PROJECT 2
MIPS Architecture and Assembly Language

Quy định chung

- Đồ án được làm theo nhóm: ***mỗi nhóm 2-3 sinh viên.***

- Bài làm giống nhau giữa các nhóm, tất cả các nhóm liên quan đều bị điểm **0 cả thực hành**

- Ngôn ngữ sử dụng: **MIPS 32 BIT** ; IDE: **MARS**

Yêu cầu

Áp dụng kiến thức về thuật toán Sort (sắp xếp tăng dần). Hãy xây dựng chương trình hợp ngữ MIPS 32 bits cho hai thuật toán Insertion sort và Selection sort.

Lý thuyết:

Thuật toán Insertion sort:

Sắp xếp chèn là một giải thuật sắp xếp dựa trên so sánh in-place. Ở đây, một danh sách con luôn luôn được duy trì dưới dạng đã qua sắp xếp. Sắp xếp chèn là chèn thêm một phần tử vào danh sách con đã qua sắp xếp. Phần tử được chèn vào vị trí thích hợp sao cho vẫn đảm bảo rằng danh sách con đó vẫn sắp theo thứ tự.

Với cấu trúc dữ liệu mảng, chúng ta tưởng tượng là: mảng gồm hai phần: một danh sách con đã được sắp xếp và phần khác là các phần tử không có thứ tự. Giải thuật sắp xếp chèn sẽ thực hiện việc tìm kiếm liên tiếp qua mảng đó, và các phần tử không có thứ tự sẽ được di chuyển và được chèn vào vị trí thích hợp trong danh sách con (của cùng mảng đó).

Giải thuật này không thích hợp sử dụng với các tập dữ liệu lớn khi độ phức tạp trường hợp xấu nhất và trường hợp trung bình là $O(n^2)$ với n là số phần tử.

Nguồn tham khảo:

https://en.wikipedia.org/wiki/Insertion_sort

http://vietjack.com/cau-truc-du-lieu-va-giai-thuat/giai-thuat-sap-xep-chen.jsp

Giải thuật:

```
Bước 1: Kiểm tra nếu phần tử đầu tiên đã được sắp xếp. trả về 1
Bước 2: Lấy phần tử kế tiếp
Bước 3: So sánh với tất cả phần tử trong danh sách con đã qua sắp xếp
Bước 4: Dịch chuyển tất cả phần tử trong danh sách con mà lớn hơn giá trị để
được sắp xếp
Bước 5: Chèn giá trị đó
Bước 6: Lặp lại cho tới khi danh sách được sắp xếp
```

Giải thuật mẫu

Không sử dụng đệ quy:

```
for i = 1 to length(A)
    x = A[i]
    j = i - 1
    while j >= 0 and A[j] > x
        A[j+1] ← A[j]
        j ← j - 1
    end while
    A[j+1] = x[3]
 end for
```

Sử dụng đệ quy:

```
function insertionSortR(array A, int n)
     if n>1
        insertionSortR(A,n-1)
        x = A[n]
        j = n-1
        while j >= 0 and A[j] > x
            A[j+1] ← A[j]
            j ← j-1
        end while
        A[j+1] = x
     end if
 end function
```

Thuật toán Selection Sort:

Giải thuật sắp xếp chọn (Selection Sort) là một giải thuật đơn giản. Giải thuật sắp xếp này là một giải thuật dựa trên việc so sánh **in-place**, trong đó danh sách được chia thành hai phần, phần được sắp xếp (sorted list) ở bên trái và phần chưa được sắp xếp (unsorted list) ở bên phải. Ban đầu, phần được sắp xếp là trống và phần chưa được sắp xếp là toàn bộ danh sách ban đầu.

Phần tử nhỏ nhất được lựa chọn từ mảng chưa được sắp xếp và được tráo đổi với phần bên trái nhất và phần tử đó trở thành phần tử của mảng được sắp xếp. Tiến trình này tiếp tục cho tới khi toàn bộ từng phần tử trong mảng chưa được sắp xếp đều được di chuyển sang mảng đã được sắp xếp.

Giải thuật này không phù hợp với tập dữ liệu lớn khi mà độ phức tạp trường hợp xấu nhất và trường hợp trung bình là $O(n^2)$ với n là số phần tử.

Nguồn tham khảo:

https://en.wikipedia.org/wiki/Selection_sort

http://vietjack.com/cau-truc-du-lieu-va-giai-thuat/giai-thuat-sap-xep-chon.jsp

Giải thuật

```
Bước 1: Thiết lập MIN về vị trí 0
Bước 2: Tìm kiếm phần tử nhỏ nhất trong danh sách
Bước 3: Tráo đổi với giá trị tại vị trí MIN
Bước 4: Tăng MIN để trỏ tới phần tử tiếp theo
Bước 5: Lặp lại cho tới khi toàn bộ danh sách đã được sắp xếp
```

Giải thuật mẫu

```
/* a[0] to a[n-1] is the array to sort */
int i,j;
int n;

/* advance the position through the entire array */
/*   (could do j < n-1 because single element is also min element) */
for (j = 0; j < n-1; j++) {
    /* find the min element in the unsorted a[j .. n-1] */

    /* assume the min is the first element */
    int iMin = j;
    /* test against elements after j to find the smallest */
    for (i = j+1; i < n; i++) {
        /* if this element is less, then it is the new minimum */
        if (a[i] < a[iMin]) {
            /* found new minimum; remember its index */
            iMin = i;
        }
    }

    if(iMin != j) {
        swap(a[j], a[iMin]);
    }
}
```

Yêu cầu chương trình

1. Gồm có 2 chương trình hợp ngữ MIPS: InsertionSort.asm và SelectionSort.asm
2. Dữ liệu input được đọc từ file, dữ liệu output là chuỗi được sort.

a. Cấu trúc file input.txt:
    i. n: số lượng phần tử (max=1000 phần tử)
    ii. array_int: dãy số nguyên dương (mỗi số nguyên cách nhau bởi khoảng trắng)
    ví dụ:

    10
    34 21 43 29 21 453 1 12 0 34

b. Cấu trúc file output.txt
    i. array_int: dãy số nguyên đã được sắp xếp tăng dần

3. Cách chạy chương trình:
    a. File chương trình *.asm, input.txt, output.txt nằm cùng thư mục.
    b. Run *.asm → Thực hiện sort → Kết thúc
    c. Chương trình khi chạy sẽ không nhập bất kỳ dữ liệu nào, mà chỉ lấy dữ liệu từ file input.txt (ở cùng thư mục) và lưu kết quả vào file output.txt (cùng thư mục).

Hình thức nộp và chấm bài:

- File nộp: *MSSV1_MSSV2_MSSV3.zip* hoặc *rar gồm*
    - Hai file: InsertionSort.asm và SelectionSort.asm
    - File báo cáo MSSV1_MSSV2_MSSV3.docx trình bày:
      - Thông tin thành viên nhóm, bảng phân công công việc.
      - Nêu rõ môi trường lập trình.
      - Nêu rõ ý tưởng để thiết kế và thực hiện đồ án. Cho biết phạm vi biểu diễn của các kiểu dữ liệu đã thiết kế.
      - Chạy kiểm tra và chụp hình, chú thích từng chức năng của chương trình minh họa.
      - Nêu rõ chức năng làm được, và chưa được.
      - Đánh giá mức độ hoàn thành theo tỉ lệ phần trăm (%) của toàn bộ đồ án.
      - Các nguồn tài liệu tham khảo.
      - Lưu ý: không chép source code vào báo cáo.

- Hạn chót: *23/04/2017 vào lúc 23h55* tại moodle môn học