TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
- - - - - - - - - - - -

BÁO CÁO

BÀI TẬP LỚN 01

MÔN: KIẾN TRÚC MÁY TÍNH
ĐỀ TÀI: SẮP XẾP CHUỖI THEO THỨ TỰ TĂNG DẦN THEO GIẢI THUẬT QUICK SORT

Giảng viên: Trần Thanh Bình

Sinh viên thực hiện:

1927001 Lê Nhựt Anh

1811775 Lê Tiến Dũng

MỤC LỤC

Đề bài tập lớn:

Cho một chuỗi số nguyên 20 phần tử, sử dụng hợp ngữ assembly MIPS, viết thủ tục sắp xếp, chuỗi đó theo thứ tự tăng dần theo giải thuật quick sort.

1. Giải thuật quick sort

Cho chuỗi số nguyên:

Index	0	1	2	3	4	5	6
Giá trị	10	15	1	2	9	16	11

Ta có nhiều cách để chọn pivot: là số đầu tiên, là số trung vị hoặc là số cuối. Ta sẽ chọn pivot là số đầu cuối => pivot = 11

Ý tưởng giải thuật: chuỗi số sẽ được chia làm 3 vùng ở giữa là key và 2 partition 2 bên (các partition 2 bên chưa được sắp xếp thứ tự)

key

Partition 1	Pivot	Partition 2

giá trị < key key > giá trị

Partition 1: chứa giá trị bé hơn key
Pivot : là giá trị so sánh để sắp xếp
Partition 2: chứa giá trị lớn hơn key

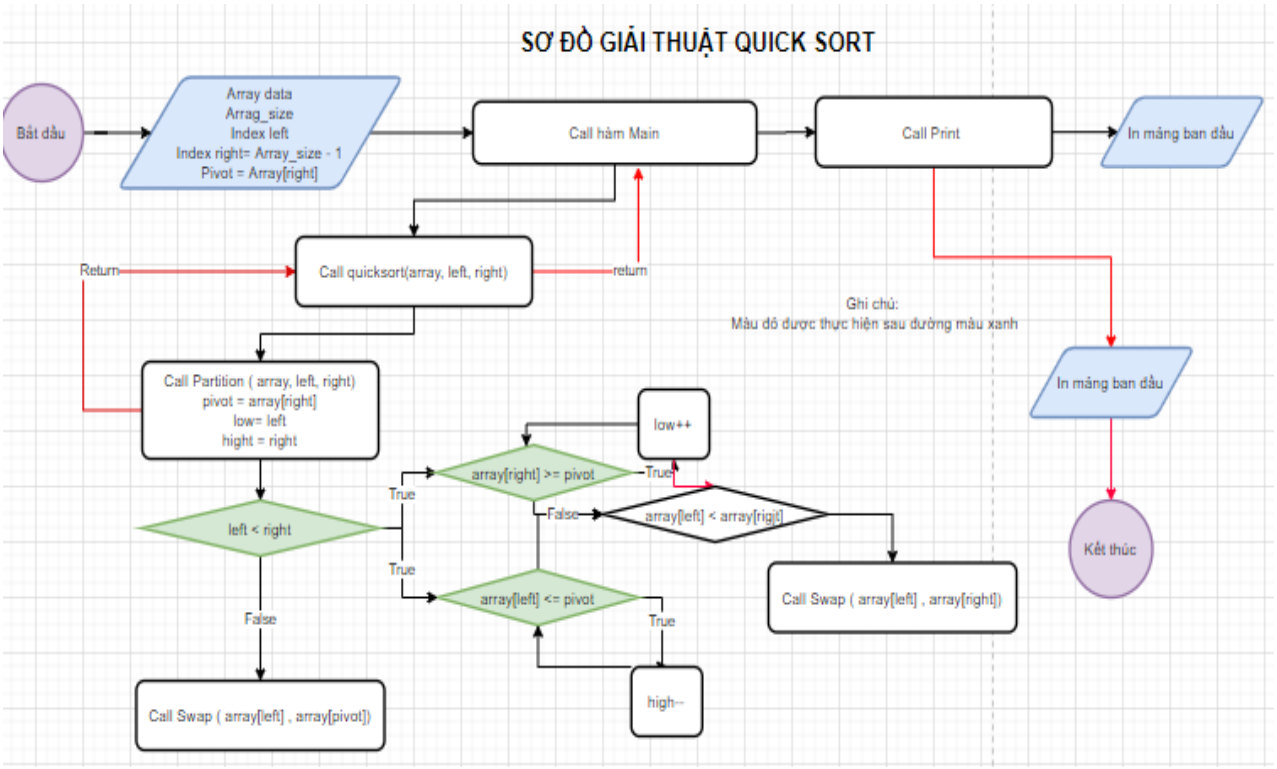
Chương trình bắt đầu với 2 con trỏ lưu lưu index của mảng ta tạm gọi là left và right. Con trỏ left có khởi đầu bằng index đầu tiên của chuỗi và index = 0. Con trỏ right có khỏi đầu bằng index của pivot hay index của mảng trừ đi 1 phần tử do chỉ số index trên mảng bắt đầu từ 0.

Chừng nào left bé hơn right mà giá trị tại index left lớn hơn giá trị pivot và giá trị tại index right bé hơn giá trị pivot thì đổi chỗ hai phần tử left và right. Sau cùng, ta đổi chỗ hai phần tử left và pivot cho nhau, kết thúc lần phân vùng đầu tiên.

Tiếp tục như vậy ta chia nhỏ Partition 1 và Partition 2 thành 3 phần và lặp lại việc so sánh trên để tạo ra phân vùng mới mà ở đó các chỉ số bên trái đều bé hơn pivot 1 và các chỉ số bên phải đều lớn hơn pivot 1…. lặp lại cho đến khi các phần tử đều đã được sắp xếp thức tự.

Partition 1	Pivot 1	Partition 2	Pivot	Partition 1	Pivot 2	Partition 2

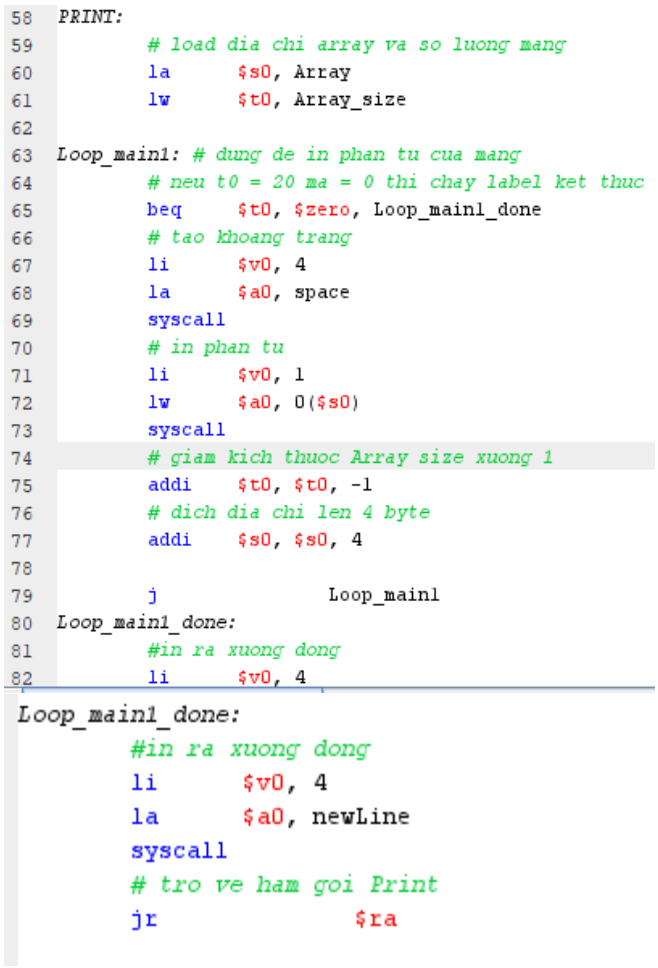
2. Flow Chart:



3. Giải thuật

3.1. Hàm in mảng

Giải thuật	**void print (int *array, int array_size)** **{** **for(int i = 0 ; i < array_size ; i++)** **printf(%d, a[i])** **}**

Assembly	

```
58  PRINT:
59          # load dia chi array va so luong mang
60          la      $s0, Array
61          lw      $t0, Array_size
62
63  Loop_main1: # dung de in phan tu cua mang
64          # neu t0 = 20 ma = 0 thi chay label ket thuc
65          beq     $t0, $zero, Loop_main1_done
66          # tao khoang trang
67          li      $v0, 4
68          la      $a0, space
69          syscall
70          # in phan tu
71          li      $v0, 1
72          lw      $a0, 0($s0)
73          syscall
74          # giam kich thuoc Array size xuong 1
75          addi    $t0, $t0, -1
76          # dich dia chi len 4 byte
77          addi    $s0, $s0, 4
78
79          j               Loop_main1
80  Loop_main1_done:
81          #in ra xuong dong
82          li      $v0, 4
```

```
Loop_main1_done:
        #in ra xuong dong
        li      $v0, 4
        la      $a0, newLine
        syscall
        # tro ve ham goi Print
        jr              $ra
```

3.2. Hàm Quick Sort

Giải thuật	**void quickSort(int arr[], int low, int high)** **{** ** if (low < high)** ** {** ** int pi = partition(arr, low, high);** ** quickSort(arr, low, pi - 1);** ** quickSort(arr, pi + 1, high);** ** }** **}**
Assembly	

```
QuickSort:
        # cap vung nho stack farm
        add     $sp, $sp, -20   # khoi tao stack

        sw      $a0, 0($sp)             # luu doi so vao stacj stack
        sw      $a1, 4($sp)             #low
        sw      $a2, 8($sp)             #high
        sw      $s1, 12($sp)            # luu lai gia tri cua mang
        sw      $ra, 16($sp)            # luu dia chi tra ve cua thanh ghi ra vi trong ham co goi ham khac

        bge     $a1, $a2, EndQuickSort  # neu (low >= high) ket thuc QuickSort, nhay den label EndQuickSort
                                        # nguoc lai neu low < high tiep tuc goi ham partition de phan vung
        jal     partition               # goi ham partition

        addu    $s4, $v0,0              # tra ve pivot moi (p), dua gia tri tra ve vao v0


        lw      $a1, 4($sp)             # a1 = low
        subi    $a2, $s4, 1     # a2 = pivot - 1
        jal     QuickSort               # goi ham QuickSort
EndQuickSort:
         lw      $a0, 0($sp)             # tai doi so duoc luu trong stack ve thanh ghi
         lw      $a1, 4($sp)             # Pop gia tri thanh ghi da luu vao stack luc dau
         lw      $a2, 8($sp)
         lw      $s1, 12($sp)
         lw      $ra, 16($sp)
         addi    $sp, $sp, 20    # tra lai stack farm da cap
         jr      $ra                     # tro lai ham goi bang dia chi duoc luu trong thanh ghi
```

3.3. Hàm Partition

Giải thuật

```
int partition(int arr[], int low, int high)
{
        int pivot = arr[high]; // pivot
        int i = (low - 1);
        int left = low;
        int right = high - 1;
        while (left <= right) {
        while (arr[left] < pivot)
        {
          left++;
        }
        while (arr[right] > pivot)
        {
          right--;
        }
        if (left <= right) {
        swap(&arr[left], &arr[right]);
        left++;
        right--;
    }
  }
        swap(&arr[left], &arr[high]);
```

	return left;
Assembly	

```
partition:
        # Khoi tao stack de luu cac doi so, cap phat vung nho (xem giai thuat can luu 4 gia tri: pivot, i,
        add     $sp, $sp, -16
        sw      $a0, 0($sp)             # luu cac doi so vao stack
        sw      $a1, 4($sp)
        sw      $a2, 8($sp)
        sw      $ra, 12($sp)            # luu dia chi tra ve

        addu    $s0, $a1,$0             # s0 = low = left
        addu    $s1, $a2,$0             # s1 = high

        addu    $t5, $s1,$0             # t5 = high: save pivot index

        sll     $t0, $s1, 2             # t0 = 4 * high
        add     $t0, $a0, $t0           # t0 = array + 4*high
        lw      $s3, 0($t0)             # s3 = arr[high]=== (pivot)

        subi    $s1, $s1, 1             # s1 = high - 1 = right
#---------------------------------

BigLoop:
        # while(left <= right && arr[left] < pivot) left++;
        LeftLoop:
                sll     $t1, $s0, 2             # t1 = left * 4
                add     $t1, $a0, $t1   # t1 = array + 4*left
                lw      $t2, 0($t1)             # t2 = &array[left]

                # kiem tra left <= right
                blt     $s1, $s0, LeftLoopDone
                #kiem tra arr[left] < pivot
                bge     $t2, $s3, LeftLoopDone

                #left++
                addi    $s0, $s0, 1

                j       LeftLoop
```

```
LeftLoopDone:

# while(right >= left && arr[right] > pivot) right--;
RightLoop:
        sll     $t3, $s1, 2             # t3 = right * 4
        add     $t3, $a0, $t3   # t3 = array + 4*right
        lw      $t4, 0($t3)             # t4 = &array[right]

        # kiem tra right >= left
        bgt     $s0, $s1, RightLoopDone
        #kiem tra arr[right] > pivot
        ble     $t4, $s3, RightLoopDone

        #right--
        subi    $s1, $s1, 1

        j       RightLoop
```

```
RightLoopDone:

#if (left >= right) break;
bge     $s0, $s1, BreakLoop

addu    $a1, $s0,$0
addu    $a2, $s1,$0
jal     swap
jal     PRINT1
                                                        # swap(a[left], a[right])
continue1:      addi    $s0, $s0, 1             # left++
                subi    $s1, $s1, 1             # right--
                j       BigLoop
#-----------------------------------------------
        BreakLoop:
                addu    $a1, $s0,$0
                addu    $a2, $t5,$0
                jal     swap
                jal     PRINT1          # swap(a[left], pivot)
                add     $v0, $s0, 0             # Return left

                lw      $a0, 0($sp)             # tai doi so duoc luu trong stack ve thanh ghi
                lw      $a1, 4($sp)
                lw      $a2, 8($sp)
lw      $ra, 12($sp)
addi    $sp, $sp, 16    # stack trong
jr      $ra                     # tro lai ham goi
```

3.4. Hàm Swap

Giải thuật	void swap(array[left] , array[right]) { int temp = array[left] array[left] = array[right] array[right]= temp

	}
Assembly	

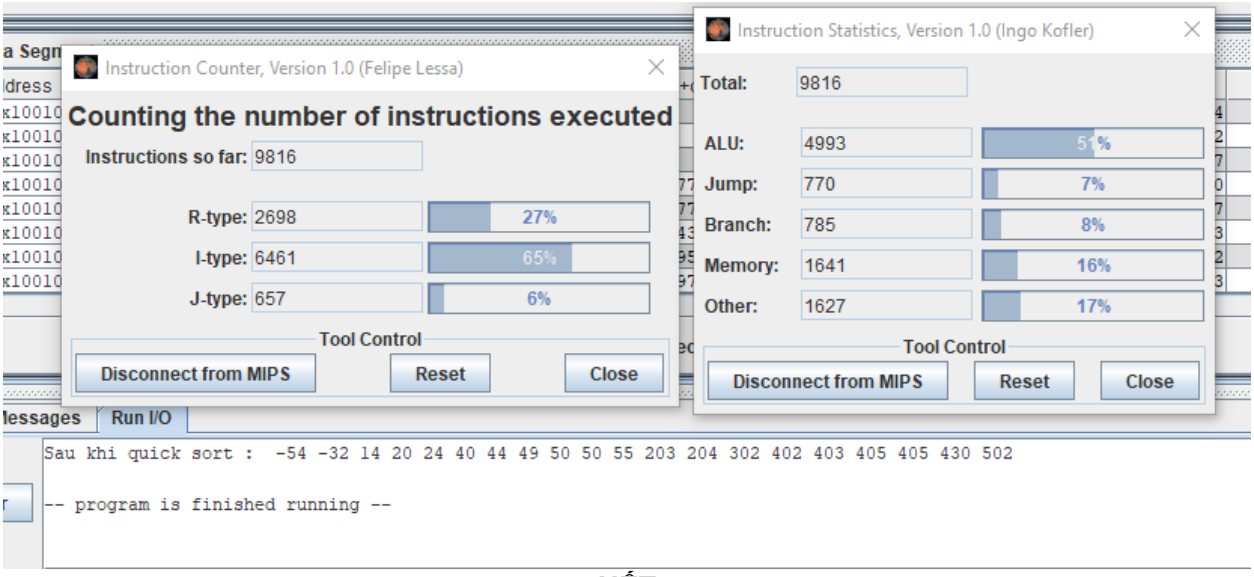
```
swap:
        # khoi tao cap phat vung nho cho stack fram
        add     $sp, $sp, -20
        # thong so truyen vao thu 1
        sw      $a0, 0($sp)             # luu doi so vao stack
        # thong so truyen vao thu 2
        sw      $a1, 4($sp)
        # thong so truyen vao thu 3
        sw      $a2, 8($sp)
        sw      $s0, 12($sp)
        sw      $s1, 16($sp)
        # con tro phan tu left
        sll     $t0, $a1, 2             # t0 = x * 4
        add     $t0, $a0, $t0   # t0 = array + 4*x
        lw      $s0, 0($t0)             # s0 = &array[x]
        # con tro phan tu right
        sll     $t1, $a2, 2             # t1 = y * 4
        add     $t1, $a0, $t1   # t1 = array + 4*y
        lw      $s1, 0($t1)     # s1 = &array[y]
        # Swap
        sw      $s0, 0($t1)
        sw      $s1, 0($t0)

lw      $a0, 0($sp)             # tai doi so tu stack ve thanh ghi
lw      $a1, 4($sp)
lw      $a2, 8($sp)
lw      $s0, 12($sp)
lw      $s1, 16($sp)

# tra lai vung nho da cap phat o tren
addi    $sp, $sp, 20
jr       $ra                    # nhay ve ham goi, tra lai gia tri
-----------------------------------------------------------------------
```

4.Test case

Test case 1	**10,-4,29,3,-1,-9,-12,-5,14,44,23,49,68,-70,55,27,99,2,7,-100**
Kết quả	Sau khi quick sort : -100 -70 -12 -9 -5 -4 -1 2 3 7 10 14 23 27 29 44 49 55 68 99 -- program is finished running --
Test case 2	**-30,1,5,9,20,4,25,50,400,50,60,40,30,30,20,10,400,500,-200,100, 92**
Kết quả	Sau khi quick sort : -200 -30 1 4 5 9 10 20 20 25 30 30 40 50 50 60 92 100 400 400 500
Test case 3	**11,15,10,-2,-3,-10,-20,40,13,30,20,99,56,33,49,42,3,4,1,5**
Kết quả	Sau khi quick sort : -20 -10 -3 -2 1 3 4 5 10 11 13 15 20 30 33 40 42 49 56 99
Test case 4	**200,-94,-100,-101,3,14,50,60,20,30,33,54,52,6,-10,-11, -5,4,9,20**
Kết quả	Sau khi quick sort : -101 -100 -94 -11 -10 -5 3 4 6 9 14 20 20 30 33 50 52 54 60 200
Test case 5	**2,30,1000,4,3,-1,45,6,500,5040,50432,403,430,-1,-2,-1000,23,3,4,5**

Kết quả	Sau khi quick sort : -1000 -2 -1 -1 2 3 3 4 4 5 6 23 30 45 403 430 500 1000 5040 50432
Test case 6	**-432,-1000,-230,-3210,-320,-23,-53,-4,32,-5,-40,5,-40,-20,-2,-2,-4,-2320,-43**
Kết quả	Sau khi quick sort : -3210 -2320 -1000 -432 -320 -230 -53 -43 -40 -40 -23 -20 -5 -4 -4 -2 -2 5 32
Test case 7	**100,55,66,99,43,680,65,33,-1,-3,4,-5,-11,-443,-553,445,43,9,8,10**
Kết quả	Sau khi quick sort : -553 -443 -11 -5 -3 -1 4 8 9 10 33 43 43 55 65 66 99 100 445 680
Test case 8	**100,203,2,53,68,80,60,35,69,43,29,50,60,62,50,10,3,5,6,7**
Kết quả	Sau khi quick sort : 2 3 5 6 7 10 29 35 43 50 50 53 60 60 62 68 69 80 100 203
Test case 9	**2031,42,3021,4103,2304,1002,3060,7054,5690,5960,3045,5903,5903,3200 ,4302,4306,4030,6030,4032,5005**
Kết quả	Sau khi quick sort : 42 1002 2031 2304 3021 3045 3060 3200 4030 4032 4103 4302 4306 5005 5690 5903 5903 5960 6030 7054
Test case 10	**-32,-54,14,50,204,49,24,40,403,20,203,402,55,44,50,502,405,302,405,430**
Kết quả	Sau khi quick sort : -54 -32 14 20 24 40 44 49 50 50 55 203 204 302 402 403 405 405 430 502

5. Thống kê loại lệnh



-=HẾT=-