Kiến trúc tập lệnh MIPS

Giáo viên hướng dẫn: Nguyễn Xuân Minh
Võ Tấn Phương
Sinh viên: Nguyễn Văn Hoàng
Nguyễn Xuân Hiến
1711400
1652192

ĐAI HOC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH

Nhóm: Nh_L03_14 — Bài tập lớn 1

Mục lục

1 Đề bài

Viết chương trình sắp xếp chuỗi số thực 20 phần tử nhập từ bàn phím dùng giải thuật Quick sort. Yêu cầu xuất kết quả từng bước chạy ra màn hình.

2 Yêu cầu

- Sử dụng tâp lệnh MIPS để hiện thực các thủ tục bên dưới.
- Thống kê số lệnh loại lệnh (instruction type) của mỗi chương trình.
- Tính thời gian chạy của chương trình.
- Code:
- Code style phải rõ ràng, có comment, nên viết theo cách gọi hàm.
- Truyền và nhận kết quả khi gọi hàm theo quy ước của thanh ghi (thanh ghi $a chứa tham số, thanh ghi $v hoặc $f chứa giá trị trả về khi gọi hàm).
- In kết quả ra màn hình để kiểm tra.
- Báo cáo:
- Trong báo cáo cần nêu rõ các dữ liệu mẫu dùng để kiểm tra.
- Báo cáo gồm có file báo cáo (không source code) định dạng .PDF (Nhom##_rp.pdf) và phần source code đi kèm (Nhom##_sc.asm).

3 Ý tưởng hiện thực

Quick sort là một trong những thuật toán chia để trị. Quick sort chia một mảng lớn của chúng ta thành hai mảng con nhỏ hơn: mảng có phần tử nhỏ và mảng có phần tử lớn. Sau đó Quick sort có thể sort các mảng con này bằng phương pháp đệ quy. Các bước ý tưởng của Quick sort là:

- Chọn một phần tử để so sánh, chúng ta gọi đây là phần tử key, từ trong mảng đầu tiên của chúng ta.
- Sau đó phân vùng và sort mảng con của sau phân vùng của chúng ta làm sao cho các phần tử lớn hơn phần tử Key nằm sau(bên phải) và các phần tử bé hơn phần tử Key nằm trước(bên trái). Đây được gọi là quá trình phân vùng.
- Cuối cùng là đệ quy sử dụng các bước trên cho các mảng với phần tử bé hơn và phân tách với các phần tử lớn hơn sau khi phân vùng.

4 Chọn phần tử key (pivot)

Kỹ thuật chọn phần tử chốt ảnh hưởng khá nhiều đến khả năng rơi vào các vòng lặp vô hạn đối với các trường hợp đặc biệt. Tốt nhất là chọn phần tử chốt là trung vị của danh sách. Khi đó sau $\log_2 n$ lần phân chia ta sẽ đạt tới kích thước danh sách bằng 1 . Tuy nhiên điều đó rất khó. Có các cách chọn phần tử chốt như sau:

- Chọn phần tử đứng đầu hoặc đứng cuối làm phần tử chốt.
- Chọn phần tử đứng giữa danh sách làm phần tử chốt.
- Chọn phần tử trung vị trong 3 phần tử đứng đầu, đứng giữa và đứng cuối làm phần tử chốt.
- Chọn phần tử ngẫu nhiên làm phần tử chốt. (Cách này có thể dẫn đến khả năng rơi vào các trường hợp đặc biệt)

Trong bài tập lớn này chúng ta sẽ chọn cách thứ 2 .

5 Độ phức tạp giải thuật Quick sort

- Trường hợp tốt nhất: $O(n \log_2 n)$
- Trường hợp xấu nhất: $O(n^2)$
- Trường hợp trung bình: $O(n \log_2 n)$

6 Hướng dẫn sử dụng

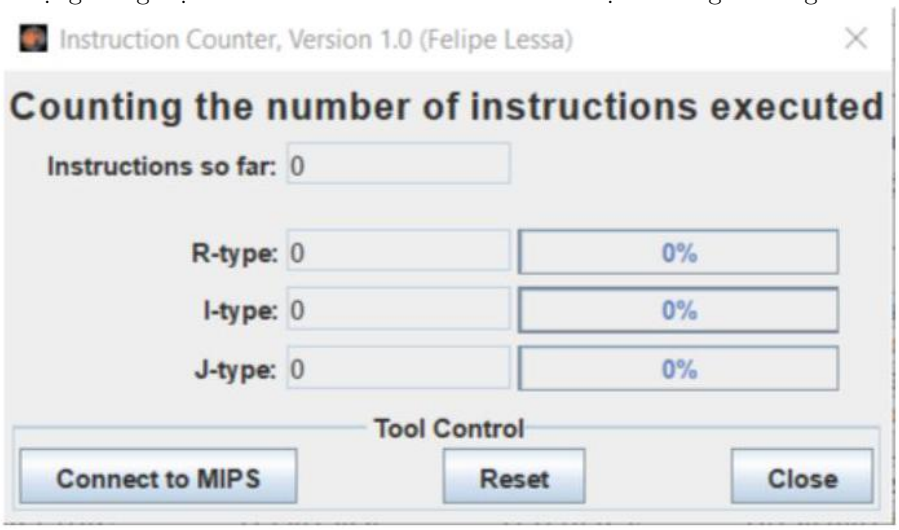
Bạn sẽ có ba cách để nhập vào một mảng:

- Cách 1: Bạn tự nhập vào một mảng số thực 20 phần tử.
- Cách 2: Ramdom một mảng số thực 20 phần tử.

Sau khi có được mảng số thực, chương trình sẽ sử dụng thuật toán quick sort để sắp xếp mảng của bạn và in ra màn hình kết quả sau khi sắp xếp.

7 Thống kê số lệnh, loại lệnh và thời gian chạy

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



Sử dụng công thức sau để tính thời gian chạy:

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

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 cyles: Tổng số chu kỳ thực thi.
- Instruction Count 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: 4GHz = trong một giây có 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 và thời gian chạy khác nhau.

Dưới đây là một số ví dụ cho thấy thời gian chạy khác nhau:

7.1 Ví dụ 1

Mảng nhập vào: [-17.6 -91.0 -18.8 27.785 .8 -38.4 35.5 -13.2 -19.3 −82.041.070.0−
90.0 − 13.2]

Tổng số lệnh: 3470

Trong đó:

- R-type: 1345 lệnh chiếm 38%
- I-type: 1927 lệnh chiếm 55%
- J-type: 198 lệnh chiếm 5%

Tính toán thời gian chạy:

CPU time = $\frac{\text{Instruction Count xCPI}}{\text{Clock rate}} = \frac{3470\text{x}1}{2.5\times10^9} = 1.388 \times 10^{-6}s = 1.388\mu\text{s}$

7.2 Ví dụ 2

Mảng nhập vào: [1.5 -3.2 4.5 -6.8 9.82 .30 .01 .03 .06 .890 .0 -100.0 156.079 .0
-65.0 89.0236 .0 -5.0 6.056.0]

Tổng số lệnh: 3542

Trong đó:

- R-type: 1375 lệnh chiếm 38%
- I-type: 1983 lệnh chiếm 55%
- J-type: 184 lệnh chiếm 5%

Tính toán thời gian chạy:

CPU time = $\frac{\text{Instruction Count x CPI}}{\text{Clock rate}} = \frac{3542\text{x}1}{2.5\times10^9} = 1.4168 \times 10^{-6}s = 1.4168\mu\text{s}$

7.3 Ví dụ 3

Mảng nhập vào bằng cách Ramdom mảng: [-28.801712 92.84820697 .89546 -
66.22838 -56.985153 −7.4053497 42.824326 73.07504 10.617569 −79.36971 62.384277 4.974922 76
51.923875.409561 − 0.920700137.3679252.0551458.778427]

Tổng số lệnh: 3400

Trong đó:

- R-type: 1332 lệnh chiếm 39%
- I-type: 1876 lệnh chiếm 55%
- J-type: 192 lệnh chiếm 5%

Tính toán thời gian chạy:

CPU time = $\frac{\text{Instruction Count }\times\text{CPI}}{\text{Clock rate}} = \frac{3400\text{x}1}{2.5\times10^9} = 1.36 \times 10^{-6}$ s $= 1.36\mu\text{s}$

8 Code c + +

```
#include
    #include
    #define NUMBER 19
    #define Maxrand 100.0
```

```
using namespace std;

void swap(float data[], int a, int b) {
   float temp = data[a];
   data[a] = data[b];
   data[b] = temp;
}
void printArray(float data[], int low, int high) {
   for (int i = low; i <= high; i++) {
      cout << data[i] << " ";
   }
   cout << endl;
}
void quickSort(float Data[], int l, int r) {
   if (l <= r) {
      float key = Data[(l + r) / 2];
      int i = l;
      int j = r;
      while (i <= j) {
         while (Data[i] < key)
            i++;
         while (Data[j] > key)
            j--;
         if (i <= j) {
            swap(Data, i, j);
            i++;
            j--;
         }
      }
      if (l < j)
         quickSort(Data, l, j);
      if (r > i)
         quickSort(Data, i, r);
   }
}
int main() {
   int nhap;
   cout << "1. Nhap vao mang." << endl
      << "2. Ramdom mot mang." << endl
      << "Lua chon phuong an: ";
   cin >> nhap;
   float data[NUMBER + 1];
   switch (nhap) {
      case 1: {
         int i = 0;
         while (i <= NUMBER) {
            cout << "Nhap vao phan tu thu " << i << ": ";
            cin >> data[i];
            i++;
         }
         break;
      }
      case 2: {
         int i = 0;
         srand(time(NULL));
         while (i <= NUMBER) {
            data[i] = -Maxrand + (Maxrand + Maxrand) * rand() / RAND_MAX;
            i++;
```

```
            }
            break;
        }
        default:
            cout << "Ban lua chon sai!" << endl;
    }
    cout << "Mang cua ban truoc khi su dung quickSort:" << endl;
    printArray(data, 0, NUMBER);
    quickSort(data, 0, NUMBER);
    cout << "Mang cua ban sau khi su dung quickSort:" << endl;
    printArray(data, 0, NUMBER);
    return 0;
}
```

Tài liệu tham khảo

[1] MIPS Technologies, Inc MIPS32TM Architecture For Programmers Volume II: The MIPS32 ^{TM} Instruction Set. 2003.

[2] Hợp ngữ MIPS: https://vietcodes.github.io/algo/mips.

[3] Sắp xếp nhanh: https://vi.wikipedia.org/wiki/Sắp_xếp_nhanh.

[4] QuickSort: https://www.geeksforgeeks.org/quick-sort/.