ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BÀI TẬP LỚN
KIẾN TRÚC MÁY TÍNH - CO2008

GVHD: KS Nguyễn Xuân Minh

Lớp L04 Nhóm 08

Tp. Hồ Chí Minh, Tháng 11/2021

Mục lục

1 Thành viên Nhóm L04-08

1. Vũ Trần Hoàng
   - MSSV: 2013245
   - Email: hoang.vu141102@hcmut.edu.vn
2. Hoàng Văn Khải
   - MSSV: 2013473
   - Email: khai.hoang271611@hcmut.edu.vn

2 Đề bài.

Merge sort số nguyên .
Viết chương trình sắp xếp dãy số nguyên có 15 phần tử nhập từ bàn phím dùng giải thuật Merge sort. Yêu cầu xuất dãy ra màn hình mỗi bước. Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa INT15.BIN (15 phần tử x 4 bytes = 60 bytes).

3 Giải pháp hiện thực.

- Với hàm mergeSort, ta thực hiện việc chia mảng thành 2 nửa bằng cách sử dụng địa chỉ hoặc index, thay vì thật sự chia mảng ban đầu thành 2 mảng con, ta sử dụng phần tử đầu đến phần tử ở giữa đại diện cho mảng bên trái, và từ phần tử giữa đến phần tử cuối đại diện cho mảng bên phải.
- Về hàm merge, giải thuật gốc yêu cầu ta cấp phát một mảng có độ dài bằng tổng độ dài 2 mảng con cần merge, với mips việc này khá khó khăn, vì một khi ta cấp mảng ở vùng nhớ heap (nâng break pointer lên), ta không thể giải phóng vùng nhớ đó, vì vậy ta sử dụng một cách làm thay thế mà không cần cấp phát mảng mới, cách làm như sau:
Giả sử ta có mảng arr, left là index phần tử đầu tiên của mảng bên trái, mid là index của phần tử giữa, và right là index phần từ cuối cùng của mảng phải. Ta thực hiện so sánh arr[left] và arr[mid], nếu arr[left] nhỏ hơn, giữ nguyên mảng vì phần tử này đã nằm đúng vị trí, xét phần tử tiếp theo của mảng trái (left+1), còn nếu arr[mid] nhỏ hơn, ta tiến hành dịch chuyển arr[mid] đến vị trí left, tức đẩy phần tử nhỏ hơn lên trên, tiếp tục như vậy đến hết mảng.

4 Giải thuật.

Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa, tiếp tục lại lại việc chia này cho 2 nửa vừa chia cho đến khi mỗi nửa thì còn một phần tử, cuối cùng gộp dần các mảng vừa chia lại. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất, hàm sẽ gộp hai nửa mảng là arr[l...m] và arr[m+1...r] thành một mảng duy nhất đã sắp xếp.
a) Hàm mergeSort, đây là hàm ta sẽ gọi đệ quy để chia mảng thành từng nửa.

```
1 Algorithm recursiveMergeSort(ref arr
    <array>, val hi <int> , val lo <int>)

2 if hi > lo then
3 |   mid = lo + (hi - lo) / 2
4 |   recursiveMergeSort(arr, lo, mid)
5 |   recursiveMergeSort(arr, mid + 1, hi)
6 |   merge(arr, lo, mid, hi)
7 end
8 End recursiveMergeSort
```

b) Hàm merge, đây là hàm sẽ thực hiện việc gộp 2 mảng con lại.

```
MERGE(A, p, q, r)
 1  n1 ← q − p + 1
 2  n2 ← r − q
 3  create arrays L[1 . . n1 + 1] and R[1 . . n2 + 1]
 4  for i ← 1 to n1
 5       do L[i] ← A[p + i − 1]
 6  for j ← 1 to n2
 7       do R[j] ← A[q + j]
 8  L[n1 + 1] ← ∞
 9  R[n2 + 1] ← ∞
10  i ← 1
11  j ← 1
12  for k ← p to r
13       do if L[i] ≤ R[j]
14            then A[k] ← L[i]
15                 i ← i + 1
16            else A[k] ← R[j]
17                 j ← j + 1
```

5 Các câu lệnh đã sử dụng.

```
Addi   $v0, $zero, 30
syscall
add   $t4, $zero, $a0
la   $a0, file
la   $a1, arr
lw   $a2, size
jal   readFile
li   $v0, 4
la   $a0, Nhac_1
la   $a0, arr
lw   $a1, size
jal   printArray
la   $a0, Nhac_2
lw   $t0, size
sll   $t0, $t0, 2
```

```
sw   $a0, 0($sp)
sw   $a1, 4($sp)
sw   $a2, 8($sp)
li   $v0, 13
li   $a1, 0
move   $s1, $v0
lw   $t0, 8($sp)
li   $v0, 14
move   $a0, $s1
lw   $a1, 4($sp)
move   $a2, $t0
li   $v0, 16
lw   $a0, 0($sp)
lw   $a2, 8($sp)
addi   $sp, $sp, 12
```

```
add   $a1, $a0, $t0
jal   mergeSort
la   $a0, Nhac_3
lw   $a1,size
la   $a0,Nhac_4
add   $t5, $zero, $a0
sub   $t4, $t5, $t4
li   $v0, 1
add   $a0,$zero,$t4
li   $v0, 10
addi   $sp, $sp, -12
```

```
jr   $ra
addi   $sp, $sp, -16
sw   $ra, 0($sp)
sw   $a0, 4($sp)
sw   $a1, 8($sp)
sub   $t0, $a1, $a0
ble   $t0, 4, endMergeSort
srl   $t0, $t0, 3
add   $t0, $a0, $t0
sw   $t0, 12($sp)
lw   $a0, 4($sp)
```

```
lw   $a1, 12($sp)
lw   $a0, 12($sp)
lw   $a1, 8($sp)
lw  $ra, 0($sp)
addi   $sp, $sp, 16
sw   $a2, 12($sp)
move   $s0, $a0
move   $s1, $a1
lw   $t0, 0($s0)
lw   $t1, 0($s1)
ble   $t0,$t1,nextElement
move   $a1, $s0
jal   shiftElement
addi   $s1, $s1, 4
```

```
sw   $t2, 0($a0)
addi   $a0, $a0, -4
j   loopshiftElement
addi   $sp, $sp, 8
addi   $sp,$sp,-12
sw   $a0,8($sp)
sw   $a1,4($sp)
sw   $ra,0($sp)
li   $t0,0
slt   $t1,$t0,$a1
beq   $t1,$zero,end1
move   $t6,$a0
sll   $t1,$t0,2
add   $t1,$t1,$a0
```

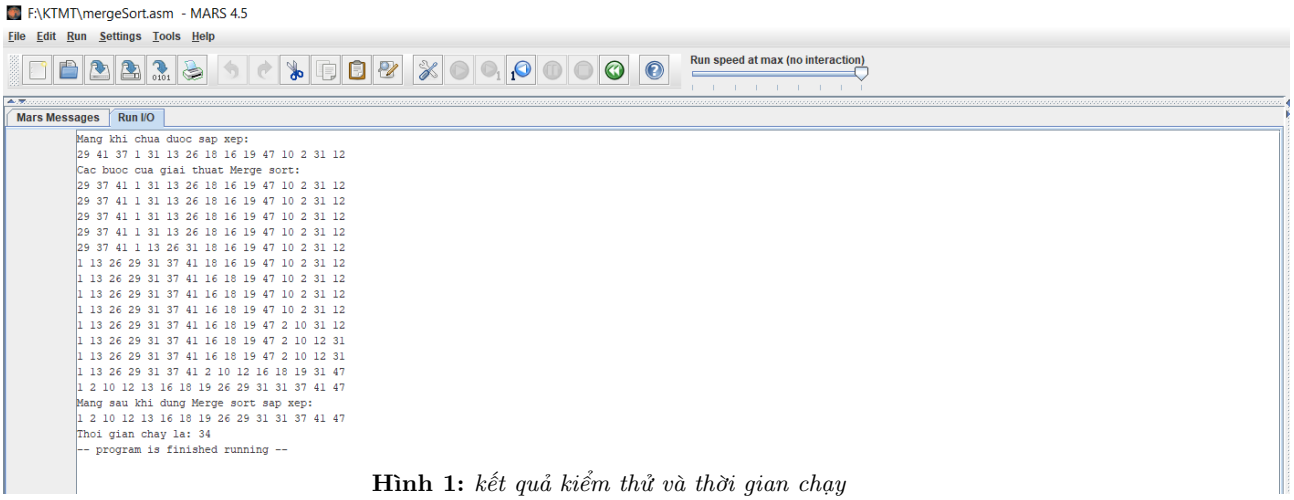
```
addi   $s0, $s0, 4
lw   $a2, 12($sp)
bge   $s0, $s1, endMerge
bge   $s1, $a2, endMerge
j   mergeloop
addi   $sp, $sp, -8
beq   $a0, $a1, endShiftElement
addi   $t0, $a0, -4
lw   $t1, 0($a0)
lw   $t2, 0($t0)
sw   $t1, 0($t0)
```

```
lw   $a0,0($t1)
la   $a0, spacechac
move   $a0, $t6
addi   $t0,$t0,1
j   loop1
lw   $a0,8($sp)
lw   $a1,4($sp)
lw   $ra,0($sp)
addi   $sp,$sp,12
la   $a0, endline
```

6 Kết quả kiểm thử và thời gian chạy.

- Thời gian chạy là: 34
- Mảng khi chưa được sắp xếp:
- 29 41 37 1 31 13 26 18 16 19 47 10 2 31 12
- Các bước của giải thuật Merge Sort:
- 29 37 41 1 31 13 26 18 16 19 47 10 2 31 12
- 29 37 41 1 31 13 26 18 16 19 47 10 2 31 12
- 29 37 41 1 31 13 26 18 16 19 47 10 2 31 12
- 29 37 41 1 31 13 26 18 16 19 47 10 2 31 12
- 29 37 41 1 13 26 31 18 16 19 47 10 2 31 12
- 1 13 26 29 31 37 41 18 16 19 47 10 2 31 12
- 1 13 26 29 31 37 41 16 18 19 47 10 2 31 12
- 1 13 26 29 31 37 41 16 18 19 47 10 2 31 12
- 1 13 26 29 31 37 41 16 18 19 47 10 2 31 12
- 1 13 26 29 31 37 41 16 18 19 47 2 10 31 12
- 1 13 26 29 31 37 41 16 18 19 47 2 10 12 31
- 1 13 26 29 31 37 41 16 18 19 47 2 10 12 31
- 1 13 26 29 31 37 41 2 10 12 16 18 19 31 47
- 1 2 10 12 13 16 18 19 26 29 31 31 37 41 47

- Mảng sau khi dùng Merge Sort sắp xếp:
- 1 2 10 12 13 16 18 19 26 29 31 31 37 41 47



Hình 1: *kết quả kiểm thử và thời gian chạy*