ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH

Bài tập lớn

KIẾN TRÚC TẬP LỆNH

GVHD: NGUYỄN THÀNH LỘC

SINH VIÊN THỰC HIỆN

STT	MSSV	HỌ	TÊN	% ĐIỂM BTL	ĐIỂM BTL	GHI CHÚ
1	2211361	Hồ Minh	Hưng			
2	2211761	Nguyễn Lê Gia	Kiệt			
3	2211384	Tô Thế	Hưng			

TP. HỒ CHÍ MINH, NĂM HỌC 2023 -2024

Mục lục

1. Đề bài:

Cho một chuỗi 50 phần tử số thực. Sử dụng hợp ngữ assembly MIPS, viết thủ tục sắp xếp chuỗi đó theo thứ tự tăng dần theo giải thuật quick sort.

2. Yêu cầu:

- Sử dụng tập lệnh MIPS để hiện thực các thủ tục bên dưới.
- Thống kê số lệnh, loại lệnh (instruction type) của mỗi chương trình.
- Tính và trình bày cách tính thời gian chạy của chương trình trên máy tính kiến trúc MIPS có tần số 3.4 GHz.
- Code:
  - Code style phải rõ ràng, có comment, phân hoạch công việc theo từng hàm, CHỈ DÙNG LỆNH MIPS CHUẨN.
  - Truyền nhận và trả kết quả khi gọi hàm theo quy ước của thanh ghi (thanh ghi $a argument, thanh ghi $v giá trị trả về khi gọi hàm).
  - Xuất kết quả để kiểm tra.
- Báo cáo ngắn
  - Báo cáo ngắn gồm các phần sau:
    * Trình bày giải pháp hiện thực.
    * Giải thuật (nếu có).
    * Ghi rõ các trường hợp test (test-cases ít nhất 30 cases) và kết quả các test cases.
    * Mỗi testcase thống kê số lệnh chạy được (R, I, J) và thời gian thực thi của mỗi testcase (CPI=1 cho 1 lệnh MIPS chuẩn, nên trình bày ở dạng bảng).
- File báo cáo đặt tên theo dạng KTMT_assignment_DE_x_Nhom_x.
- Submit bài tập lớn: file báo cáo (file pdf) và source code (mỗi nhóm đại diện 1 thành viên nộp bài)

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:

- 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. Giải thuật (bằng C++):

```
#include <iostream>
#include <random>
#define NUMBER 49
#define Maxrand 100.0
using namespace std;

void swap(float data[], int i, int j)
{
    float temp = data[i];
    data[i] = data[j];
    data[j] = 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;
```

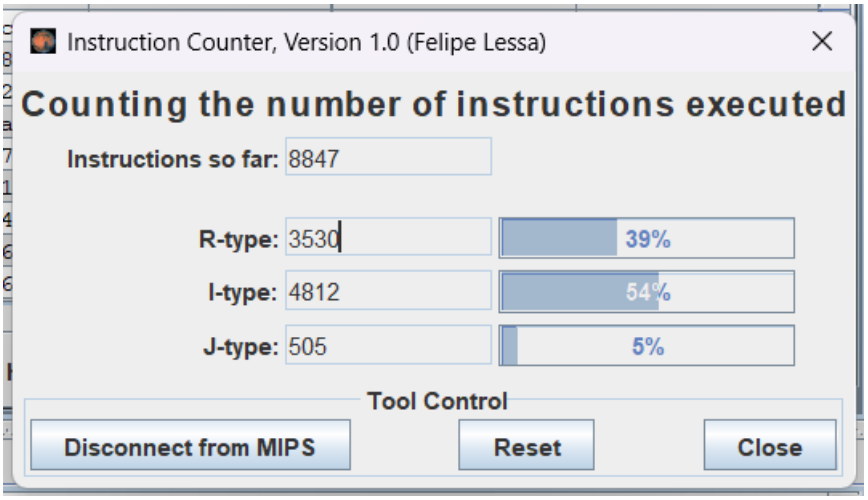
```
        int j = r;
        while (i <= j){
            while (data[i] < key)
                i++;
            while (data[j] > key)
                j--;
            if (i <= j){
                swap(data, i, j);
                i++;
                j--;
            }
        }
        if (l < j)
            quickSort(data, l, j);
        if (r > i)
            quickSort(data, i, r);
    }
}

int main()
{
    int nhap;
    cout << "1. Nhap vao mang." << endl
        << "2. Random 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;
}
```

7. Test case:

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



Công thức tính thời gian CPU Time:

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

Trong đó:

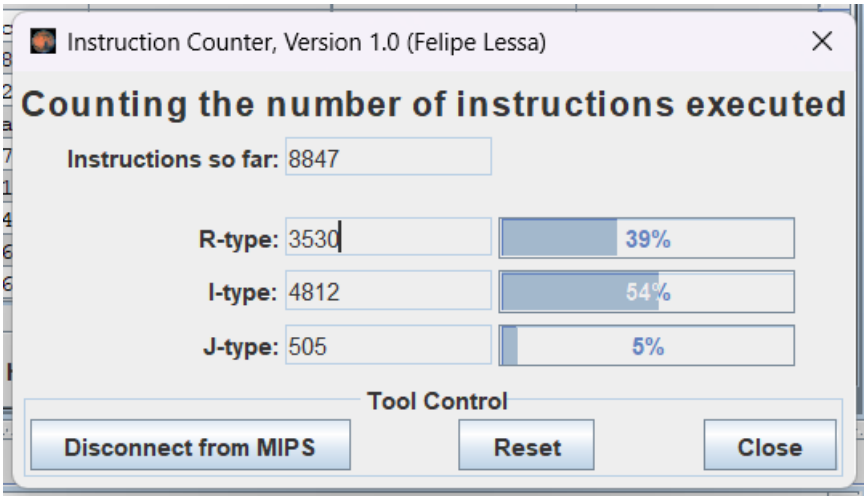
- CPU time là thời gian xử lý của chương trình (không tính thời gian giao tiếp I/O, thời gian chờ ...).
- CPU Clock cyles: Tổng số chu kỳ thực thi.
- Instruction Count là tổng số lệnh thực thi của chương trình.
- CPI (cycle per instruction) là số chu kỳ thực thi trên một lệnh.
- Clock rate là số chu kỳ thực thi trên một giây hay còn gọi là tần số, ví dụ: 4GHz = trong một giây có $4x10^9$ dao động.

Dưới đây là ví dụ minh họa cho cách chương trình hoạt động trên MARS 4.5:

- Input: 7.2383194 -1.4366903 -7.8685703 -7.900481 8.895193 -8.500227 -2.9825983 -2.7107916 -3.9063478 9.417728 -0.16656399 -9.2636795 3.3676186 -5.3474402 -4.687922 -4.678196 -6.075802 -5.590484 3.914132 -2.741767 2.6748219 -2.4845943 4.91486 -2.8761506

-1.7499809 -3.6293793 -7.1935215 -5.854442 9.841999 6.70393 1.5993824 -0.74398804 0.9972496 9.668304 2.8807936 1.3736706 -6.121951 2.5333405 5.3237925 7.96031 -4.3913317 -1.8299942 1.8449326 -8.454372 -4.9858785 -3.8094902 -5.2890453 1.4833946 -9.446802 -4.4337487

- Output: -9.446802 -9.2636795 -8.500227 -8.454372 -7.900481 -7.8685703 -7.1935215 -6.121951 -6.075802 -5.854442 -5.590484 -5.3474402 -5.2890453 -4.9858785 -4.687922 -4.678196 -4.4337487 -4.3913317 -3.9063478 -3.8094902 -3.6293793 -2.9825983 -2.8761506 -2.741767 -2.7107916 -2.4845943 -1.8299942 -1.7499809 -1.4366903 -0.74398804 -0.16656399 0.9972496 1.3736706 1.4833946 1.5993824 1.8449326 2.5333405 2.6748219 2.8807936 3.3676186 3.914132 4.91486 5.3237925 6.70393 7.2383194 7.96031 8.895193 9.417728 9.668304 9.841999



Tổng số lệnh: 8847

- Lệnh R: 3530
- Lệnh I: 4812
- Lệnh J: 505
- Thời gian CPU:

$$CPU\ time = \frac{CPU\ Clock\ cycles}{Clock\ rate} = \frac{Instruction\ Count\ .\ CPI}{Clock\ rate}$$
$$= \frac{8847\ .\ 1}{3{,}4\ .\ 10^9} = 2602{,}058824(\mu s)$$

Về chi tiết 29 test case còn lại, xem tại đây.

Tài liệu tham khảo

[1] MIPS Technologies, Inc *MIPS32™ Architecture For Programmers Volume II: The MIPS32™ Instruction Set.* 2003
[2] Hợp ngữ MIPS: https://vietcodes.github.io/algo/mips.
[3] Sắp xếp nhanh: https://vi.wikipedia.org/wiki/S%E1%BA%AFp_x%E1%BA%BFp_nhanh.
[4] QuickSort: https://www.geeksforgeeks.org/quick-sort/.