

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CNTT&TT



BÁO CÁO BÀI TẬP LỚN GIỮA KÌ

Học phần: Thực hành kiến trúc máy tính

Mã học phần: IT3280

Mã lớp: 147795

Đề bài: Bài 4, bài 18

Sinh viên: Trần Khánh Quỳnh

MSSV: 20225762

Mục lục

1. Bài 4	2
Yêu cầu	2
Ý tưởng thực hiện	2
Mã nguồn	3
Ý nghĩa các thanh ghi được sử dụng	8
Kết quả chạy demo	8
2. Bài 18	9
Yêu cầu	9
Ý tưởng thực hiện	10
Mã nguồn	10
Ý nghĩa các thanh ghi được sử dụng	10
Kết quả chạy demo	10

1. Bài 4

Yêu cầu

(4). Create a program to:

- Input an array of integers from the keyboard.
- Find the maximum element of the array
- Calculate the number of elements in the range of (m, M). Range m, M are inputted from the keyboard.

Ý tưởng thực hiện

- Nhập vào số phần tử của mảng, sau đó nhập lần lượt các phần tử của mảng
- Gán giá trị max bằng giá trị đầu tiên của mảng, lần lượt so sánh các phần tử trong mảng với max. Nếu như tìm thấy phần tử lớn hơn phần tử max thì sẽ thay thế max bằng phần tử đó
- Nhập vào m và M. Lần lượt kiểm tra các phần tử trong mảng xem phần tử nào bé hơn M. Nếu như thỏa mãn điều kiện vừa rồi thì kiểm tra tiếp phần tử đó có lớn hơn m không. Nếu thỏa mãn cả hai điều kiện thì phần tử đó thuộc khoảng (m, M).
- Triển khai ý tưởng bằng mã nguồn C:

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("Enter the number of elements: ");
5     int n;
6     scanf("%d", &n);
7     printf("\n");
8     int arr[n];
9     for(int i = 0; i < n; i++){
10         printf("Enter the elements: ");
11         scanf("%d", &arr[i]);
12         printf("\n");
13     }
14
15     int max = arr[0];
16     for(int i = 0; i < n; i++){
17         if(arr[i] > max){
18             max = arr[i];
19         }
20     }
21     printf("The max element: ");
22     printf("%d\n", max);
23 }
```

```

24     int m, M, int count = 0;
25     printf("Enter the lower bound: ");
26     scanf("%d", &m);
27     printf("\n");
28     printf("Enter the upper bound: ");
29     scanf("%d", &M);
30     printf("\n");
31     for(int i = 0; i < n; i++){
32         if(arr[i] < M){
33             if(arr[i] > m){
34                 count++;
35             }
36         }
37     }
38     printf("Number of elements in range: ");
39     printf("%d", count);
40 }

```

Mã nguồn

.data

msg1: .ascii "Enter the number of elements: "

msg2: .ascii "Enter the element: "

msg3: .ascii "The max element: "

msg4: .ascii "Enter the lower bound: "

msg5: .ascii "Enter the upper bound: "

msg6: .ascii "Number of elements in range: "

enter: .ascii "\n"

array: .word 0: 100

.text

Start:

add \$t1, \$0, \$0 # i=0

la \$t2, array # t2 is the address of arr[0]

Input_num_of_elements:

Print msg1

li \$v0, 4

la \$a0, msg1

```

syscall

# Read num_of_elements in the array
li $v0, 5

syscall

move $t0, $v0    # t0 = n

```

Input_elements:

```

# Print msg2

li $v0, 4

la $a0, msg2

syscall

# Read elements in the array

li $v0, 5

syscall

sw $v0, ($t2)

addi $t2, $t2, 4

addi $t1, $t1, 1

blt $t1, $t0, Input_elements

```

Set_max:

```

la $t2, array    # reset t2

add $t1, $0, $0   #reset i = 0

lw $t3, ($t2)     # t3 = A[0] = max

```

Loop_max:

```
lw $t4, ($t2)    # t4 = arr[i]

blt $t3, $t4, Handle  #if arr[i] > max then jump to handle

addi $t1, $t1, 1  #i++

addi $t2, $t2, 4  #Point to the next element

blt $t1, $t0, Loop_max  #i < n then loop

j print_max #else then print the max number
```

Handle:

```
move $t3, $t4    #max = t4 = arr[i]

addi $t1, $t1, 1  #i++

addi $t2, $t2, 4  #Point to the next element

blt $t1, $t0, Loop_max  #i < n then loop
```

print_max:

```
# Print msg3

li $v0, 4

la $a0, msg3

syscall

# Print max element

li $v0, 1

move $a0, $t3

syscall
```

print_enter:

```

        # Print enter

        li $v0, 4

        la $a0, enter

        syscall

input_lowerbound:

        #Print msg4

        li $v0, 4

        la $a0, msg4

        syscall

        #Read the lower bound

        li $v0, 5

        syscall

        move $s0, $v0      #s0 = m

input_upperbound:

        #Print msg5

        li $v0, 4

        la $a0, msg5

        syscall

        #Read the upperbound

        li $v0, 5

        syscall

        move $s1, $v0      #$s1 = M

        la $t2, array      #Reset to the address of arr[0]

        add $t1, $0, $0     #reset i = 0

        add $s2, $0, $0     #s2 = count = 0

count_in_range:

```

```

    lw $t4, ($t2)      #t4 = arr[i]

    blt $t4, $s1, check1    #if arr[i] < M then branch to check1

    j handle2

check1:

    bgt $t4, $s0, handle1    #if arr[i] < M && arr[i] > m then count++

    j handle2

handle1:

    addi $s2, $s2, 1    #count++

    addi $t2, $t2, 4    #point to the next elements

    addi $t1, $t1, 1    #i++

    ble $t1, $t0, count_in_range    #i < n then loop

    j quit

handle2:

    addi $t2, $t2, 4    #point to the next element

    addi $t1, $t1, 1    #i++

    ble $t1, $t0, count_in_range    #i < n then loop

quit:

print_count:

    #Print msg6

    li $v0, 4

    la $a0, msg6

    syscall

    #Print count

    li $v0, 1

    add $a0, $s2, $0

    syscall

```


Ý nghĩa các thanh ghi được sử dụng

\$t0	Số phần tử của mảng
\$t1	Chỉ số i để tham chiếu đến các phần tử của mảng
\$t2	Lưu địa chỉ cơ sở của mảng
\$t3	Giá trị max
\$t4	Giá trị của phần tử trong mảng (Arr[i])
\$v0	Gọi đến các services để thực hiện in ra màn hình, đọc số từ bàn phím,....
\$a0	Tham số cho các services
\$s0	m: cận dưới của khoảng
\$s1	M: cận trên của khoảng
\$s2	Giá trị đếm, đếm số phần tử trong mảng thuộc khoảng đã nhập vào

Kết quả chạy demo

Nhập dữ liệu vào mảng:

The screenshot displays a debugger interface with three main panels:

- Text Segment:** Shows assembly instructions with addresses, codes, and comments. For example, at address 0x0040013c, the instruction is `0x0040013c 0x0040013c 0x0040013c 108: 1 quit`.
- Data Segment:** Shows memory addresses and their corresponding values. For example, at address 0x10010040, the value is 540701806.
- Console Window:** Displays the output of the program, including prompts like "Enter the element: 1" and "The max element: 156".

2.Bài 18

Yêu cầu

(18) Two arrays are called similar if one can be obtained from another by swapping at most one pair of elements in one of the arrays. Given two arrays a and b, check whether they are similar. Example:

- For a = [1, 2, 3] and b = [1, 2, 3], the output should be `areSimilar(a, b) = true`. The arrays are equal, no need to swap any elements.
- For a = [1, 2, 3] and b = [2, 1, 3], the output should be `areSimilar(a, b) = true`. We can obtain b from a by swapping 2 and 1 in b.
- For a = [1, 2, 2] and b = [2, 1, 1], the output should be `areSimilar(a, b) = false`. Any swap of any two elements either in a or in b won't make a and b equal.

Ý tưởng thực hiện

```
1 #include <stdbool.h>
2 #include <stdio.h>
3
4 bool areSimilar(int a[], int b[], int n) {
5     int diffCount = 0, firstDiff = -1, secondDiff = -1;
6     for (int i = 0; i < n; i++) {
7         if (a[i] != b[i]) {
8             diffCount++;
9             if (firstDiff == -1) firstDiff = i;
10            else secondDiff = i;
11        }
12    }
13    if (diffCount == 0 || (diffCount == 2 && a[firstDiff] == b[secondDiff] && a[secondDiff] == b[firstDiff])) {
14        return true;
15    }
16    return false;
17 }
18
19 int main() {
20     int n; scanf ("%d", &n); int a[n], b[n];
21     for(int i = 0; i < n; i++){
22         scanf("%d", &a[i]);
23     }
24     for(int i = 0; i < n; i++){
25         scanf("%d", &b[i]);
26     }
27
28     if (areSimilar(a, b, n)) {
29         printf("areSimilar(a, b) = true\n");
30     } else {
31         printf("areSimilar(a, b) = false\n");
32     }
33     return 0;
34 }
35
```

-Đầu tiên nhập n là số phần tử, nhập các phần tử của mảng a và b.

-Kiểm tra xem mảng a và mảng b có giống nhau không (có thể đổi nhiều nhất 1 cặp phần tử trong mảng):

+Đầu tiên ta đặt biến diffCount là biến đếm xem có bao nhiêu phần tử khác nhau. Mảng a và mảng b nếu như có biến diffCount khác 0 hoặc khác 2 thì chắc chắn sẽ không thể giống nhau được (nhiều hơn 1 cặp phần tử có thể hoán đổi vị trí).

+Nếu như biến diffCount bằng 0 tức là mảng a và mảng b giống nhau hoàn toàn → kết quả sẽ là “areSimilar(a,b) = true”

+Nếu như biến diffCount bằng 2 tức là trong a và b có 2 phần tử đang khác nhau. Tuy nhiên ta phải kiểm tra điều kiện: nếu như a firstDiff - phần tử khác nhau đầu tiên của mảng a bằng với b secondDiff - phần tử khác nhau thứ 2 của mảng b, và b firstDiff - phần tử khác nhau đầu tiên của mảng b bằng với a secondDiff - phần tử khác nhau thứ 2 của mảng a, tức là cặp phần tử có thể hoán đổi cho nhau thì mới thỏa mãn điều kiện giống nhau của hai mảng.

Mã nguồn

```
.data
true_msg: .asciiz "areSimilar(a,b) = true"
false_msg: .asciiz "areSimilar(a,b) = false"
```

```

enter: .asciiz "\n"
msg1: .asciiz "Input number of elements: "
msg2: .asciiz "Input the array a: "
msg3: .asciiz "Input the array b: "
array_a: .word 0:100
array_b: .word 0:100

.text
main:
#Print msg1
    li $v0, 4
    la $a0, msg1
    syscall
#Read the number of elements
    li $v0, 5
    syscall
    move $s0, $v0          #s0 = n

    la $t0, array_a        #Store address of a to t0
    la $t1, array_b        #Store address of b to t1
    add $t3, $0, $0        #i = t3 = 0
Input_a:
    li $v0, 4
    la $a0, msg2
    syscall
    li $v0, 5              #Read the elements of array a
    syscall
    sw $v0, ($t0)          #Store to address of array a
    addi $t0, $t0, 4        #Point to next address to store next elements
    addi $t3, $t3, 1        #Move to next elements
    blt $t3, $s0, Input_a  #If i < n then loop, continue to input array a
Reset_a:
    add $t3, $0, $0        #Reset i to 0
    la $t0, array_a        #Reset t0 point to the address of array a
Input_b:
    li $v0, 4              #Print msg3
    la $a0, msg3
    syscall
    li $v0, 5              #Read the elements of array b
    syscall
    sw $v0, ($t1)          #Store the address of array b
    addi $t1, $t1, 4        #Point to the next address to store the next
elements
    addi $t3, $t3, 1        #Move to next elements
    blt $t3, $s0, Input_b  #If i < n then loop, continue to input array b
Reset_b:
    add $t3, $0, $0        #Reset i to 0
    la $t1, array_b        #Reser t1 point to the address of array b

```

```

#Check if a and b is similar or not
areSimilar:
    add $s1, $0, $0          #s1 = diffCount = 0
    addi $s2, $0, -1         #s2 = firstDiff = -1
    addi $s3, $0, -1         #s3 = secondDiff = -1
    addi $t4, $0, -1         #t4 = const = -1
    addi $t5, $0, 2          #t5 = const = 2

loop:
    lw $s4, ($t0)            #s4 = a[i]
    lw $s5, ($t1)            #s5 = b[i]
    bne $s4, $s5, handle

move_up:
    addi $t0, $t0, 4
    addi $t1, $t1, 4
    addi $t3, $t3, 1         #i++
    blt $t3, $s0, loop
    j quit_loop

handle:
    addi $s1, $s1, 1         #diffCount++
    bne $t4, $s2, else       #firstDiff != -1 then else
    add $s2, $t3, $0         #firstDiff = i
    j move_up

else:
    add $s3, $t3, $0         #secondDiff = i
    j move_up

quit_loop:
    add $t3, $0, $0          #Reset i = 0
    la $t0, array_a          #Reser t0 point to the 1st element of array a
    la $t1, array_b          #Reset t1 point to the 1st element of array b

check:
    beq $s1, $0, result_true
    bne $s1, $t5, result_false
    sll $s2, $s2, 2          #firstDiff = firstDiff * 4
    sll $s3, $s3, 2          #secondDiff = secondDiff * 4
    add $t6, $t0, $s2        #address of a[firstDiff]
    add $s6, $t0, $s3        #address of a[secondDiff]
    add $t7, $t1, $s2        #address of b[firstDiff]
    add $s7, $t1, $s3        #address of b[secondDiff]
    lw $t6, ($t6)            #t6 = a[firstDiff]
    lw $s6, ($s6)            #s6 = a[secondDiff]
    lw $t7, ($t7)            #t7 = b[firstDiff]
    lw $s7, ($s7)            #s7 = b[secondDiff]
    bne $t6, $s7, result_false
    bne $s6, $t7, result_false
    j result_true

result_true:

```

```

        li $v0, 4
        la $a0, true_msg
        syscall
    j end
result_false:
    li $v0, 4
    la $a0, false_msg
    syscall
end:
    li $v0, 10
    syscall

```

Ý nghĩa các thanh ghi được sử dụng

\$v0	Gọi đến các dịch vụ nhập số, in ra màn hình,...
\$a0	Tham số cho các dịch vụ
\$t0	Địa chỉ của thành phần đầu tiên trong mảng a
\$t1	Địa chỉ của thành phần đầu tiên trong mảng b
\$t3	Biến i để thực hiện các vòng lặp
\$t4	Hằng số -1
\$t5	Hằng số 2
\$s0	Biến n : số phần tử của mảng
\$s1	diffCount: đếm xem có bao nhiêu phần tử khác nhau giữa 2 mảng a và b
\$s2	firstDiff: vị trí trong mảng của phần tử đầu tiên khác nhau
\$s3	secondDiff: vị trí trong mảng của phần tử thứ 2 khác nhau
\$t6	Dùng để lưu địa chỉ của a[firstDiff], sau

	đó lưu giá trị a[firstDiff]
\$s6	Dùng để lưu địa chỉ của a[secondDiff], sau đó lưu giá trị a[secondDiff]
\$t7	Dùng để lưu địa chỉ của b[firstDiff], sau đó lưu giá trị b[firstDiff]
\$s7	Dùng để lưu địa chỉ của b[secondDiff], sau đó lưu giá trị b[secondDiff]

Kết quả chạy demo

TH1: a = [1,2,3]; b = [1,2,3]. Kết quả là giống nhau

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0040013c	0x00100050	0x00400140	122: 3 result true
	0x00400140	0x24020004	addiu \$2,\$0,4	125: 13 \$r0, 4
	0x00400144	0x3c011001	lui \$1,4097	126: 1a \$a0, true msg
	0x00400148	0x34240000	ori \$4,\$1,0	
	0x0040014c	0x0000000c	syscall	127: syscall
	0x00400150	0x00100059	0x00400164	128: 3 end
	0x00400154	0x24020004	addiu \$2,\$0,4	130: 1i \$r0, 4
	0x00400158	0x3c011001	lui \$1,4097	131: 1a \$a0, false msg
	0x0040015c	0x34240017	ori \$4,\$1,23	
	0x00400160	0x0000000c	syscall	132: syscall
	0x00400164	0x2402000a	addiu \$2,\$0,10	134: 1i \$r0, 10
	0x00400168	0x0000000c	syscall	135: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010200	0	1	2	3	0	0	0	0
0x10010220	0	0	0	0	0	0	0	0
0x10010240	0	0	0	0	0	0	0	0
0x10010260	0	0	0	0	0	0	0	0
0x10010280	0	0	0	0	0	0	0	0
0x100102a0	0	0	0	0	0	0	0	0
0x100102c0	0	0	0	0	0	0	0	0

Mars Messages

Run I/O

Input number of elements: 3
Input the array a: 1
Input the array a: 2
Input the array a: 3
Input the array b: 1
Input the array b: 2
Input the array b: 3
areSimilar(a,b) = true
-- program is finished running --

Registers

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	268500992
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	268501108
\$t1	9	268501508
\$t2	10	0
\$t3	11	0
\$t4	12	-1
\$t5	13	2
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	0
\$s2	18	-1
\$s3	19	-1
\$s4	20	3
\$s5	21	3
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479348
\$fp	30	0
\$ra	31	0
pc		4194668
hi		0
lo		0

TH2: a = [1,2,3]; b = [2,1,3]. Kết quả vẫn là giống nhau vì có thể hoán đổi 2 và 1 ở mảng a để tạo ra mảng b

Name	Number	Value
Zero	0	0
One	1	268500992
Two	2	10
Three	3	0
Four	4	268500992
Five	5	0
Six	6	0
Seven	7	0
Eight	8	268501108
Nine	9	268501508
Ten	10	0
Eleven	11	0
Twelve	12	-1
Thirteen	13	2
Fourteen	14	1
Fifteen	15	2
Sixteen	16	3
Seventeen	17	2
Eighteen	18	0
Nineteen	19	4
Twenty	20	3
Twenty One	21	3
Twenty Two	22	2
Twenty Three	23	1
Twenty Four	24	0
Twenty Five	25	0
Twenty Six	26	0
Twenty Seven	27	0
Twenty Eight	28	268468224
Twenty Nine	29	2147479548
Thirty	30	0
Thirty One	31	0
pc		4194668
hi		0
lo		0

☐ Text Segment

Bkpt	Address	Code	Basic	Source
	4194620	0x08100050j	4194624	59: j result true
	4194624	0x24020004	addiu \$2,\$0,4	102: li \$v0, 4
	4194628	0x3c011001lui	\$1,4097	103: la \$a0, true msg
	4194632	0x34240000ori	\$4,\$1,0	
	4194636	0x00000000	syscall	104: syscall
	4194640	0x08100059j	4194660	105: j end
	4194644	0x24020004	addiu \$2,\$0,4	107: li \$v0, 4
	4194648	0x3c011001lui	\$1,4097	108: la \$a0, false msg
	4194652	0x34240017ori	\$4,\$1,23	
	4194656	0x00000000	syscall	109: syscall
	4194660	0x24020004	addiu \$2,\$0,10	111: li \$v0, 10
	4194664	0x00000000	syscall	112: syscall

☐ Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
269501504	0	2	1	1	0	0	0	0
269501536	0	0	0	0	0	0	0	0
269501568	0	0	0	0	0	0	0	0
269501600	0	0	0	0	0	0	0	0
269501632	0	0	0	0	0	0	0	0
269501664	0	0	0	0	0	0	0	0
269501696	0	0	0	0	0	0	0	0

☐ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

Mars Messages

Run I/O

Clear

```

Input number of elements: 3
Input the array a: 1
Input the array a: 2
Input the array a: 2
Input the array b: 2
Input the array b: 1
Input the array b: 1
areSimilar(a,b) = false
-- program is finished running --
        
```

Name	Number	Value
zero	0	0
at	1	269500992
to	2	10
r1	3	0
ao	4	269501015
a1	5	0
a2	6	0
a3	7	0
io	8	269501108
i1	9	269501508
o	10	0
i3	11	0
a4	12	-1
e5	13	2
e6	14	0
e7	15	0
ao	16	3
e1	17	3
e2	18	0
e3	19	2
e4	20	2
e5	21	1
e6	22	0
e7	23	0
e8	24	0
e9	25	0
io	26	0
i1	27	0
ip	28	269468224
ep	29	2147479548
ep	30	0
ra	31	0
c		4194668
i		0
o		0