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

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:

```c
#include <stdio.h>
int main()
{
    printf("Enter the number of elements: ");
    int n;
    scanf("%d", &n);
    printf("\n");
    int arr[n];
    for(int i = 0; i < n; i++){
        printf("Enter the elements: ");
        scanf("%d", &arr[i]);
        printf("\n");
    }

    int max = arr[0];
    for(int i = 0; i < n; i++){
        if(arr[i] > max){
            max = arr[i];
        }
    }
    printf("The max element: ");
    printf("%d\n", max);
```

```
    int m, M; int count = 0;
    printf("Enter the lower bound: ");
    scanf("%d", &m);
    printf("\n");
    printf("Enter the upper bound: ");
    scanf("%d", &M);
    printf("\n");
    for(int i = 0; i < n; i++){
        if(arr[i] < M){
            if(arr[i] > m){
                count++;
            }
        }
    }
    printf("Number of elements in range: ");
    printf("%d", count);
}
```

Mã nguồn

```
.data

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

msg2: .asciiz "Enter the element: "

msg3: .asciiz "The max element: "

msg4: .asciiz "Enter the lower bound: "

msg5: .asciiz "Enter the upper bound: "

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

enter: .asciiz "\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:
```

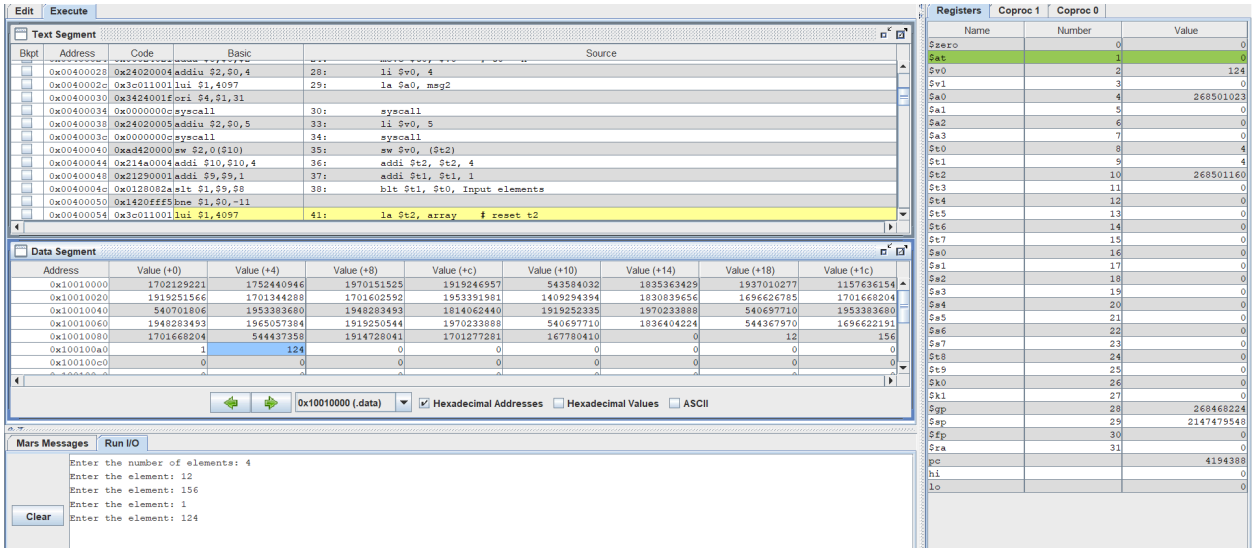
```
	lw $t4, ($t2)	#t4 = arr[i]
	blt $t4, $s1, check1	#if arr[i] < M then branch to check1
	j handle2
check1:
	bgt $t4, $s0, handle1	#if arr[i] < M && arr[i] > m then count++
	j handle2
handle1:
	addi $s2, $s2, 1	#count++
	addi $t2, $t2, 4	#point to the next elements
	addi $t1, $t1, 1	#i++
	ble $t1, $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

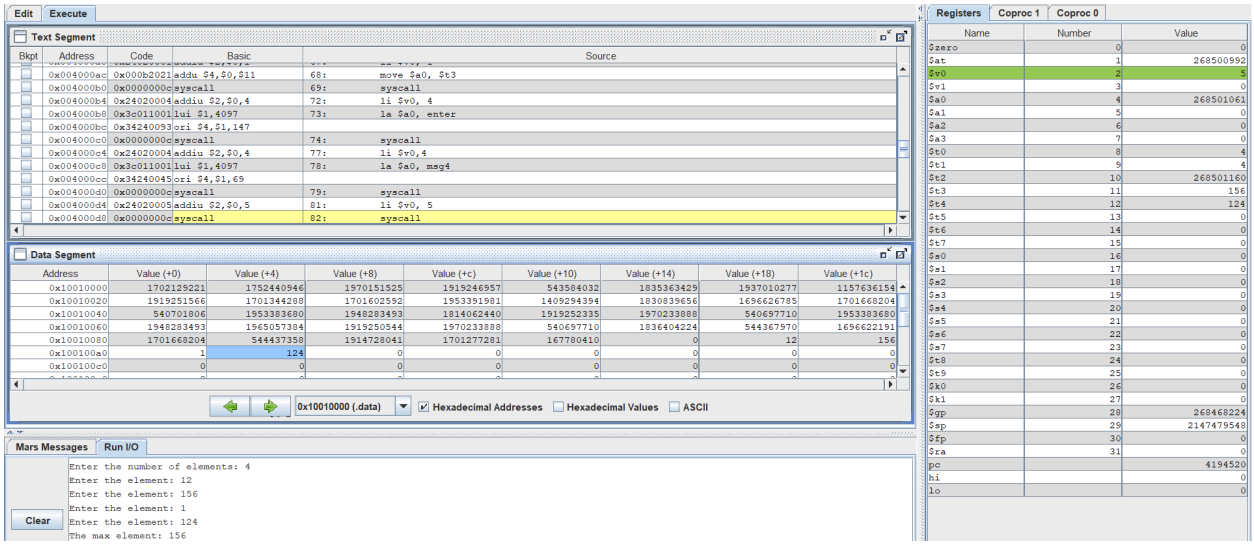
$t0	Số phần tử của mảng
$t1	Chỉ số i để tham chiếu đến các phần tử của mảng
$t2	Lưu địa chỉ cơ sở của mảng
$t3	Giá trị max
$t4	Giá trị của phần tử trong mảng (Arr[i])
$v0	Gọi đến các services để thực hiện in ra màn hình, đọc số từ bàn phím,....
$a0	Tham số cho các services
$s0	m: cận dưới của khoảng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

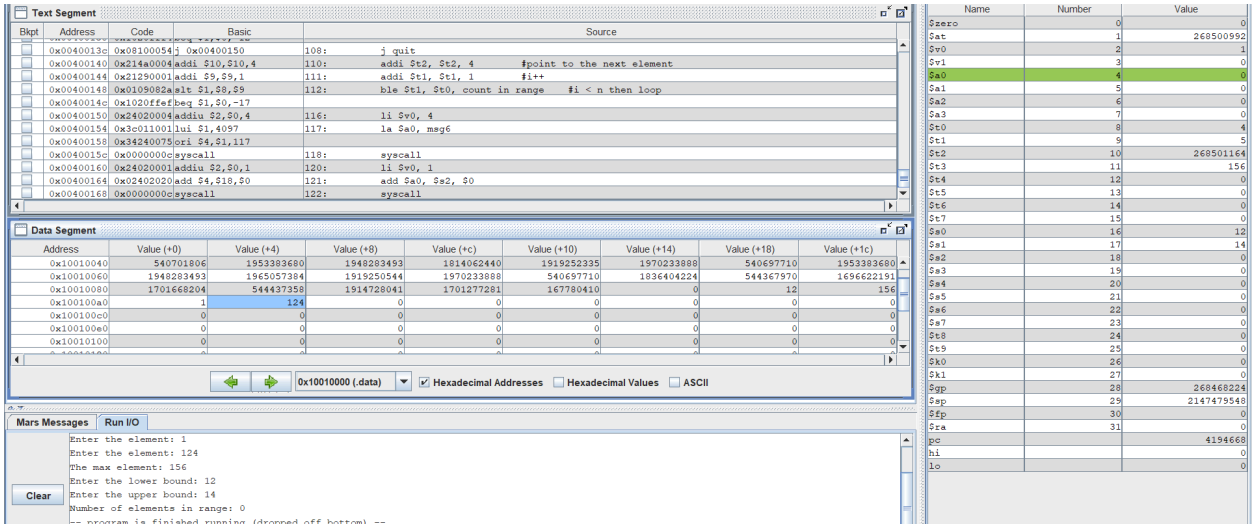
Nhập dữ liệu vào mảng:



Tìm phần tử lớn nhất trong mảng đã nhập vào:



Đếm số phần tử thuộc khoảng (m,M): chọn m = 12, M = 14



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

```c
#include <stdbool.h>
#include <stdio.h>

bool areSimilar(int a[], int b[], int n) {
    int diffCount = 0, firstDiff = -1, secondDiff = -1;
    for (int i = 0; i < n; i++) {
        if (a[i] != b[i]) {
            diffCount++;
            if (firstDiff == -1) firstDiff = i;
            else secondDiff = i;
        }
    }
    if (diffCount == 0 || (diffCount == 2 && a[firstDiff] == b[secondDiff] && a[secondDiff] == b[firstDiff])) {
        return true;
    }
    return false;
}

int main() {
    int n; scanf ("%d", &n); int a[n], b[n];
    for(int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    for(int i = 0; i < n; i++){
        scanf("%d", &b[i]);
    }

    if (areSimilar(a, b, n)) {
        printf("areSimilar(a, b) = true\n");
    } else {
        printf("areSimilar(a, b) = false\n");
    }
    return 0;
}

```

-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 seconDiff - 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
```

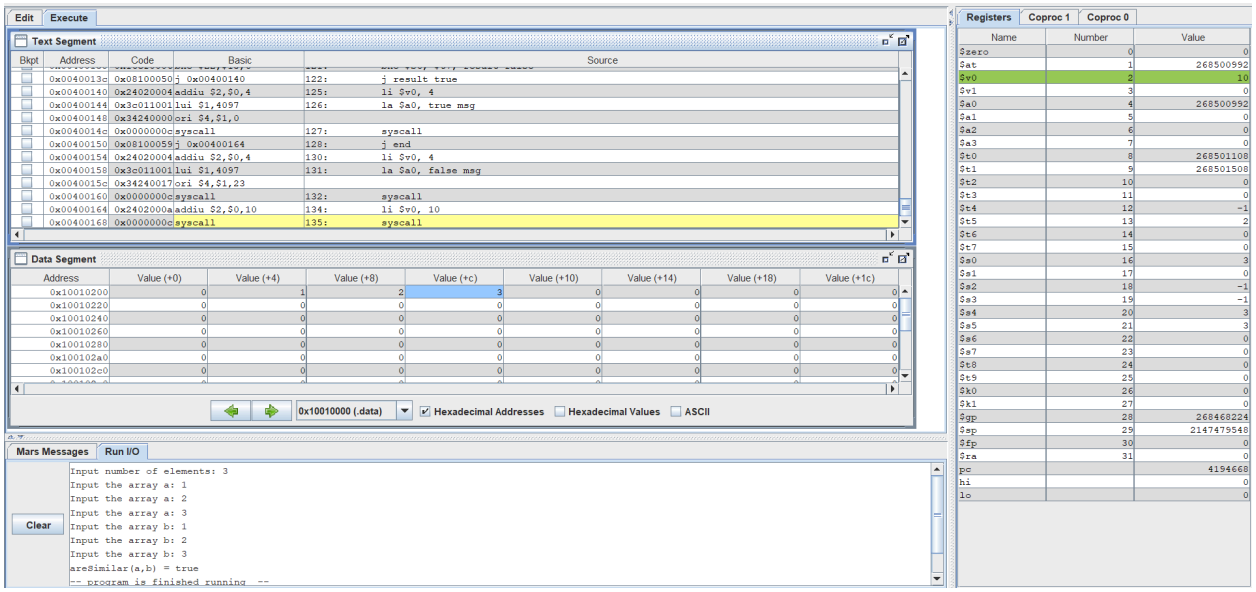
Ý nghĩa các thanh ghi được sử dụng

$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

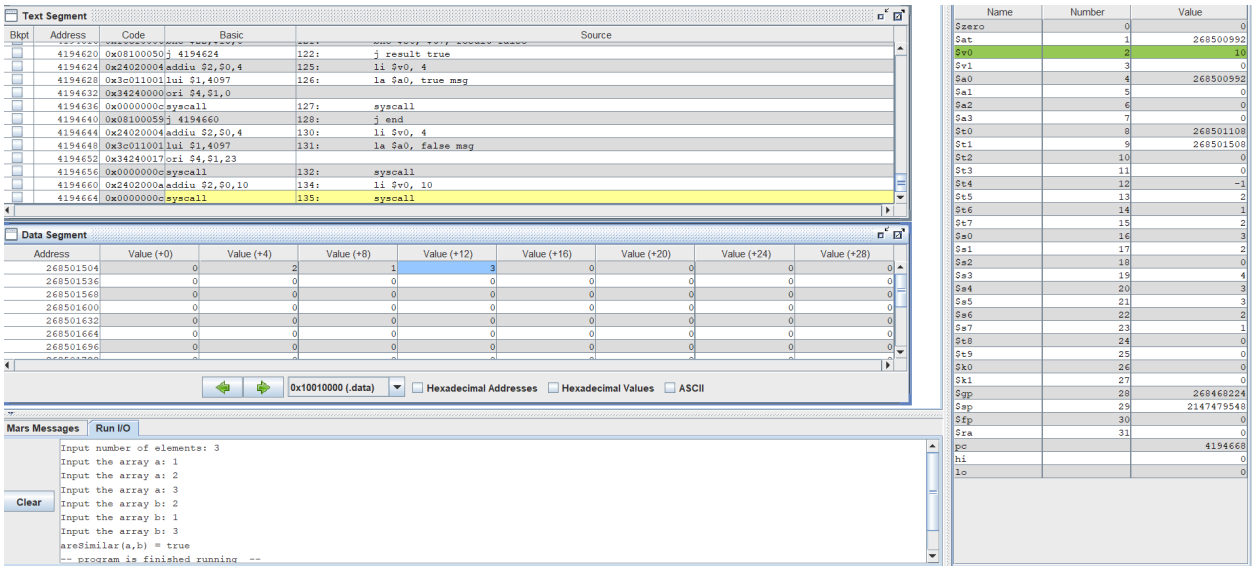
	đó lưu giá trị a[firstDiff]
$s6	Dùng để lưu địa chỉ của a[secondDiff], sau đó lưu giá trị a[secondDiff]
$t7	Dùng để lưu địa chỉ của b[firstDiff], sau đó lưu giá trị b[firstDiff]
$s7	Dùng để lưu địa chỉ của b[secondDiff], sau đó lưu giá trị b[secondDiff]

Kết quả chạy demo

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



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



TH3: a = [1, 2, 2] and b = [2, 1, 1]. Kết quả là không giống nhau