ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA TPHCM
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH (CO2008)

BÀI TẬP LỚN

BÀI TẬP NHÓM (ĐỀ 5 - SELECTION SORT)

Giáo viên hướng dẫn: Nguyễn Xuân Minh

Sinh viên báo cáo: Đỗ Duy Cương - 2210413 (L15)
Nguyễn Lê Hoàng Phúc - 2212629 (L14)

Thành phố Hồ Chí Minh, Ngày 26 tháng 12 năm 2023

Contents

1 Đề bài

Viết chương trình sắp xếp dãy số nguyên có 10 phần tử dùng giải thuật Selection sort. Yêu cầu xuất ra màn hình các bước có thay đổi thứ tự trong dãy. Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa INT10.BIN (10 phần tử x 4 bytes = 40 bytes).

2 Yêu cầu

2.1 Code

Chương trình cần được viết và chạy trên MARS MIPS 4.5 với các yêu cầu sau:

- Code style phải rõ ràng, có chú thích.
- Phải có gọi hàm. Truyền tham số và trả kết quả khi gọi hàm theo quy ước của thanh ghi ($A1 chứa tham số, $V1 hoặc $f1 chứa giá trị trả về).
- In thông tin ra màn hình để kiểm tra.

2.2 Nội dung báo cáo

Nội dung báo cáo bao gồm:

- Trình bày giải pháp hiện thực.
- Giải thuật (nếu có).
- Thống kê số lệnh, loại lệnh (instruction type) sử dụng trong chương trình.
- Tính thời gian chạy của chương trình (CR=1GHz).
- Kết quả kiểm thử.

2.3 Nộp báo cáo

Báo cáo nên được nộp dưới dạng 3 files:

- File báo cáo (không bao gồm source code) định dạng .PDF (Bc_nhom###.pdf).
- File mã nguồn (Mn###.asm).
- File dữ liệu đầu vào (xxx.BIN nếu có).

3 Giải pháp hiện thực

3.1 Quá trình hoàn thiện báo cáo:

Ngày	Nhiệm vụ
01/12/2023	Tìm hiểu yêu cầu.
02/12/2023	Xác định và phân công nhiệm vụ.
03/12/2023	Tiến hành hoàn thành nhiệm vụ, trao đổi thông tin nội bộ.
16/12/2023	Tổng hợp nội dung và hoàn thiện báo cáo.

3.2 Phân tích đề bài

- Mục tiêu tiên quyết: nắm vững ý tưởng của thuật toán sắp xếp lựa chọn (selection sort) và hiện thực thuật toán đó bằng ngôn ngữ lập trình hợp ngữ (assembly language), đồng thời xuất ra màn hình các bước có sự thay đổi thứ tự trong quá trình sắp xếp.
- Các ràng buộc tiên quyết: dữ liệu đầu vào được đọc từ file lưu trữ dạng nhị phân trên đĩa INT10.BIN (10 phần tử x 4 bytes = 40 bytes).
- Các ràng buộc bổ sung: Về nội dung, mã nguồn phải mô phỏng đúng tinh thần của thuật toán selection sort, không tồn tại lỗi và không phát sinh lỗi trong quá trình thực thi chương trình. Về hình thức, phong cách lập trình – code style phải trong sáng, rõ ràng, không gây ra sự nhập nhằng khi đọc và điều chỉnh mã nguồn, đồng thời đảm bảo có chú thích phù hợp cho các dòng code không rõ ràng về mặt ngữ nghĩa.

Từ đó, ta xác định nhiệm vụ cần phải hoàn thành trong bài tập lớn này bao gồm các nội dung sau: xác định giải pháp hiện thực, thiết kế và tối ưu hóa mã nguồn, soạn thảo văn bản báo cáo. Ba nhiệm vụ này lần lượt được giải quyết ngay sau đây.

3.3 Xác định giải pháp hiện thực

- Thuật toán chính: selection sort (được thực thi bằng việc chọn ra phần tử bé nhất và đặt về trước trong mỗi lần lặp).
- Độ phức tạp: về mặt thời gian cho cả 3 trường hợp: tốt nhất, trung bình, tệ nhất và về mặt không gian.

- Các hàm chính trong chương trình: Dựa trên ý tưởng của việc hiện thực selection sort bằng ngôn ngữ cấp cao như C, C++, có tổng cộng 03 hàm chính và 01 quá trình nhập/xuất: 01 hàm SelectionSort, 01 hàm Swap, 01 hàm Print, 01 quá trình nhập dữ liệu từ file INT10.BIN. Trong đó quá trình nhập dữ liệu diễn ra đầu tiên để khởi tạo các giá trị (nội dung) cho mảng ban đầu, hàm selectionSort sẽ gọi hàm Swap ở mỗi vòng lặp để hoán đổi giá trị của hai phần tử xác định và gọi hàm Print ở mỗi vòng lặp có diễn ra sự thay đổi dữ liệu để xuất ra màn hình sự thay đổi đó.

Quá trình thiết kế được hỗ trợ bởi các macro thích hợp, ở bài tập lớn này ta cần: macro PrintInteger, macro PrintString lần lượt dùng trong việc xuất ra màn hình số nguyên và chuỗi tương ứng (về mặt bản chất, đây được xem là một phương pháp tối ưu hóa mã nguồn).

3.4 Thiết kế và tối ưu hóa mã nguồn

Thiết kế lần lượt 03 hàm và 01 quá trình như đã phân tích ở mục 3, ở mỗi hàm cần truyền tham số và lưu kết quả trả về đúng theo yêu cầu của bài tập lớn: `$ai` chứa tham số, `$vi` hoặc `$fi` chứa giá trị trả về (riêng với thuật toán selection sort, các registers số thực thuộc coprocessor1 `$fi` không cần thiết sử dụng). Ở mỗi hàm, cũng cần preserve (bảo tồn) các giá trị thanh ghi không tạm thời, như `$si`, `$ra` (với hàm không phải hàm lá), ... và restore trả về giá trị nguyên thủy – trạng thái ban đầu của các thanh ghi được preserved trước khi kết thúc hàm.

Tối ưu hóa mã nguồn được thực hiện sau khi hình hài cơ bản của mã nguồn được hình thành. Quá trình đó bao gồm việc đọc lại mã nguồn, truy tìm lệnh ảo (lệnh không đóng góp giá trị cho mã nguồn), rút gọn mã nguồn, thay đổi phương án thực thi một đoạn code nhỏ trong mã nguồn, tối ưu hóa lượng thanh ghi sử dụng, tận dụng tối đa tài nguyên về bộ 32 thanh ghi, ...

Một mã nguồn "sạch" và "sáng" là mã nguồn đảm bảo hai tiêu chí song song, thứ nhất, được thiết kế dựa trên một cơ sở lý thuyết có logic chặt chẽ, cụ thể, dễ tiếp cận – đó là tính sáng, thứ hai, mang hình hài với phong cách chỉn chu, bố cục rõ ràng, dễ nắm bắt – đó là tính sạch. Để làm được điều này, quá trình thiết kế ngay từ đầu phải dựa trên lịch trình chi tiết, cùng với đó là quá trình tối ưu cũng phải được truy xét kỹ càng.

3.5 Soạn thảo văn bản báo cáo:

Văn bản báo cáo được soạn thảo trực tiếp bằng latex và trình bày tất cả những khía cạnh có thể đề cập liên quan đến yêu cầu đề bài.

4 Giới thiệu giải thuật Selection Sort

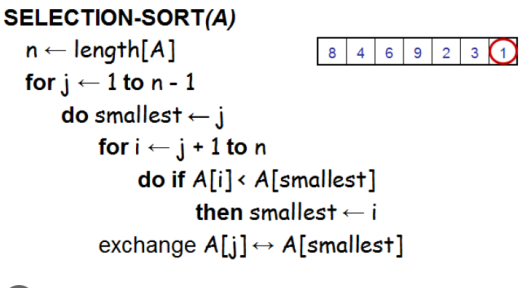
Thuật toán Selection Sort là một thuật toán sắp xếp cơ bản được sử dụng trong lĩnh vực khoa học máy tính. Đây là một phương pháp đơn giản để sắp xếp một danh sách hoặc mảng. Trong báo cáo này, chúng tôi sẽ trình bày về cơ chế hoạt động, mã giả và đánh giá hiệu suất của thuật toán Selection Sort.

4.1 Mô tả thuật toán

Thuật toán Selection Sort làm việc bằng cách lặp qua mảng và chọn ra phần tử nhỏ nhất (hoặc lớn nhất, tùy thuộc vào yêu cầu sắp xếp) từ phần chưa được sắp xếp và hoán đổi nó với phần tử đầu tiên của phần chưa được sắp xếp.

4.2 Mã giả

Presented below is the pseudocode for the Selection Sort algorithm:



Hình 1: Pseudocode of the Selection Sort algorithm.

4.3 Hiện thực giải thuật bằng ngôn ngữ cấp cao

```
void printArr (int *arr, const int N)
{
    for (int i = 0; i < N; i++) {
        cout << *(arr + i) << " ";
    }
    cout << endl;
}
```

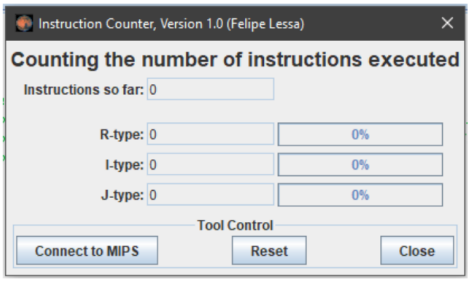
```
void selectionSort (int *arr, const int N)
{
    int count = 1;
    for (int i = 0; i < N - 1; i++) {
        int min = i;
        for (int j = i + 1; j < N; j++) {
            if (*(arr + min) > *(arr + j)) {
                min = j;
            }
        }
        if (min != i) {
            cout << "step " << count++ << " : ";
            swap(*(arr + min), *(arr + i));
            printArr(arr, N);
        }
    }
}
```

4.4 Kết luận

Thuật toán Selection Sort có độ phức tạp thời gian là $O(n^2)$, khiến nó không hiệu quả cho những danh sách lớn. Tuy nhiên, nó rất dễ hiểu và thực hiện, làm cho nó trở thành một lựa chọn tốt cho thuật toán sắp xếp cơ bản.

5 Thống kê số lệnh sử dụng, thời gian chạy

Sử dụng công cụ Instruction Counter để đếm số lệnh trong chương trình:



Hình 2: Instruction Counter Tool.

Sử dụng công thức sau để tính thời gian chạy:

$$\text{CPU time} = \frac{\text{CPU Clock cycles}}{\text{Clock rate}} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}}$$

Trong đó:

- CPU Time: là thời gian xử lý của chương trình (không tính thời gian giao tiếp I/O, thời gian chờ...).
- CPU Clock Cycles: Tổng số chu kỳ thực thi.
- Instruction Counts: là tổng số lệnh thực thi của chương trình.
- CPI (Cycle Per Instruction): là số chu kỳ thực thi trên một lệnh.
- Clock Rate: là số chu kỳ thực thi trên một giây hay còn gọi là tần số, ví dụ: 3.4×10^9 giao động.

Tuỳ vào cách bạn nhập mảng và nội dung mảng số mà chương trình có tổng số lệnh, loại lệnh va thời gian chạy khác nhau.
Dưới đây là một số ví dụ cho thấy thời gian chạy khác nhau:

5.1 Ví dụ 1

Mảng nhập vào:

$$\text{Array} = [100, 200, 300, 213, 212, 121, 123, 456, 822, 122]$$

Tổng số lệnh: 2866

Trong đó:

- R-type: 802 lệnh chiếm 27%
- I-type: 1860 lệnh chiếm 64%
- J-type: 204 lệnh chiếm 7%

Tính toán thời gian chạy::

$$\begin{aligned}\text{CPU time} &= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}} \\ &= \frac{2866 \times 1}{1 \times 10^9} \\ &= 2.866 \times 10^{-6} \\ &= 2.866\mu s\end{aligned}$$

5.2 Ví dụ 2

Mảng nhập vào:

$$\text{Array} = [151, 12, 16, 14, 18, 17, 111, 113, 112, 457]$$

Tổng số lệnh: 2531

Trong đó:

- R-type: 708 lệnh chiếm 27%
- I-type: 1645 lệnh chiếm 64%
- J-type: 178 lệnh chiếm 7%

Tính toán thời gian chạy::

$$\begin{aligned}\text{CPU time} &= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}} \\ &= \frac{2531 \times 1}{1 \times 10^9} \\ &= 2.531 \times 10^{-6} \\ &= 2.531\mu s\end{aligned}$$

5.3 Ví dụ 3

Mảng nhập vào:

$$\text{Array} = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$$

Tổng số lệnh: 2126

Trong đó:

- R-type: 617 lệnh chiếm 29%
- I-type: 1364 lệnh chiếm 64%
- J-type: 145 lệnh chiếm 6%

Tính toán thời gian chạy::

$$\begin{aligned}\text{CPU time} &= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}} \\ &= \frac{2126 \times 1}{1 \times 10^9} \\ &= 2.126 \times 10^{-6} \\ &= 2.126\mu s\end{aligned}$$

6 Kết quả kiểm thử

Data	Result
10 9 8 7 6 5 4 3 2 1	10, 9, 8, 7, 6, 5, 4, 3, 2, 1, Step 1: 1, 9, 8, 7, 6, 5, 4, 3, 2, 10 Step 2: 1, 2, 8, 7, 6, 5, 4, 3, 9, 10 Step 3: 1, 2, 3, 7, 6, 5, 4, 8, 9, 10 Step 4: 1, 2, 3, 4, 6, 5, 7, 8, 9, 10 Step 5: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Sorted Array: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 -- program is finished running --
102 91 884 714 697 52 484 31 2 1	102, 91, 884, 714, 697, 52, 484, 31, 2, 1, Step 1: 1, 91, 884, 714, 697, 52, 484, 31, 2, 102 Step 2: 1, 2, 884, 714, 697, 52, 484, 31, 91, 102 Step 3: 1, 2, 31, 714, 697, 52, 484, 884, 91, 102 Step 4: 1, 2, 31, 52, 697, 714, 484, 884, 91, 102 Step 5: 1, 2, 31, 52, 91, 714, 484, 884, 697, 102 Step 6: 1, 2, 31, 52, 91, 102, 484, 884, 697, 714 Step 7: 1, 2, 31, 52, 91, 102, 484, 697, 884, 714 Step 8: 1, 2, 31, 52, 91, 102, 484, 697, 714, 884 Sorted Array: 1, 2, 31, 52, 91, 102, 484, 697, 714, 884 -- program is finished running --
10 92 800 7 6 50 454 31 22 1478	10, 92, 800, 7, 6, 50, 454, 31, 22, 1478, Step 1: 6, 92, 800, 7, 10, 50, 454, 31, 22, 1478 Step 2: 6, 7, 800, 92, 10, 50, 454, 31, 22, 1478 Step 3: 6, 7, 10, 92, 800, 50, 454, 31, 22, 1478 Step 4: 6, 7, 10, 22, 800, 50, 454, 31, 92, 1478 Step 5: 6, 7, 10, 22, 31, 50, 454, 800, 92, 1478 Step 6: 6, 7, 10, 22, 31, 50, 92, 800, 454, 1478 Step 7: 6, 7, 10, 22, 31, 50, 92, 454, 800, 1478 Sorted Array: 6, 7, 10, 22, 31, 50, 92, 454, 800, 1478 -- program is finished running --