Báo cáo Mini Projects thực hành KTMT giữa kỳ

Họ và tên: Đặng Hồng Minh

MSSV: 20225740

Bài 18:

1. Yêu cầu

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.

2. Thuật toán thực hiện

B1: Nhập dữ liệu: Nhập n (kích thước của hai mảng) và các phần tử của mảng sử dụng vòng lặp, sau đó lưu vào bộ nhớ

B2: So sánh tuần tự từng phần tử của mỗi mảng sử dụng vòng lặp. Nếu phần tử bên mảng a khác với phần tử có chỉ số tương ứng bên mảng b, chương trình sẽ ghi nhận vị trí của sự khác biệt đó theo chỉ số.

B3: Sau khi đã so sánh hết các phần tử của hai mảng, chương trình sẽ kiểm tra số lượng sự khác biệt. Nếu không có sự khác biệt nào hoặc chỉ có hai sự khác biệt và các phần tử tại các vị trí khác biệt đều giống nhau (tức là, hoán vị giữa các phần tử không thay đổi sự tương tự của hai mảng), thì hai mảng được coi là giống nhau. Nếu không, là khác nhau.

B4: In ra kết quả

3. Các thanh ghi sử dụng

$v0: chứa mã lệnh cho syscall

$a0: truyền tham số cho syscall

$s0: chứa kích thước mảng n

$t0, $t1: lưu địa chỉ của mảng a và b

$t3: lưu giá trị của biến đếm i cho vòng lặp

$s1: lưu số lượng khác biệt giữa 2 mảng

$s2, $s3: lưu vị trí (chỉ số) của hai vị trí khác biệt đầu tiên

$t4, $t5: lưu hằng số phục vụ việc so sánh

$s4, $s5: lưu dữ liệu tương ứng từ mảng a và b

$t6, $s6: lưu địa chỉ của hai phần tử khác biệt đầu tiên của mảng a

$t7, $s7: lưu địa chỉ của hai phần tử khác biệt đầu tiên của mảng b

4. Mã chương trình

```
.data
result_similar: .asciiz "The arrays are similar."
result_not_similar: .asciiz "The arrays are not similar."
prompt_size: .asciiz "Enter the size of arrays: "
prompt_a: .asciiz "Input the array a: "
prompt_b: .asciiz "Input the array b: "
array_a: .word 0:100
array_b: .word 0:100
.text
main:
#Print the arrays' size message
      li $v0, 4
      la $a0, prompt_size
      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          #Print message prompt_a
      la $a0, prompt_a
      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 message prompt_b
      la $a0, prompt_b
      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 are 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, result_similar
	syscall
	j end
result_false:
	li $v0, 4
	la $a0, result_not_similar
	syscall
end:
	li $v0, 10
	syscall
```

5. Kết quả chạy chương trình

```
Enter the size of arrays: 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
The arrays are similar.
```

```
Enter the size of arrays: 3
Input the array a: 1
Input the array a: 2
Input the array a: 3
Input the array b: 2
Input the array b: 1
Input the array b: 3
The arrays are similar.
```

```
Enter the size of arrays: 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
The arrays are not similar.
```

```
Enter the size of arrays: 4
Input the array a: 1
Input the array a: 2
Input the array a: 3
Input the array a: 4
Input the array b: 1
Input the array b: 2
Input the array b: 3
Input the array b: 4
The arrays are similar.
```

```
Enter the size of arrays: 4
Input the array a: 2
Input the array a: 1
Input the array a: 3
Input the array a: 4
Input the array b: 1
Input the array b: 2
Input the array b: 3
Input the array b: 4
The arrays are similar.
```

```
Enter the size of arrays: 4
Input the array a: 1
Input the array a: 2
Input the array a: 3
Input the array a: 4
Input the array b: 2
Input the array b: 1
Input the array b: 4
Input the array b: 3
The arrays are not similar.
```

Bài 20:

1. Yêu cầu

Given a string which consists of lower alphabetic characters (a-z), count the number of different characters in it. Example: For s = "cabca", the output should be differentSymbolsNaive(s) = 3. There are 3 different characters a, b and c.

2. Thuật toán thực hiện

B1: Khởi tạo mảng **unique_chars** có 26 phần tử (tương ứng 26 chữ cái alphabet viết thường), mỗi phần tử có giá trị ban đầu là 0 và **unique_count** được khởi tạo giá trị ban đầu là 0 để đếm số chữ cái khác nhau.

B2: Nhập chuỗi, chuỗi này được lưu trong mảng **input_string.**

B3: Xử lý các ký tự trong chuỗi nhập:

- Mỗi ký tự trong chuỗi đều được lấy ra và kiểm tra. Nếu gặp ký tự là null (kết thúc chuỗi), thì dừng.

- Nếu ký tự là chữ cái viết thường, chúng ta sẽ tính chỉ số tương ứng trong mảng **unique_chars.** Sau đó, kiểm tra xem ký tự đã được gặp trước đó chưa. Nếu chưa, đánh dấu ký tự này trong mảng **unique_chars** và tăng số lượng ký tự duy nhất đã tìm thấy lên 1. Nếu rồi, bỏ qua và kiểm tra ký tự tiếp theo.

B4: Xử lý chuỗi xong, in số lượng ký tự khác nhau ra màn hình.

3. Các thanh ghi sử dụng

$v0: chứa mã lệnh cho syscall

$a0: truyền tham số cho syscall

$t0: lưu giá trị của biến đếm i trong vòng lặp

$t1: lưu địa chỉ của ký tự đang xét trong chuỗi **input_string** tại vị trí tương ứng với vòng lặp hiện tại

$t2, $t3: lưu giá trị ASCII của hai ký tự chữ cái viết thường ‘a’ và ‘z’

$t4: lưu giá trị đang đếm số ký tự khác nhau có trong chuỗi

$t5: lưu giá trị kết quả cuối cùng để in ra màn hình

4. Mã chương trình

```
.data
input_string:   .space 100      # Reserve space for input string
unique_chars:   .space 26       # One byte for each lowercase letter (a-z)
unique_count:   .word 0
prompt_string:  .asciiz "Enter a string: "
.text
main:
    # Prompt the user to enter a string
    li $v0, 4
    la $a0, prompt_string
    syscall
    # Read the input string from the keyboard
    li $v0, 8
    la $a0, input_string
    li $a1, 100
    syscall
    # Initialize the loop counter to 0
    li $t0, 0
loop:
    # Load the current character from the input string
```

```
    lb $t1, input_string($t0)
    # Check if the character is null (end of string)
    beqz $t1, done
    # Check if the character is a lowercase letter (a-z)
    li $t2, 'a'         # Load ASCII value of 'a'
    li $t3, 'z'         # Load ASCII value of 'z'
    blt $t1, $t2, next_char   # If character is less than 'a', skip it
    bgt $t1, $t3, next_char   # If character is greater than 'z', skip it
    # Calculate the index in the unique_chars array
    sub $t2, $t1, 'a'
    # Check if the character has been seen before
    lb $t3, unique_chars($t2)
    beqz $t3, add_unique
    # Character already seen, skip adding to unique_chars
    j next_char
add_unique:
    # Mark the character as seen
    sb $t1, unique_chars($t2)
    # Increment the unique count
    lw $t4, unique_count
    addi $t4, $t4, 1
    sw $t4, unique_count
next_char:
    # Move to the next character in the input string
    addi $t0, $t0, 1
    j loop
done:
    # Load the unique count from memory
    lw $t5, unique_count
    # Print the result
```

```
li $v0, 1
move $a0, $t5
syscall
# Exit
li $v0, 10
syscall
```

5. Kết quả chạy chương trình

```
Enter a string: cabca
3
Enter a string: dceffdec
4
Enter a string: awderfafgwds
8
```