử với bộ dữ liệu:

A: .word -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55

- Kết quả :

```
-7 -2 1 5 58 17 3 6 9 8 59 55 64
-7 -2 1 5 17 3 6 9 8 58 55 59 64
-7 -2 1 5 3 6 9 8 17 55 58 59 64
-7 -2 1 3 5 6 8 9 17 55 58 59 64

-- program is finished running (0) --
```

+ Chương trình thực hiện logic bubble sort sắp xếp tăng dần, kiểm tra 2 phần tử liên tiếp nhau trong mảng xem đã đúng vị trí chưa, nếu chưa thì đổi chỗ 2 phần tử.

+ Trong hình ta thấy, trong vòng lặp đầu, chương trình đã đổi chỗ 5 với 1, 64 với 17, 64 với 3, 64 với 6, 64 với 9, 64 với 8, 64 với 59 và 64 với 55. Do đã đi hết mảng nên chương trình lặp lại với mảng đã sắp số 64.

+ Vòng lặp kế tiếp, 58 đã đổi chỗ với 17, 3, 6, 9, 8 và gặp 59. Lúc này do  $58 < 59$  nên tạm coi là đúng vị trí, ta tiếp tục với 59 đổi chỗ cho 55 và đi đến cuối mảng, 59 đã đúng vị trí.

+ Tiếp tục với các vòng lặp còn lại.

+ Ta thấy chương trình hoạt động như dự kiến.

+ Thử với các bộ dữ liệu khác cũng cho kết quả chính xác :

A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

```
-2 5 1 5 6 7 3 6 7 8 8 59 5 |
-2 1 5 5 6 3 6 7 7 8 5 8 59
-2 1 5 5 3 6 6 7 7 5 8 8 59
-2 1 5 3 5 6 6 7 5 7 8 8 59
-2 1 3 5 5 6 6 5 7 7 8 8 59
-2 1 3 5 5 6 5 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59
```

## Assignment 4:

### SOURCE CODE :

.data

A: .word -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55

Aend: .word

.text

main:

la a0, A

la a1, Aend

addi a1, a1, -4

j sort

after\_sort:

li a7, 10

ecall

end\_main:

sort:

addi t0, a0, 4 # Sao chep du lieu

outer\_loop:

bgt t0, a1, done # Xet dieu kien ket thuc vong lap

lw t2, 0(t0) # key = A[i]

addi t1, t0, -4 # j = i - 1

inner\_loop:

blt t1, a0, end\_inner\_loop # Xet dieu kien vong lap

lw t3, 0(t1) # t3 = A[j]

ble t3, t2, end\_inner\_loop # A[j] <= key, ket thuc lap

sw t3, 4(t1) # A[j + 1] = A[j]

```

        addi t1, t1, -4 # j = j - 1
        j inner_loop
end_inner_loop:
        sw t2, 4(t1)    # A[j + 1] = key
        la t4, A
        la t5, Aend
        addi t5, t5, -4

print_loop:
        bgt t4, t5, end_print_loop
        lw a0, 0(t4)
        li a7, 1
        ecall
        li a0, 32
        li a7, 11
        ecall
        addi t4, t4, 4
        j print_loop
end_print_loop:
        li a0, 10
        li a7, 11
        ecall
        addi t0, t0, 4 # i = i + 1
        j outer_loop
done:
        j after_sort

```

- Chạy thử với bộ dữ liệu :  
A: -7, -2, 5, 1, 58, 64, 17, 3, 6, 9, 8, 59, 55
- Kết quả:

Messages	Run I/O
	-7 -2 5 1 58 64 17 3 6 9 8 59 55
	-7 -2 5 1 58 64 17 3 6 9 8 59 55
	-7 -2 1 5 58 64 17 3 6 9 8 59 55
	-7 -2 1 5 58 64 17 3 6 9 8 59 55
	-7 -2 1 5 58 64 17 3 6 9 8 59 55
	-7 -2 1 5 17 58 64 3 6 9 8 59 55
	-7 -2 1 3 5 17 58 64 6 9 8 59 55
Clear	-7 -2 1 3 5 6 17 58 64 9 8 59 55
	-7 -2 1 3 5 6 9 17 58 64 8 59 55
	-7 -2 1 3 5 6 8 9 17 58 64 59 55
	-7 -2 1 3 5 6 8 9 17 58 59 64 55
	-7 -2 1 3 5 6 8 9 17 55 58 59 64
	-- program is finished running (0) --

- Chương trình thực hiện lấy từng phần tử của mảng A và so sánh với các phần tử trước nó, nếu nó nhỏ hơn thì đẩy vị trí lên. Mảng sắp xếp tăng dần.
- Vòng lặp 1, phần tử -2 > -7, đúng vị trí, không có sự thay đổi.
- Vòng lặp 2, phần tử 5 vẫn đúng vị trí.
- Vòng lặp 3, phần tử 1 < 5 nên thay đổi vị trí 2 phần tử này, 1 > -2 nên không đổi vị trí tiếp.
- Tương tự cho các vòng lặp còn lại.  
➔ Vậy chương trình hoạt động như mong muốn.
- Thử với các bộ số khác cũng cho kết quả tương tự :

	5 6 9 2 3
	5 6 9 2 3
Clear	2 5 6 9 3
	2 3 5 6 9
	-- program is finished running (0) --